FaceFindr.
Face Recognition Project

Department of Electronics, Telecommunications, and Informatics
University of Aveiro

Group 3
Advisor: Professor António Neves
Collaborator: Marilia Maia e Moura

Alexandre Martins (103552) alexandremartins@ua.pt,
Bruno Gomes (103320) brunofgomes@ua.pt,
Daniel Frank (96127) danielbueno@ua.pt,
Diogo Silva (104341) diogobranco.as@ua.pt,
João Gonçalves (98287) joao.mendes.goncalves@ua.pt,
Miguel Marques (103162) miguelgoncalvesmarques@ua.pt

November 2023

Abstract

This report examines the integration of facial recognition technology into a platform aimed at improving photo management for photographers. It addresses the growing necessity for image organization tools, fueled by the continuous growth of online visual content. Focusing on simplifying image filtering and fetching, the website uses facial recognition-based filtering to make use of more efficient photo management.

Through an in-depth exploration of scientific papers, articles, and existing technologies, the report validates the project's objectives by drawing comparisons between its aspirations and established facial recognition methodologies. It presents the initial platform demonstration using different facial recognition algorithms and analyze different projects to extract valuable information.

Based on stages like setting goals, exploring technology, and talking with stakeholders, the report lists what the system needs. What we learned from them helps organize who will use it, diagrams showing how it'll be used, and details about what the system should do.

The report ends by highlighting the importance of overcoming challenges while optimizing system efficiency and explore implications, both in terms of effectiveness and ethics, of integrating facial recognition into photo management.

Acknowledgements

We would like to thank the availability, support and valuable suggestions of both our Adivsor Professor António Neves as well as our main stakeholder Marília Maia e Moura.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 23

API Application programming interface. 15

AWS Amazon Web Services. 29, 31, 32

BRIEF Binary Robust Independent Elementary Features. 4, 8

CDN Content Delivery Network. 31, 32

CF Correlation Filter. 7

CNN Convolutional Neural Network. 4, 6–8, 13

CRUD Create, Read, Update, Delete. 14

CSRF Cross-site Request Forgery. 14

DOM Document Object Model. 16

EC2 Elastic Compute Cloud. 32

FAIR Facebook's AI Research. 13

GDPR General Data Protection Regulation. 29

HOG Histogram of Oriented Gradients. 4, 8

KPCA kernel PCA. 4, 8

LBP Local Binary Pattern. 4, 8

LDA Linear Discriminant Analysis. 4, 8

LPQ Local Phase Quantization. 4, 8

MTCNN Multi-Task Cascaded Convolutional Neural Networks. 4–7

MVC Model-view-controller. 14

ORM Object-Relational Mapping. 14, 15

PCA Principal Component Analysis. 4, 8

REST Representational State Transfer. 15

SIFT Scale-invariant feature transform. 4, 8

SQL Structured Query Language. 14

SSD Single-shot Detector. 4–7

SURF Speeded-Up Robust Features. 4, 8

SVM Support Vector Machine. 4, 8

UI User Interface. 2, 16, 19

WCAG Web Content Accessibility Guidelines. 30

XSS Cross-site Scripting. 14

Chapter 1

Introduction

1.1 Motivation

This project stemmed from several key motivations. Firstly, the surge in demand for online photographic content underscored the need for tools that streamline image organization and search functionalities. Moreover, it aimed to assist photographers in storing and managing photo albums from large gatherings efficiently. Equally important was the objective to aid users in reducing the time spent searching for their photos by employing pattern and facial recognition algorithms, thereby expediting the process of locating desired images.

1.2 Goals

The main goal for this project is to provide a platform to help photographers manage photos of large gatherings of people by letting them upload, edit, publish and share photos/photo albums while making use of facial recognition, depth estimation and image processing algorithms.

In a first phase the project members will test various face recognition algorithms and implement them into a simple prototype to evaluate their performance and identify possible challenges and limitations of the algorithms used.

In the second phase image processing will be used to enhance the photos provided to be better used by the face recognition algorithms and will also be tested to ensure the level of performance required for the project.

In the third phase the members of the project will research and test depth estimation algorithms to help facial recognition algorithms with their tasks and test if

the results are reliable.

In the final phase all the other three phases will be integrated into a web-based application

1.3 Dashboard/Website

We developed a prototype for our project, which serves as a preliminary representation of the final product. The majority of decisions and selections concerning the User Interface (UI) and its color scheme were based on the findings and recommendations outlined in [SL12]. Before initiating development, we designed mockups to inform this process, and the upcoming figures will visually illustrate these conceptual designs. While we acknowledge the potential for future modifications, the prototype closely aligns with the established use cases and incorporates both functional and non-functional requirements that have been identified thus far. This prototype serves as a foundational model for the subsequent stages of development, providing a tangible manifestation of our conceptualization.

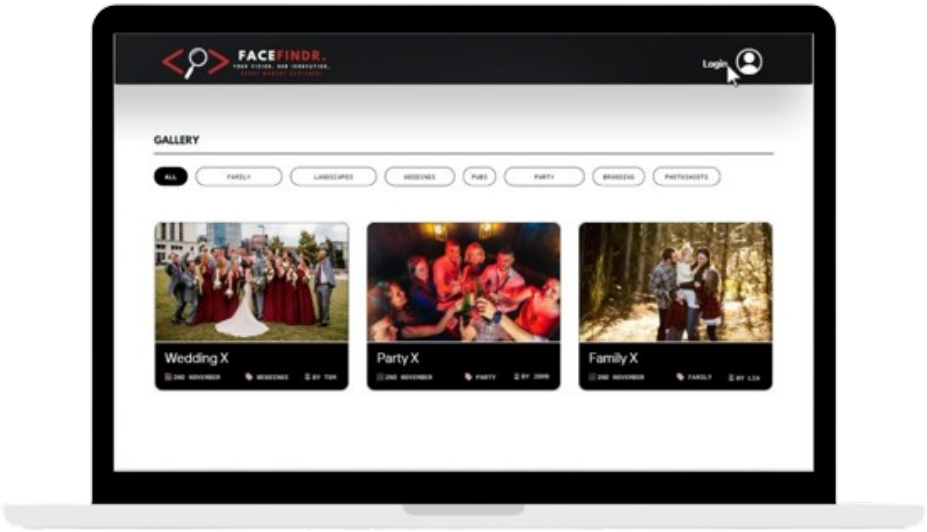


Figure 1.1: Landing Page

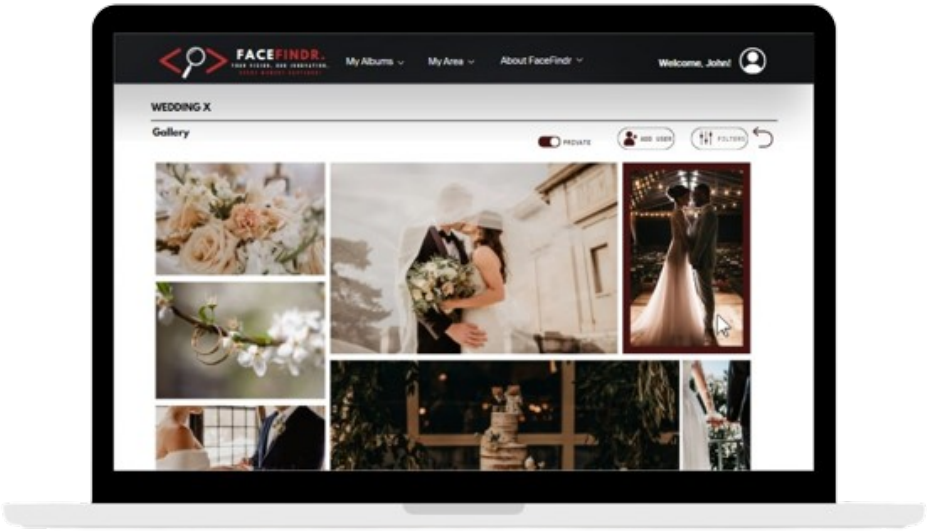


Figure 1.2: Album view and photo selection for edition

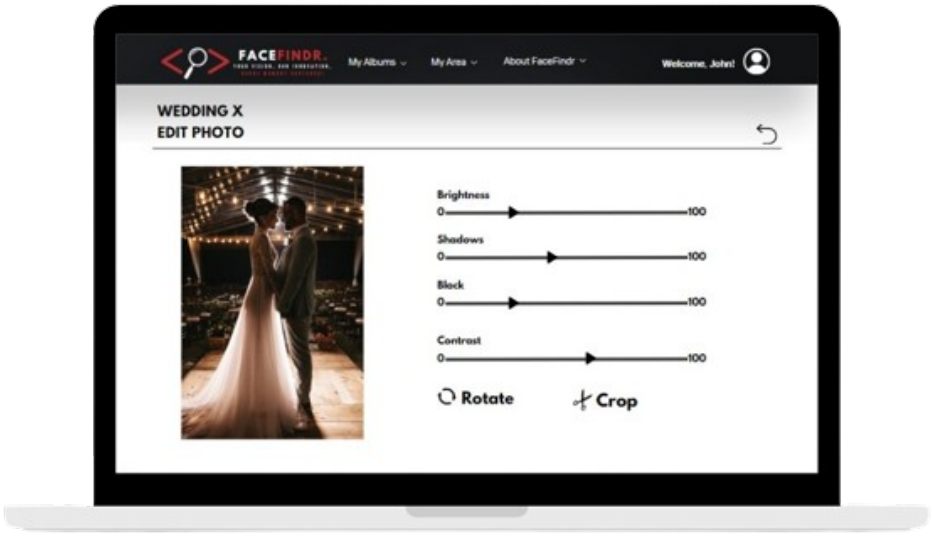


Figure 1.3: Photo edition

Chapter 2

State of the Art

The purpose of this section is to provide a comprehensive overview on the current state of knowledge and technology in the field relevant to our project. Only by **examining different projects and articles** related to the field in which we are working, can we progress and surpass the several difficulties that we will face during the development of our system.

2.1 Introduction

In order to begin the development of our platform we had to compile some scientific papers and articles with the sole purpose of learning how facial recognition works. After carefully analysing them, we identified similarities between our project and existing technologies. This comparative analysis not only validated the relevance of our objectives but also elucidated potential challenges that others in the field have grappled with. This thorough analysis allowed us to test some facial recognition algorithms, thus creating the first demonstration of our platform.

2.2 Face recognition algorithms

A face recognition system is a technology designed to identify or verify personal information from digital images or video files. Due to its numerous advantages, face recognition finds applications across various domains such as mobile payment, password protection, finance, attendance access control, and more. The technology plays a crucial role in these fields. Consequently, studying face recognition algorithms holds significant importance. Face recognition, as a comprehensive term, is summarised in two primary tasks [WD21]:

- **Face detection:** it refers to whether there is a face in the input image. If there

is, it locates the face with bounding box coordinates.

- **Face recognition:** if there is a face in the picture or data stream, it will be compared with the existing face in the database to display the task information.

2.2.1 Face Detection

At the core of various face analysis algorithms lies **face detection**, serving as the fundamental component for tasks such as **face alignment**, **face modeling**, **face reillumination**, **face recognition**, **face authentication**, **head tracking**, **facial expression tracking/recognition**, and **gender/age recognition**. When presented with a digital image, the objective of face detection is to identify the location and dimensions of all objects within the image that pertain to a specified class[wJ19]. Other objects, such as trees, buildings, and animals, are disregarded in the digital image. This process can be viewed as a specific instance of object class detection.

To further understand the operations of these algorithms, we chose to investigate and compare the Single-shot Detector (SSD)[2] and Multi-Task Cascaded Convolutional Neural Networks (MTCNN)[ZLG20] algorithms, which are two algorithms that represent substantial advancements in the field of computer vision, with **SSD** excelling in **general object detection** scenarios and **MTCNN** specializing in **face detection**.

- Comparison and Analysis of SSD and MTCNN Algorithms

The SSD algorithm is classified as a one-stage representative detection algorithm. Its underlying principle is: When given an input image, SSD initially resizes it to a dimension of 300x300. To prevent distortion caused by resizing, gray bars are added to the image. SSD employs multiple layers to promptly identify objects within the input image. The grayed image is partitioned into six grids of varying sizes: 38x38, 19x19, 10x10, 5x5, 3x3, and 1x1. Lower-resolution layers, such as 3x3, are utilized for detecting large objects, while higher-resolution layers, like 38x38, are employed for smaller objects. If the object's center falls within a grid region, the object's position is determined by the corresponding grid point. In figure 2.1, the detection of cats in the specified area serves as an example, illustrating the specific steps of SSD[WD21]:

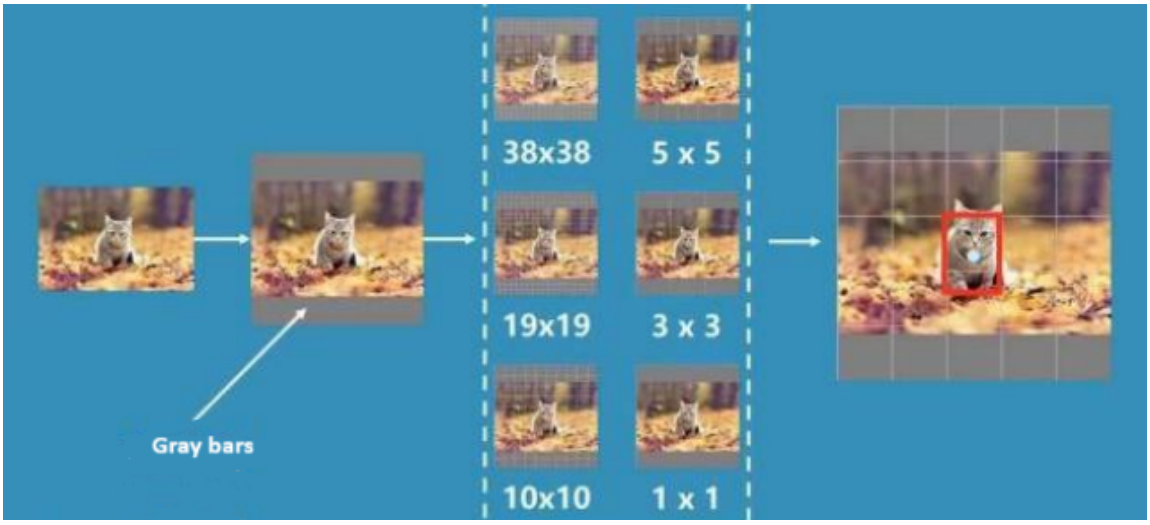


Figure 2.1: Image testing to show SSD specific steps [WD21]

MTCNN [ZLG20] is a multi-stage algorithm designed for face detection. It can be succinctly described in four stages: Initially, to identify faces of various sizes, MTCNN generates image pyramids with different scales by resizing the input images. Subsequently, two CNNs, namely "P-net" and "R-net", are employed to filter out a significant number of windows, progressively improving accuracy. Finally, the "O-net" stage is responsible for creating the ultimate bounding box and determining the positions of the five key facial landmarks [WD21].

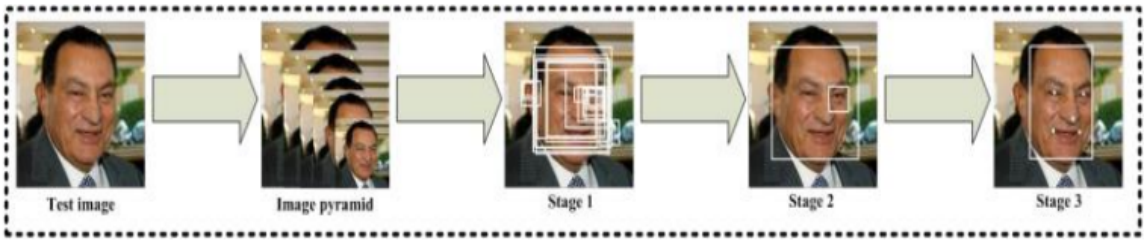


Figure 2.2: Process of each stage of MTCNN [WD21]

- Experimental comparison of complex and multi-character scenes [1]

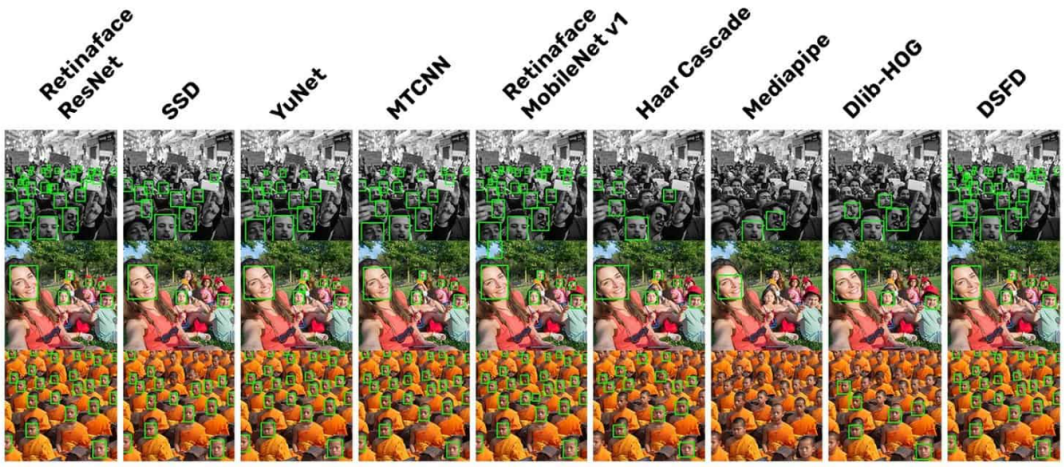


Figure 2.3: Testing of several algorithms [1]

As anticipated, SSD and MTCNN algorithms exhibit distinct performance characteristics in face detection, with each having its own strengths and weaknesses. Consequently, a significant portion of our development process will involve testing multiple algorithms to determine the most suitable one for specific situations.

2.2.2 Face Recognition

During the face recognition process, the features extracted from the background in the feature extraction step are examined and compared with recognized faces stored in a specific database. The two main applications of this process are called **identification** and **verification**. In the **identification** step, a test face undergoes comparison with a group of faces in the pursuit of determining the most likely match. During the **verification** step, a test face is compared with a known face in the database in order to make the acceptance or rejection decision. Correlation Filters(CFs) and Convolutional Neural Networks(CNNs) are known to effectively address this task.

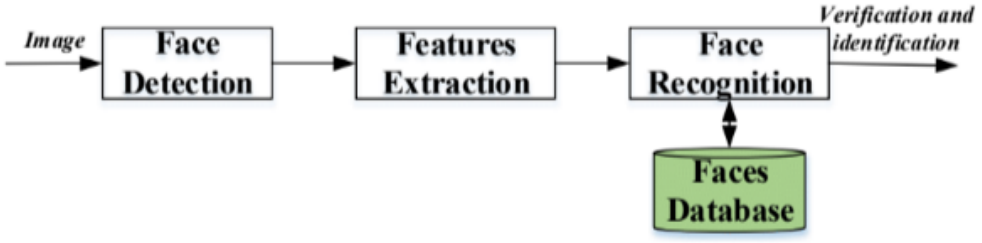


Figure 2.4: Face recognition steps[KJAFA20]

- Classification of Face Recognition Systems

Comparing the face recognition system with other biometric systems such as the eye, iris or fingerprint recognition systems, we can say that the first one is not 100% reliable, in fact, it is far from it. Additionally, this biometric system has many challenges that stem from diverse sources. Recognition in controlled environments has reached saturation. However, the challenge persists in uncontrolled environments due to significant variations in lighting conditions, facial expressions, age, dynamic backgrounds, and more. This paper presents a survey where we examine the latest advanced face recognition techniques designed for both controlled and uncontrolled environments, utilizing diverse databases.

Numerous systems are employed for human face identification in both 2D and 3D images. This review paper categorizes these systems into three approaches based on their detection and recognition methods (see Figure 2.5): **(1) local**, **(2) holistic (subspace)**, and **(3) hybrid approaches**. The first approach categorizes facial features without considering the entire face. The second approach utilizes the complete face as input data and projects it into a small subspace or correlation plane. The third approach integrates local and global features to enhance face recognition accuracy[KJAFA20].

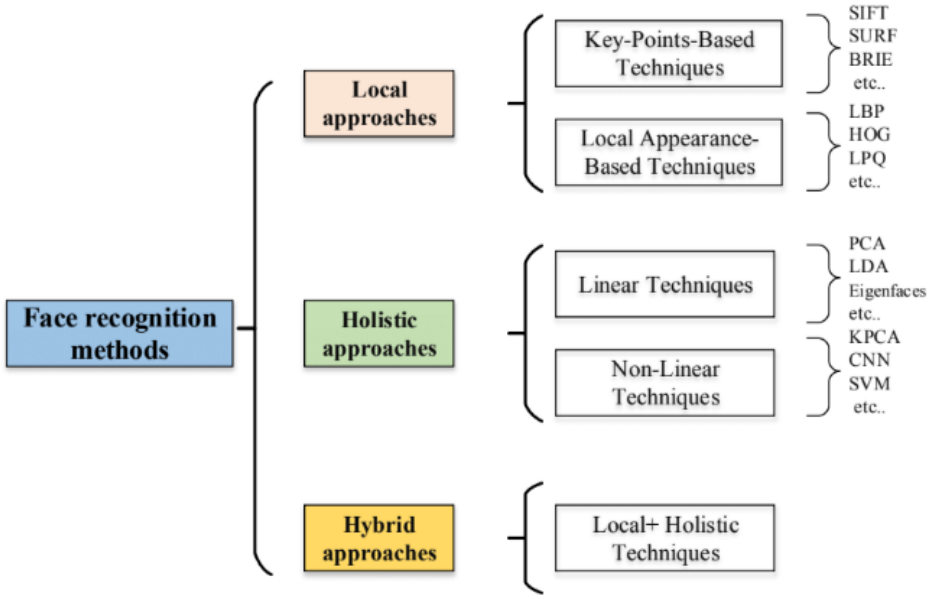


Figure 2.5: Face recognition methods: SIFT, SURF, BRIEF, LBP, HOG, LPQ, PCA, LDA, KPCA, CNN, SVM [KJAFA20]

Considering the abundance of available face recognition methods, it is essential to precisely define the context in which we are developing our project.

2.2.3 Depth Estimation

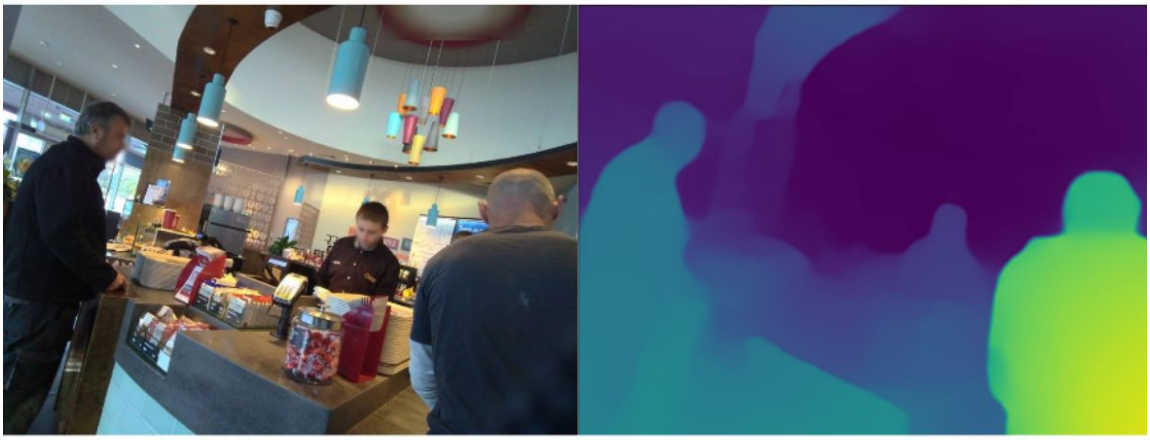
Depth Estimation is the task of measuring the distance of each pixel relative to the camera. Depth is extracted from either monocular (single) or stereo (multiple views of a scene) images[3].

To identify the most relevant objects in the context of a scene, Ferreira et al. [4] proposes the use of salient object detection maps. Saliency maps identify the most visually significant regions in an image by assigning a saliency value to each pixel in the image, based on its features such as color, texture, and contrast. These high-saliency regions contribute the most to understanding the scene depicted in an image.

Nevertheless, given the intricate nature of lifelog images, saliency maps often exhibit suboptimal performance compared to simpler and more posed images. Lifelogs frequently present challenges such as difficult lighting conditions, blurriness, or clutter, which can impact the accurate calculation of saliency values. As a viable alternative, depth maps have been investigated for identifying objects of interest in lifelog images. Depth maps demonstrate greater effectiveness in handling diverse illumination settings and complex object layouts, making them a suitable substitute for saliency maps [LLS16].

Depth maps offer insights into the distance of objects from the camera, facilitating the recognition of objects in closer proximity to the lifelogger. Such objects are more likely to be crucial for comprehending the scene, given their location in the foreground. For each lifelog image, a depth map was generated using a machine learning model [5, 6], estimating visual depth values for individual pixels. The threshold was established by computing the average of these depth values. Objects with a majority of pixels exceeding the threshold were categorized as belonging to the foreground plane, while the remainder were considered part of the background. This approach demonstrated greater efficacy in identifying relevant objects compared to saliency maps, as illustrated in Fig. 2.6.

(a) Saliency map of a lifelog image



(b) Depth map of a lifelog image

Figure 2.6: Example of saliency map for non-lifelog and lifelog egocentric images and depth map for lifelog egocentric image

2.3 Related Projects

During our research process, we came across several projects that shared plenty of characteristics with our system, some of them were developed by well-known companies and others were more simple, however, we decided to take this opportunity to explore the details of each one of them, in order to gather as much information as we can, to help us steer our development process strategically, with a keen eye on surpassing existing challenges and optimizing our system for efficiency and accuracy.

To keep track of the articles that our team reads throughout the development

of our project, we decided to use *Mendeley*. Additionally, we deemed it essential to bookmark certain open-source *GitHub* repositories housing relevant projects.

Below, we highlight the systems that aligned better with our project. It should be noted that the last two projects allowed us to do our first testing of some images, and therefore gave us some insights on how these algorithms worked for different occasions. They are not 100% reliable, each algorithm has its own strengths and weaknesses. We understood that, if we want to build a powerful system we need to use a framework that utilizes several face recognition libraries.

Project/System	**Goals**	**Software**
runporto.fotop.pt[14]	Allow photographers to upload sports events photos.	Undisclosed
	Allow users of the platform to filter out photos where they are not present.	
Apple Photos	Allow *Apple* devices users to manage and organize videos and photos, through categorization of faces, objects, and landscapes.	Undisclosed
	Provide users of Apple devices with a vast array of editing tools for photos and videos.	
Adobe Lightroom	*Adobe Lightroom* is specifically designed for photographers and provides a range of features for organizing, editing, and sharing digital photographs.	Undisclosed

Table 2.1: Related Projects

2.4 Software Frameworks

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. *OpenCV* was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception the commercial products. Being an Apache 2 licensed product, *OpenCV* makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning

algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, and so forth[7].

Our **initial testing** utilized exclusively *OpenCV*. We developed a concise Python script that processed images, detecting and outlining faces within each image with a square drawn by the algorithm. This served as an indicator of the presence of faces in the pictures.

Dlib

dlib is a modern C++ toolkit that contains machine learning algorithms and tools for a variety of applications. It is designed to be efficient, portable, and versatile, making it suitable for a range of tasks, including computer vision, machine learning, image processing, and facial recognition[8].

Key features:

- **Machine Learning Algorithms:** *dlib* provides a collection of powerful machine learning algorithms, including support vector machines, deep learning tools, and clustering techniques;
- **Computer Vision:** The library includes functionalities for computer vision tasks such as image processing, object detection, and shape prediction;
- **Facial Recognition:** *dlib* is renowned for its facial recognition capabilities. It offers pre-trained models for facial landmark detection and face recognition, making it a popular choice in the computer vision community;
- **Cross-Platform:** *dlib* is designed to be cross-platform, making it compatible with various operating systems. This portability allows developers to deploy their applications seamlessly across different platforms;
- **Optimization:** This library is optimized for performance, leveraging modern C++ features to achieve efficiency in both speed and memory usage.

Not only is this library widely used for detecting facial landmarks, enabling applications such as facial expression analysis and emotion recognition, but it is also suitable for object tracking in video streams, making it a powerful tool to use in contexts like surveillance and motion analysis. Furthermore, to test this software, we created an additional testing script. In this script, we analyzed an input image by comparing each face within it with a small database of known faces. Subsequently, the script assigned a label, representing the person's name, to each identified face in the analyzed image.

All things considered, *dlib* is a versatile and efficient C++ library that has gained popularity for its robust implementation of machine learning and computer vision algorithms. Its ease of use and active community support make it a valuable asset for developers working on projects ranging from facial recognition to general-purpose machine learning applications.

DeepFace

DeepFace is a facial recognition system designed by Facebook's AI Research (FAIR) lab. This framework employs deep learning techniques to perform face recognition tasks with high accuracy. It gained attention for its ability to recognize faces in images with significant variations in pose, lighting conditions, and facial expressions[9].

Key features:

- **Deep Learning Architecture:** *DeepFace* utilizes a deep neural network architecture, leveraging Convolutional Neural Network (CNN) for facial feature extraction and identification;
- **Facial Verification and Recognition:** This framework is capable of both face verification (determining if two faces belong to the same person) and face recognition (identifying individuals from a database of known faces);
- **Robust to variations:** *DeepFace* is designed to handle challenges such as pose variations, illumination changes, and non-uniform expressions, making it very robust in real-word scenarios;
- **Metric Learning:** *DeepFace* employs metric learning techniques to map faces into a common feature space, facilitating accurate face comparisons.

The applications of this framework closely resemble those of the *dlib* library mentioned earlier. Utilizing this framework alongside *FastAPI* and *React*, we developed the initial demonstration of our project, which was a simple website. The website prompts users to upload a photo, and in return, it retrieves all photos containing that user.

To conclude, *DeepFace* represents a significant advancement in facial recognition technology. Its robustness in handling challenging conditions has made it a notable framework in the field.

Django

Django is a high-level, open-source web framework for *Python* that follows the Model-view-controller (MVC) architectural pattern. It is designed to make web development fast, secure, and scalable. There are some important features of *Django* that should be noted[10]:

1. **MVC Architecture:**
    - **Models:** *Django's* Object-Relational Mapping (ORM) allows developers to interact with databases using *Python* objects. Models define the structure of the database tables;
    - **Views:** Views handle the logic and processing of data. They receive input from the user, interact with models, and return responses;
    - **Templates:** Templates handle the presentation layer, defining how data is displayed in the user interface;
2. **Admin Interface:**
    - *Django* provides an automatic admin interface for managing site content. Developers can perform Create, Read, Update, Delete (CRUD) operations on the application's models without having to write custom admin views;
3. **Security features:**
    - *Django* includes built-in security features to protect against common web vulnerabilities, such as Structured Query Language (SQL) injection, Cross-site Scripting (XSS), and Cross-site Request Forgery (CSRF);
4. **REST Framework:**

- *Django* comes with the Representational State Transfer (REST) framework, an extension for building Web Application programming interface (API). It includes tools for serialization, authentication, and viewsets, making it easier to create RESTful APIs;

5. **Scalability:**

   - While *Django* is well-suited for rapid development, it is also scalable and can handle the demands of larger applications. It supports deployment on popular web servers and cloud platforms.

Initially, our team began developing the project demonstration using *Django*. However, it quickly became evident that this framework wasn't ideal for meeting our requirements, particularly in scenarios where handling multiple face recognition requests concurrently was necessary, so we decided to switch to *FastAPI*.

FastAPI

FastAPI is a modern, fast , web framework for building APIs with *Python* 3.7+ based on standard *Python* type hints. It is designed to be easy to use, high-performance, and to provide automatic interactive documentation[11].

1. **Fast and Asynchronous:**

   - *FastAPI* is built on top of *Starlette* and *Pydantic*, leveraging *Python*'s **'asyncio'** for asynchronous request handling. This makes it well-suited for I/O-bound tasks and scenarios where high concurrency is essential;

2. **Type Hints and Data Validation:**

   - *FastAPI* uses *Python* type hints for function parameters and return types. This not only serves as documentation but also enables automatic data validation and serialization using *Pydantic* models;

3. **Integrations:**

   - *FastAPI* can be easily integrated with other popular *Python* libraries and frameworks. It supports usage with tools like ORM, databases, and authentication systems.

We have chosen this framework, because we believed it is well suited to our needs, since we need to develop an application which has high concurrency requirements.

React

React is an open-source JavaScript library developed by *Facebook* for building user interfaces, especially for single-page applications where the UI needs to be highly dynamic. React follows a declarative and component-based approach, making it easier to develop interactive and reusable UI components[12].

1. **Component-Based Architecture:**
    - *React* is built around the concept of components, which are modular, reusable building blocks for UI elements. Components encapsulate their own logic, state, and rendering, making the code more modular and maintainable;
2. **Unidirectional Data Flow:**
    - *React* follows a unidirectional data flow, meaning data flows in a single direction from parent components to child components. This helps in maintaining a predictable and manageable state;
3. **Document Object Model (DOM):**
    - *React* introduces a virtual DOM, which is a lightweight copy of the actual DOM. When the state of a *React* component changes, *React* first updates the virtual DOM and then efficiently updates the actual DOM, minimizing the performance impact;
4. **React Router:**
    - *React* Router is a popular library for handling navigation in React applications. It allows developers to create single-page applications with multiple views or pages;
5. **React Hooks:**
    - Hooks are functions that enable functional components to have state and lifecycle features, which were traditionally associated with class components. They allow developers to use state and lifecycle methods in functional components.

React will be our software of choice to build our web application, its component-based architecture, virtual DOM, and reactive updates make it a powerful and efficient library for developing dynamic user interfaces.

Exploration

The following frameworks were explored and analyzed in order to gather information to create a short demonstration of the pretended platform:

1. **A Lightweight Face Recognition and Facial Attribute Analysis**[9]

    - The goal of this framework is to develop a hybrid face recognition framework wrapping several state-of-the-art models it utilizes the following software, VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace, Dlib and SFace

2. **Face-Recognition**[13]

    - The goal on this framework is to develop a simple face recognition script that identifies and classifies all faces in an image this one uses the OpenCV, Dlib software libraries

2.5 Conclusion

In summary, the State of the Art phase has significantly influenced our project's direction. Through an exploration of diverse facial recognition projects and relevant articles, we've acquired a deep understanding of the underlying algorithms and identified key challenges in the field. Notably, our analysis of projects such as *run-porto.fotop.pt*, *Apple Photos*, and open-source solutions like *Face-Recognition* and *DeepFace* have offered strategic insights for our development.

This phase has not only validated the relevance of our project objectives but has also enabled practical learning. Testing facial recognition algorithms in real-world scenarios has revealed nuances, emphasizing the importance of a versatile framework that integrates multiple libraries for optimal performance.

As we progress, the insights gained from this phase serve as a valuable foundation. They inform our approach to overcoming challenges and optimizing the efficiency and accuracy of our platform. The careful examination of existing projects not only inspires our work but also ensures that our project evolves with a nuanced understanding of both advancements and potential pitfalls within the facial recognition domain.

Chapter 3

System Requirements

This section presents the system requirements specification, as a result of the first phase of the prototype development. The following subsections begin with a description on the requirements elicitation process, followed by actors classification, use-case diagrams, functional and non-functional requirements.

3.1 Requirements Elicitation

This specification process consists of three essential phases. Initially, the system's main goals were assessed to delineate its scope and establish the system boundaries. The subsequent phase involved an in-depth investigation into the state-of-the-art, exploring projects, studies, and technologies similar to the proposed system. This research aimed to uncover the latest and most innovative works in the field, learning from both successes and challenges encountered. The objective is to distill the best practices for the proposed system.

In the final phase, several informal meetings were conducted with Marilia, a professional photographer and our main stakeholder, and António Neves, the project advisor. These discussions provided invaluable insights into the requirements and expectations for the proposed system.

Insights Gathered:

- The significance of efficiently managing a large number of event photos.
- The challenges faced by photographers in organizing and tracking people in event photos.

- The importance of labeling photos with context information for documentation and analysis.
- The need for tools to enhance photos, including cropping, noise removal, and color balance improvement.
- The utility of tools for estimating the depth of people in photos for query purposes.

Subsequent decisions were made to guide the development process, with a primary focus on meeting Marilia's specific UI requirements:

- Implement a user-friendly interface with lighter color schemes to enhance visual appeal and reduce strain.
- Modify the main page to showcase different events rather than individual faces, providing a more event-centric view.

3.2 Actors

Our system caters to two types of users: **non-registered** and **registered**. Non-registered users have limited access, primarily viewing public albums and requesting download permissions while on the other hand, registered users can take on different roles.

3.2.1 Non-Registered User:

- **Public Viewer:** Individuals who visit the site without registering. They can view public albums and request download permissions.

3.2.2 Registered User:

- **Super Admin:** The person who created the album. Can add new users to the album, define supervisors and album owners, and has all the permissions from other roles
- **Photographer:** Can add and edit photos. Can be promoted to a supervisor.
- **Supervisor:** Can add new users to the album.
- **Participant:** Can download specific photos or the entire album. Can be promoted to an album owner.

- **Album Owner:** Can add new participants (non-photographers) and define visibility (public or private).

3.3 Use Cases

UC1: Browse Public Albums

- **Actors:** Non-registered User
- **Description:** Non-registered users can browse public albums.
- **Trigger:** Non-registered user accesses the site.
- **Main Flow:**
  1. User navigates to the public albums section.
  2. User views a list of available public albums.
  3. User selects a public album to view.
- **Alternative Flow:** None
- **Postconditions:** User views the selected public album.

UC2: Request Download Permissions

- **Actors:** Non-registered User
- **Description:** Non-registered users can request download permissions for specific photos.
- **Trigger:** Non-registered user wants to download a photo.
- **Main Flow:**
  1. User selects a photo in a public album.
  2. User clicks on the "Request Download" button.
  3. System prompts the user to register or log in.
  4. User registers or logs in.
  5. User submits the download request.
- **Alternative Flow:** None
- **Postconditions:** System records the download request for review.

UC3: Create Album

- **Actors:** Registered User (Super Admin)
- **Description:** Registered User can create a new album becoming the Super Admin of that album.
- **Trigger:** Registered User wants to create a new album.
- **Main Flow:**
  1. Registered User navigates to the album creation section.
  2. Registered User provides details for the new album (name, visibility).
  3. Registered User confirms and creates the album (becoming the Super Admin of that album).
- **Alternative Flow:** None
- **Postconditions:** System creates a new album with the specified details.

UC4: Promote Photographer to Supervisor

- **Actors:** Registered User (Super Admin), Registered User (Photographer)
- **Description:** Super Admin can promote a Photographer to become a Supervisor.
- **Trigger:** Super Admin wants to give a Photographer the role of Supervisor (e.g. to the team leader of a group of Photographer) .
- **Main Flow:**
  1. Super Admin selects the option to promote a Photographer.
  2. Super Admin designates the chosen Photographer as the new Supervisor.
- **Alternative Flow:** None
- **Postconditions:** The specified Photographer is promoted to the new Supervisor.

UC5: Promote Participant to Album Owner

- **Actors:** Registered User (Super Admin), Registered User (Participant)
- **Description:** Super Admin can promote a participant to become the new Album Owner.
- **Trigger:** Super Admin wants to define a participant as Album Owner.
- **Main Flow:**
  1. Super Admin selects the option to promote a participant.
  2. Super Admin designates the chosen participant as the new Album Owner.
- **Alternative Flow:** None
- **Postconditions:** The specified participant is promoted to the new Album Owner.

UC6: Upload Photo

- **Actors:** Registered User (Photographer)
- **Description:** Photographers can upload one or more photos to an existing album.
- **Trigger:** Photographer wants to add new photos to an album.
- **Main Flow:**
  1. Photographer selects an existing album to upload photos.
  2. Photographer selects and uploads one or more photos.
  3. Photographer provides additional details for the uploaded photos (title, description, tags).
- **Alternative Flow:**
  - If there are any errors during the upload (e.g. unsupported file format), the system notifies the Photographer, and the process is aborted.
- **Postconditions:** New photos are added to the selected album with provided details.

UC7: Identify People in Photos using Computer Vision

- **Actors:** Registered User (Photographer)
- **Description:** Photographers can utilize computer vision techniques to identify people in uploaded photos.
- **Trigger:** Photographer wants to identify individuals in the photos.
- **Main Flow:**
  1. Photographer selects the option to identify people in a specific photo.
  2. The application uses computer vision techniques to analyze and identify individuals in the photo.
- **Alternative Flow:** None
- **Postconditions:** The application provides information about individuals identified in the photo.

UC8: Label Photos with Context Information

- **Actors:** Registered User (Photographer)
- **Description:** Photographers can manually or automatically label photos with context information about the scene and people.
- **Trigger:** Photographer wants to add context information to a photo.
- **Main Flow:**
  1. Photographer selects the option to add context information to a specific photo.
  2. Photographer manually adds labels or allows the application to automatically generate labels using Artificial Intelligence (AI).
- **Alternative Flow:** None
- **Postconditions:** The photo is successfully labeled with context information.

UC9: Enhance Photos

- **Actors:** Registered User (Photographer)
- **Description:** Photographers can use tools provided by the application to enhance photos, including removing noise and improving color balance.
- **Trigger:** Photographer wants to improve the quality of a photo.
- **Main Flow:**
  1. Photographer selects the option to enhance a specific photo.
  2. Photographer uses tools to remove noise and improve color balance in the photo.
- **Alternative Flow:** None
- **Postconditions:** The photo is successfully enhanced.

UC10: Estimate Depth of People in Photos for Querying

- **Actors:** Registered User (Photographer)
- **Description:** Photographers can use tools provided by the application to estimate the depth of individuals in photos for querying purposes.
- **Trigger:** Photographer wants to estimate the depth of people in a photo.
- **Main Flow:**
  1. Photographer selects the option to estimate the depth in a specific photo.
  2. Photographer uses tools to determine the depth of individuals, such as identifying first or second plane faces and people alone or in groups.
- **Alternative Flow:** None
- **Postconditions:** The depth of individuals in the photo is successfully estimated.

UC11: Add Users to Album as a Supervisor

- **Actors:** Registered User (Supervisor)
- **Description:** Supervisors can add new users to the album.
- **Trigger:** Supervisor wants to include new users in the album.
- **Main Flow:**
  1. Supervisor selects the option to add users.
  2. Supervisor provides details (email addresses) of the new users, and their roles (Participant or Photographer).
  3. System sends invitations to the specified users.
- **Alternative Flow:** None
- **Postconditions:** New users are successfully added to the album.

UC12: Download Album

- **Actors:** Registered User (Participant)
- **Description:** Participants can download specific photos or an entire album.
- **Trigger:** Participant wants to download photos or an album.
- **Main Flow:**
  1. Participant selects an album or specific photos for download.
  2. Participant initiates the download process.
  3. System generates a download link.
  4. Participant clicks the link to download the content.
- **Alternative Flow:** None
- **Postconditions:** Participant downloads the selected content.

UC13: Add Participants to Album

- **Actors:** Registered User (Album Owner)
- **Description:** Album Owners can add new participants to their albums.
- **Trigger:** Album Owner wants to include new participants in the album.
- **Main Flow:**
  1. Album Owner selects the option to add participants.
  2. Album Owner provides details (email addresses) of the new participants.
  3. System sends invitations to the specified participants.
- **Alternative Flow:** None
- **Postconditions:** New participants are successfully added to the album.

UC14: Define Visibility Settings

- **Actors:** Registered User (Album Owner)
- **Description:** Album Owners can set visibility settings for their albums.
- **Trigger:** Album Owner wants to define who can view the album.
- **Main Flow:**
  1. Album Owner selects the option to define visibility settings.
  2. Album Owner specifies whether the album is public or private.
- **Alternative Flow:** None
- **Postconditions:** Visibility settings for the album are successfully defined.

Important Note on Super Admin Capabilities

While not explicitly detailed in separate use cases to avoid redundancy, the Super Admin (album creator) holds extensive permissions, encompassing all tasks from other roles like adding users, editing photos, and downloading content. These capabilities are inherent to the Super Admin role, streamlining album management.

3.4 Functional Requirements

- **User Authentication:** The system must provide a user authentication mechanism to allow non-registered users to register and log in, enabling access to additional features.
- **Browse Public Albums:** The system must allow non-registered users to browse public albums.
- **Request Download Permissions:** The system must enable non-registered users to request download permissions for specific photos in public albums.
- **Create Album:** Registered users, must be able to create a new album.
- **Promote Photographer to Supervisor:** Super Admins must have the capability to promote a Photographer to the role of Supervisor within an album.
- **Promote Participant to Album Owner:** Super Admins must be able to promote a Participant to the role of Album Owner within an album.
- **Upload Photo:** Photographers should have the ability to upload one or more photos to an existing album.
- **Identify People in Photos using Computer Vision:** Photographers must be able to utilize computer vision techniques to identify people in uploaded photos.
- **Label Photos with Context Information:** Photographers should be able to manually or automatically label photos with context information.
- **Enhance Photos:** Photographers must be provided with tools to enhance photos, including removing noise and improving color balance.
- **Estimate Depth of People in Photos for Querying:** Photographers must have tools to estimate the depth of individuals in photos for querying purposes.
- **Add Users to Album as a Supervisor:** Supervisors must be able to add new users to an album.
- **Download Album:** Participants should have the capability to download specific photos or an entire album.

- **Add Participants to Album as a Album Owner:** Album Owners must be able to add new participants to their albums.
- **Define Visibility Settings:** Album Owners should be able to define visibility settings for their albums, specifying whether the album is public or private.
- **Comprehensive Super Admin Capabilities:** The Super Admin shall have comprehensive permissions, encompassing all functional requirements.

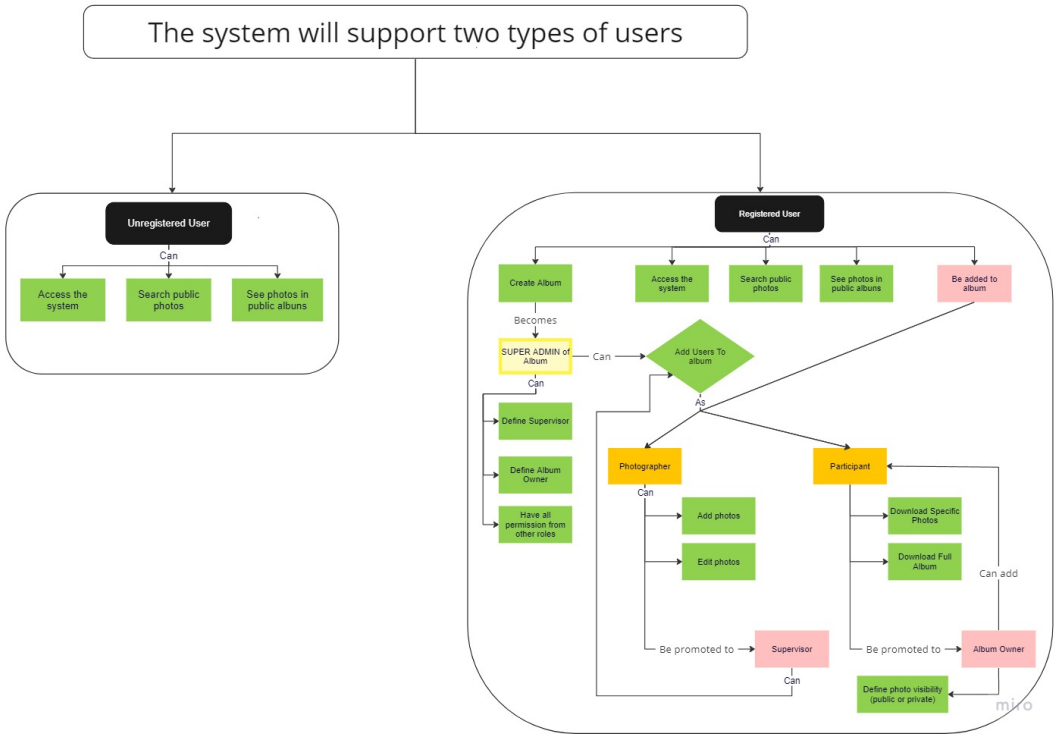


Figure 3.1: Functional Requirements

3.5 Non-Functional Requirements

- **User Interface:** The system shall have a user-friendly and modern interface to enhance the overall user experience.
- **Browser Compatibility:** The system shall be compatible with all modern browsers, including but not limited to Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, Brave and Opera GX.

- **Compliance:** The system shall adhere to the General Data Protection Regulation (GDPR) to ensure the privacy and protection of user data.
- **Confidentiality:** The system shall follow the security principle of confidentiality, ensuring that data is accessible only to authorized parties. Access controls and encryption mechanisms shall be implemented to enforce confidentiality.
- **Integrity:** The system shall follow the security principle of integrity, guaranteeing that data is not tampered with or degraded during or after submission. Hash functions and data validation techniques shall be employed to maintain data integrity.
- **Availability:** The system shall follow the security principle of availability, ensuring that information is available to authorized users when needed. Redundancy measures and failover mechanisms shall be implemented to minimize downtime.
- **Infrastructure:** The system shall have a cloud infrastructure, utilizing scalable and reliable cloud services such as Amazon Web Services (AWS) or Microsoft Azure. Cloud infrastructure will facilitate scalability, flexibility, and efficient resource management.
- **Image Optimization:** All images displayed in the front end shall be optimized for fast loading times and improved performance. Image compression techniques and responsive design principles shall be applied to ensure optimal viewing across various devices.
- **Performance:** The system shall be designed to deliver high performance, with quick response times for user interactions. Performance testing and optimization shall be conducted regularly to meet user expectations.
- **Scalability:** The system architecture shall be scalable to accommodate a growing number of users, albums, and photos. Scalability testing shall be performed to assess the system's ability to handle increased loads.
- **Documentation:** Comprehensive and up-to-date documentation shall be maintained, including user manuals, system architecture documentation, and code documentation. Documentation shall facilitate system understanding, maintenance, and future development.

- **Accessibility:** The system shall be designed to be accessible to users with disabilities, following the Web Content Accessibility Guidelines (WCAG) standards. Accessibility features such as screen reader compatibility and keyboard navigation shall be implemented.

- **Logging and Auditing:** The system shall maintain detailed logs of user activities and system events. Audit trails shall be established to trace changes and actions performed within the system, ensuring accountability and security monitoring.

Chapter 4

System Architecture

This section is a deeper dive into the idea of the System Architecture to achieve the desired results to the website. The following figure is a short diagram explaining the planning and the resources required for the system architecture

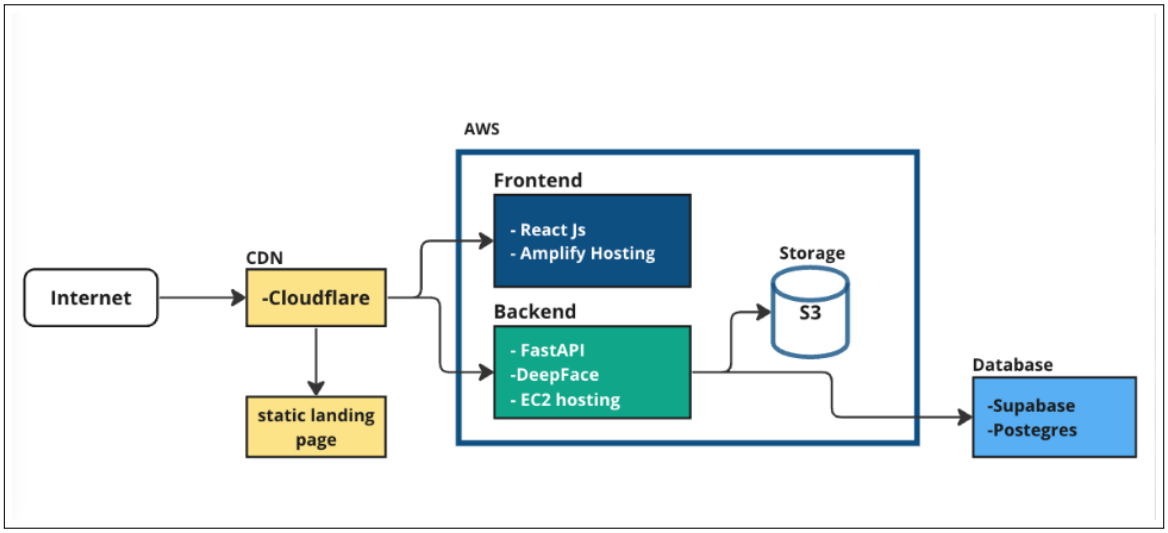


Figure 4.1: System Architecture

Breaking down the previous diagram, it is clear that the system will follow a cloud infrastructure, aiming to ensure high performance and availability, a security environment for sensible data, and easy scalability. In this perspective, **AWS** is the main chosen provider, having presented an extensive range of features and services very well documented and tested in large-scale applications.

- **Cloudflare:** Last in this chain of technologies, but the first layer after the user's request, is the **Content Delivery Network (CDN)**. In order to reduce the

delay in communication, caused by the large physical distances between servers and clients, we decided to use a **CDN** in our system, which should reduce the page load time and the bandwidth costs, increase our content availability, and improve the website security. **Cloudflare** is the system selected to do that function, which also provides a static page hosting service, that will be used to serve the product landing page.

- **Frontend:** The software frontend is one of the most relevant parts of the system since the user experience will act directly in the user's ability to conclude the expected features. As previously mentioned, the interface should be intuitive and performative, which is a challenge when leading with a large number of images - by nature heavy content to the browser. Thus, following the approach of building single-page applications, the user interface will be written in **React**, a modern JavaScript library based on components. The app will be hosted by **AWS Amplify**.

- **Backend** The challenges are in the implementation of the face recognition algorithms in a performative way, ensuring a high certainty rate and a solid and sustainable code base. Considering that cutting-edge technologies in the market are developed in Python, which has demonstrated high performance in processing massive amounts of data, that was the first selected programming language of the project. The server will be hosted in **AWS Elastic Compute Cloud (EC2)**.

- **Storage and Database:** A relational database will ensure a better management of the entity's relations within the system - besides the atomicity, consistency, isolation, and durability provided by the relational model. Willing to keep the platform at the top-notch, **PostgreSQL** will be used as a service provided by **Supabase**, to store all of the software and users' information - to store the images, **Amazon S3**.

Chapter 5

Conclusion

In our exploratory project, a significant focus has been on extensive research and requirements gathering. Initially, challenges arose due to the absence of direct contact with Marilia Maia e Moura and limited knowledge about photography algorithms, delaying the project's initiation.

After establishing communication with a seasoned photographer, our research trajectory took a pivotal turn. We shifted our focus to exploring technologies, libraries, and tools relevant to photography.

Meetings and insights from the photographer's experiences, along with the useful guidance of Professor António Neves, played a crucial role in shaping our understanding. This collective input offered perspectives on the practical application scenarios of the project. Recognizing the versatility of our application, we anticipate a wide range of future use cases across various photography scenarios and environments.

At this stage, we have solidified our objectives and established clear guidelines for development. With thorough research and analysis completed, we are well-prepared to transition to the application development phase. The insights gained during our exploration have deepened our understanding of photographers' needs and provided a comprehensive view of the existing landscape in photography technology.

In essence, we are now positioned to move from the exploratory phase to the development phase with a robust foundation and a clear sense of direction, thanks to valuable input from both the experienced photographer and Professor António Neves.

Bibliography

[1] Inference Comparison under Various Conditions. https://learnopencv.com/what-is-face-detection-the-ultimate-guide/. Accessed: 28-11-2023.

[10] Django Overview . https://www.djangoproject.com/start/overview/. Accessed: 30-11-2023.

[11] FastAPI Features . https://fastapi.tiangolo.com/features/. Accessed: 30-11-2023.

[12] React Reference Overview . https://react.dev/reference/react. Accessed: 30-11-2023.

[13] Face-Recognition . https://github.com/techwithtim/Face-Recognition. Accessed: 30-11-2023.

[14] runporto . https://runporto.fotop.pt/fotos/. Accessed: 30-11-2023.

[2] How single-shot detector (SSD) works? . https://developers.arcgis.com/python/guide/how-ssd-works/. Accessed: 28-11-2023.

[3] Depth Estimation . https://paperswithcode.com/task/depth-estimation. Accessed: 30-11-2023.

[4] Caption Analysis with Proposed Terms, Image of Objects, and Natural Language Processing . https://doi.org/10.1007/s42979-022-01322-7. Accessed: 30-11-2023.

[5] Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer . https://doi.org/10.48550/arXiv.1907.01341. Accessed: 30-11-2023.

[6] Vision Transformers for Dense Prediction . https://doi.org/10.48550/arXiv.2103.13413. Accessed: 30-11-2023.

[7] About OpenCV . https://opencv.org/about/. Accessed: 30-11-2023.

[8] dlib . http://dlib.net/. Accessed: 30-11-2023.

[9] DeepFace . https://github.com/serengil/deepface. Accessed: 30-11-2023.

[KJAFA20] Yassin Kortli, Maher Jridi, Ayman Al Falou, and Mohamed Atri. Face recognition systems: A survey. *Sensors*, 20(2), 2020.

[LLS16] Meng Li, Howard Leung, and Hubert P. H. Shum. Human action recognition via skeletal and depth based feature fusion. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 123–132, New York, NY, USA, 2016. Association for Computing Machinery.

[SL12] C. Sik-Lányi. *Choosing effective colours for websites*, pages 600–621. Elsevier Inc., 2012.

[WD21] Yi Wang and Xinwei Duan. Research on face recognition algorithm based on deep learning. In *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, pages 1139–1142, 2021.

[wJ19] Yue wu and Qiang Ji. Facial landmark detection: A literature survey. *International Journal of Computer Vision*, 127, 02 2019.

[ZLG20] Ning Zhang, Junmin Luo, and Wuqi Gao. Research on face detection technology based on mtcnn. In *2020 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pages 154–158, 2020.