Dissertation

Intelligent User Authentication

Christodoulos Constantinides

University of Cyprus

DEPARTMENT OF COMPUTER SCIENCE

December 2020

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Intelligent User Authentication

Christodoulos Constantinides

Supervisor

Prof. Andreas Pitsillides

Co-Supervisor

Dr. Marios Belk

The individual thesis submitted for partial fulfillment of the requirements for obtaining the degree of Computer Science, Department of Computer Science, University of Cyprus

December 2020

Acknowledgments

I would like to express my sincere gratitude to Professor Andreas Pitsillides for trusting me to work with him for my thesis.

A big thanks to Dr. Marios Belk for the coordination and help he provided and for introducing me in this amazing field of Human-Computer Interaction. I am also thankful to Argyris Constantinides who has been providing me constant feedback and technical knowledge.

My thesis is dedicated to my parents Maria and Chrysostomos who always supported me with everything I needed throughout my life, but also to my wonderful siblings Stella and Nicolas.

Abstract

In the era of information technology, there is a plethora of systems that allow users to interact with data. These data can be of varying importance. In fact, some of them require high security because of the criticality of the data they contain. Security of these systems is of highest importance, since they deal with very sensitive data that can cause a lot of problems if they get leaked or manipulated. In this context, user authentication is a key component in the security mechanism of a system, and refers to the process in which the user, through a particular action, proves that he is genuine in order to gain access to the system and his data. Systems often impose strong authentication policies with the goal to enhance their security. Even though users are required to create stronger passwords, in an attempt to scaffold memorability, they often employ bad security practices. In fact, when it comes to security, the weakest link is the human interacting with the system. For this reason, authentication mechanisms should be designed to be more usable in order to encourage the user to follow the security guidelines. Usable systems adapt to each user based on their characteristics, making the interaction with the system seamless and providing a positive experience to the user.

To further enhance security in systems that deal with data of critical importance, an additional factor has been introduced to further strengthen security. This additional factor is usually a One-Time Password (OTP) that is sent to the user's mobile device or email. As a result, the user has to provide something that he owns which is his email or mobile phone in order to get the one time password.

The Multi-Factor Authentication (MFA) methods included in this system are the Push Notifications, QR Code and Time-based One-time Password (TOTP) authentication. The analytics component uses image analysis techniques to provide tools that can be used for picture passwords complexity analysis. In order to extract features that describe the complexity, various analytics from each image are extracted, such as the objects present, their position, semantic labels that describe the image content and the color histogram.

Integration testing of the multi-factor authentication component in an existing graphical user authentication mechanism revealed improvement in system's security and user's trust. Evaluation of the graphical password analytics component provides evidence that object

detection algorithms used for informed brute-force can drive the design of a graphical password strength meter.

Table of Contents

1. Introduction

1.1 Thesis Overview

Technological advances, especially the growth in computing performance, enabled the rise of systems dealing with large amounts of data. With this rise of information systems, the importance of information security is greater than ever before. A basic component of their security mechanisms is user authentication. There are three main types of authentication: knowledge-based, token-based and biometric based. In this thesis, we focus on knowledge-based authentication mechanisms. The most common knowledge-based user authentication techniques used are textual passwords. However, textual passwords suffer from serious vulnerabilities [1, 29]. To combat this threat, websites enforce strict password creation policies. Such policies involve passwords containing a combination of lower case, upper case, special symbols and numbers. However, due to the plethora of accounts that a user owns, as well as due to complex passwords that aren't memorable, users either create the same password in many accounts or different passwords and store them electronically or physically, increasing the risk for a password compromise. A promising alternative are picture passwords. Research has shown that they exploit the picture superiority effect [30] and the finding that humans have greater ability recalling visual information as opposed to recalling verbal information [9]. Therefore, picture passwords are considered to be more memorable than textual passwords and this can increase the security of the system by encouraging the users to create strong passwords. There are several factors that influence the security of picture passwords. One of these factors is the background image used. Even though picture passwords rely on images, there is limited knowledge on how computer vision can be used to increase security. Being able to infer insights using computer vision techniques within picture passwords could drive the design of assistive mechanisms that can help users create more secure passwords. Furthermore, today's systems require strong security, therefore, it is also important to design and build usable second-factor authentication solutions.

1.2 Problem statement

Based on the aforementioned, picture passwords is a promising alternative authentication method. However, it has been shown that pictures used in these passwords play an important role in the security of the system. Therefore, tools that can be used to analyze the content of the image and extract any information regarding the security of these images are necessary to increase the security of picture passwords. These tools should suggest spots to the administrator that are more likely for a user to select as part of their password. It is common for knowledge-based user authentication to be combined with token-based user authentication. For this reason, it is worth investigating the effect of combining picture passwords with token-based user

authentication. Particularly, we will investigate picture passwords in a web application with a mobile application as a second factor.

1.3 Usable Security

Usable security is a research field that has been lately gaining interest which deals with the design of user authentication mechanisms that will be usable and provide a positive user experience. The philosophy behind usable and secure systems is that by engaging the users to follow good security practices without enforcing any security policies, the system's security will increase. This is because security not only depends on the security mechanisms of the system but also on the users interacting with it. To build such a system, the designer has to take into consideration several parameters that define its usability. According to Nielsen [1] these parameters are:

Learnability: The learning curve the user needs to learn how to use the system in order to achieve his goals. The shorter the curve, the better for the usability of the system.

Memorability: The amount of time that a user remembers how to interact with the system in order to achieve his goals. The longer the amount of time, the better for the user.

Efficiency: The amount of steps needed for the user to achieve his goals. Usability designers aim for the minimal amount of steps.

Errors made: The amount of errors that a user makes when interacting with a system, their seriousness and the time needed to recover. In the optimal scenario, users interacting with the systems should make the minimal amount of errors, the errors shouldn't have serious effects on the goals of the user and the amount of time to recover from these errors should be minimal.

Satisfaction: The user after interacting with the system should have a positive experience.

1.4 Motivation

Picture passwords are getting more attention lately, because of the vulnerabilities that the textual passwords contain. Introduced on Windows 8™, Picture Gesture Authentication (PGA) [2] is offered as an authentication method. Our motivation is to build a platform that will input the images used in the graphical passwords and analyze them to determine the complexity of a picture and eventually use them in combination with the gestures drawn to estimate the complexity of the graphical user password. Estimating the complexity of the graphical user password will enable us to encourage the user to create stronger passwords.

In addition to this, we plan to design, develop, and evaluate alternative multi-factor authentication mechanisms and investigate their usability when combined with picture passwords.

The multi-factor authentication and the image analysis platform have been built and will be part of the SERUMS[1] H2020 authentication component. SERUMS is an EU funded research project with the goal to secure medical data in a patient-centric environment. The authentication component that was developed for this purpose is an adaptive system with enhanced user authentication methods that are part of the SERUMS H2020 project.

1.5 Scope of this thesis

In this thesis, we will focus on the research of state-of-the-art computer vision algorithms for object detection and semantic detection, and image processing to extract other important data out of the system such as image histograms and saliency maps. We will also design this system as an independent platform with a user interface that will embed these computer vision and image processing algorithms and also expose the endpoints as a RESTful API. The system will be then developed and evaluated. Following the same procedure, we will research for usable and secure multi-factor authentication methods. Then, we will design the system as a RESTful API, as well as design a mobile application that will be used for the second factor authentication and for handling the interaction with the server. Finally, we will implement the system and evaluate its usability.

2. Background Theory

2.1 Introduction

This chapter will discuss the theory involved in user authentication. There are three categories of user authentication: knowledge-based, inherent- and token-based. The most common authentication methods are textual passwords. In textual passwords, the user registers with a specific username and password during his first session with the system and in all the subsequent sessions he has to use the same credentials in order to authenticate. Textual passwords fall into the category of knowledge-based authentication. Inherent-based authentication is the process of checking the genuinity of the user by analyzing the input of biometric sensors. There is a huge variety of sensors, whereas the most common are: infrared,

[1] https://www.serums-h2020.org/

cameras, fingerprint sensors and microphones. Their output is passed through specialized algorithms that can classify whether the signals belong to the genuine user or not. The last category is the token-based authentication, which involves the use of a token that the user owns. Typical tokens can be ID cards, mobile phones or emails and are used by the user in order to prove his genuinity. This authentication category is commonly used as a second factor in combination with knowledge-based methods.

2.2 Knowledge-based Authentication Mechanisms

2.2.1 Textual Passwords

Textual passwords have been dominant since the appearance of computer systems. During the creation of these passwords there are policies enforcing specific types of characters on the passwords to make the user create a password not so predictable. These policies vary from system to system and this can impose a big vulnerability. The passwords that users create from these websites aren't natural words as they involve the combination of various types of characters making them harder to memorize [9]. Because users find it difficult to memorize several passwords, they tend to create the same password in various websites or store their passwords physically or digitally. This can create serious security problems because if there is a data breach in one of the user's websites and passwords are leaked, the attacker can gain access to his personal accounts in other websites.

2.2.2 Picture Passwords

Picture-based passwords are divided in two broad categories: recall- and recognition-based. In recognition-based picture passwords, the users are required to choose a set of pictures which will be his password. In recall-based picture passwords, users are required to draw a number of predefined shapes in any location they wish within the picture. There are several findings that graphical passwords are easier to memorize compared to text-based passwords: Chiasson et al. [13], Meng et al. [11], Everitt et al.[10] Belk et al. [14] investigated the effect that the grid size has on recall based picture passwords on the usability and security. Huestegeand and Pimenidis [11] investigations revealed that on face-based GUA schemes there were slightly increased login times and login failures users had to memorize their choice for a longer period while an increase in memory load had no significant impact.

2.3 Machine Learning - Computer Vision

Machine Learning is a set of algorithms that make use of data to train on and learn patterns on the data. After the training, they can predict the outcome of unseen data based on the learned patterns. Machine learning is divided into two main categories: supervised and unsupervised learning. Supervised learning are the algorithms which train on classified/labeled data, whereas in unsupervised learning, algorithms are trained on unclassified/unlabeled data.

2.3.1 Neural Networks

Neural networks are machine learning models inspired by the brain's neurons [16]. They fall in the category of supervised learning. It is a machine learning algorithm which aims to find patterns in the input data. Neural networks contain layers of interconnected neurons. Neurons are mathematical functions which collect and classify data according to the architecture of the network. For example, in Multi-Layer Perceptrons (MLP), to calculate the output of the neuron we first calculate by $y = \sum_{i=0}^{n} w_i \times x_i$ where n is the count of all neurons from the previous layer that are connected with the neuron, and then this result is passed through an activation function like the sigmoid function $\frac{1}{1+e^{-y}}$which provides nonlinearity and is helpful for solving non-trivial problems. The weights are adjusted during the training, so that the network will make its prediction correctly. This is done through a process called backpropagation [15]. Using an error function of the output of the last layer of the neural network and the expected output value and backpropagate this error from the last layer to the first one.

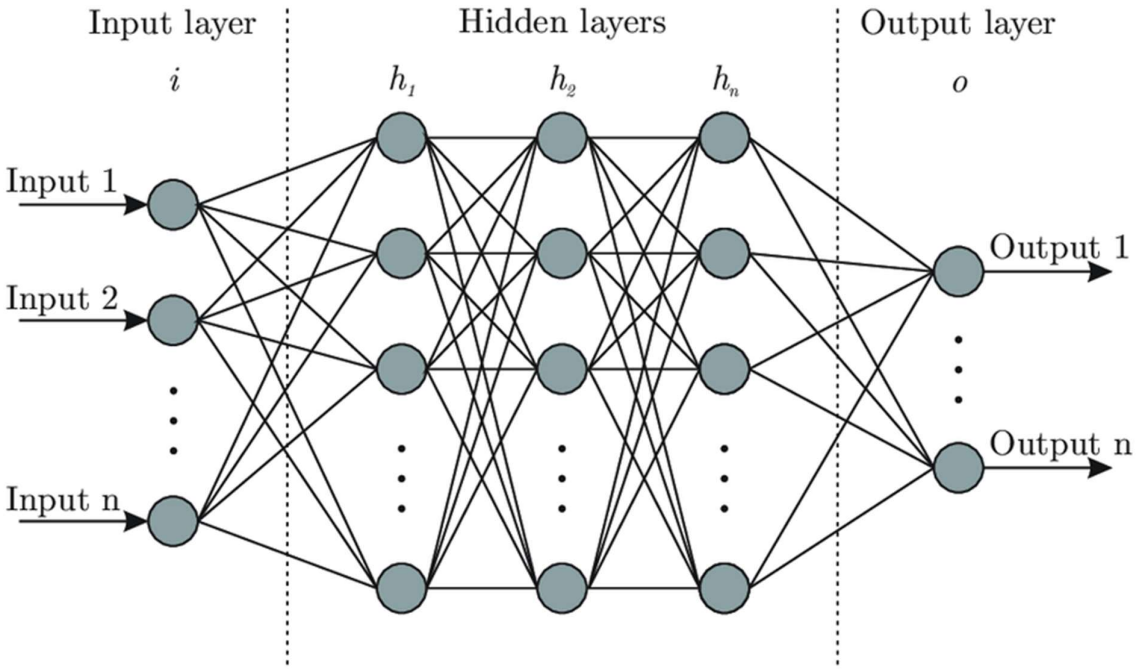


Figure 2.2 A Neural Network

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [3] is a deep learning algorithm which is widely used in computer vision. This algorithm is ideal for computer vision because it performs well when there are a lot of input features and they are related to each other. This is a property that images hold since they consist of a lot of pixels and these pixels are related with each other. The network consists of a lot of layers with repeating operations that can reduce the amount of the input features, process them and reduce the dimensions which is the procedure of feature learning. Also, as you go more deep into the network, the network learns more abstract features. For example, the network can first learn the edges of the image, in the next layer learn the shapes, after that the objects and so on. In this way, we reduce the dimensions without sacrificing accuracy. In the final layer, there is a classifier that will take the flattened matrix of the previous layer and classify the input. The convolutional neural networks consists of three basic operations that are repeated in the inner layers:

Convolution: Its purpose is to extract high-level features from an input image (e.g its edges). A filter is passed through the image and matrix multiplication is performed. Some of the hyperparameters of the network is the size of the filter and its stride which is the amount of pixels to be skipped when the filter hovers over the image. We

can also select the dimension of the output matrix from this operation. If we select a size that is equal or less than the size of the filter, then we apply no padding. Else the image can be padded with zeros to increase the output dimensions.

Activation: With an activation function we increase the non-linearity of the model. In images there is a lot of nonlinearity such as the transition between pixels, the colors or even the edges. A popular activation function is Rectified Linear Unit (RELU) which is defined as: $f(x) = max(x, 0)$, where x is each pixel of the matrix after the convolution operation.

Pooling: Pooling is used to output the most important features out from the matrix. This way, the dimensions are reduced and less computation is needed. The main types of pooling are the average pooling and the max pooling. In the max pooling, a 2d region is checked each time in the input matrix and the maximum value of that region is appended to the output matrix. The same procedure is followed for the average pooling but instead the average value is appended to the output matrix.

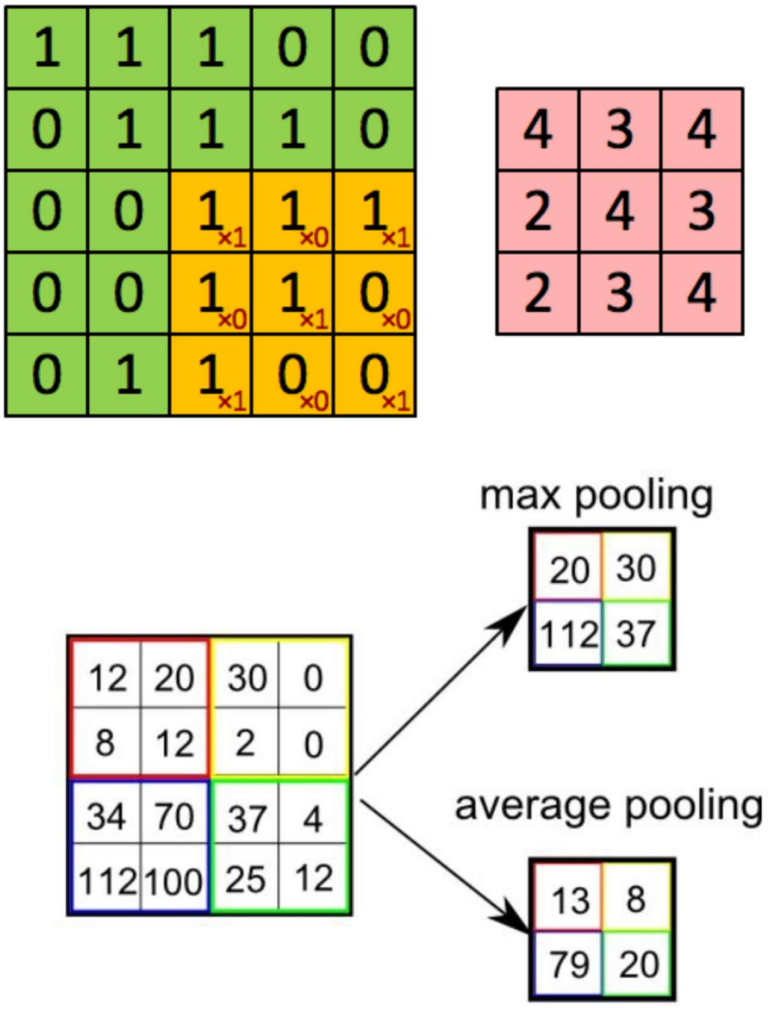


Figure 2.3 Convolution and pooling operations

2.3.3 Residual Neural Networks

By increasing the layers in a neural network, its complexity increases too. This way, even more complex cases can be modeled with a high accuracy. However there is a limit of depth that we can go and beyond that there is a drop in the accuracy of the network [17]. At a first look, this could be attributed to overfitting, but even with added dropouts and regularization there is still low accuracy. In fact, the drop of accuracy happens during training. This can be attributed to the phenomenon of "vanishing gradient". When the network updates the weights with backpropagation, the gradient is passed to previous layers and the multiplication makes the gradient really small.

Residual Neural Networks [18] allow us to build deeper neural networks without having the problem of vanishing gradients. This is achieved with residual blocks from previous layers. There is a residual connection that skips layers and avoids the extra derivatives that are responsible for vanishing the gradient. A residual block is shown in Figure 2.4. As we can see the input x is added with the output at the end of the block F(x). So, even if the gradients vanish and F(x) becomes zero, the network will still output x.

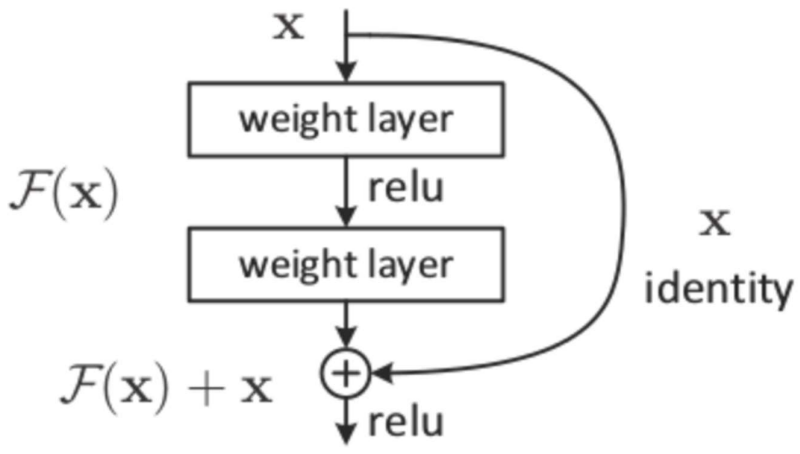


Figure 2.4 A residual block

2.4 Computer vision algorithms

2.4.1 Regions with CNN features (R-CNN)

Regions with CNN features (R-CNN) [19] inputs an image, uses selective search [4] to extract approximately 2000 regions candidate for having an object. Selective search generates some initial segments, after that uses a greedy algorithm to combine similar regions into larger ones and reshapes these regions to become a square and inputs them in a convolutional neural network which classifies extracts features for each region and then finally an SVM [31] classifier classifies the features extracted by the CNN. Additionally, there is a regressor which outputs four values that represent the offset of the bounding box. The purpose of these offsets is to make more accurate bounding boxes because the regions generated by selective search may contain an incomplete object.

2.4.2 Fast R-CNN

R-CNN was slow because each of the proposed regions (~2000) that was produced by selective search was given into the deep network for object detection. Fast R-CNN [7] takes a similar approach, but instead of inputting the regions of the image to the network, the whole image is given once to the network and a feature map is built. Selective search runs again on the image and for each proposal, a region of interest pooling (RoI) layer extracts a fixed length feature vector from the feature map. This is because the fully connected layer which inputs this feature vector only accepts a fixed size length. After a series of fully connected layers, the output is given to a softmax layer which outputs for each class the probability and to another regressor which outputs a refined bounding box position for each of the predefined classes.

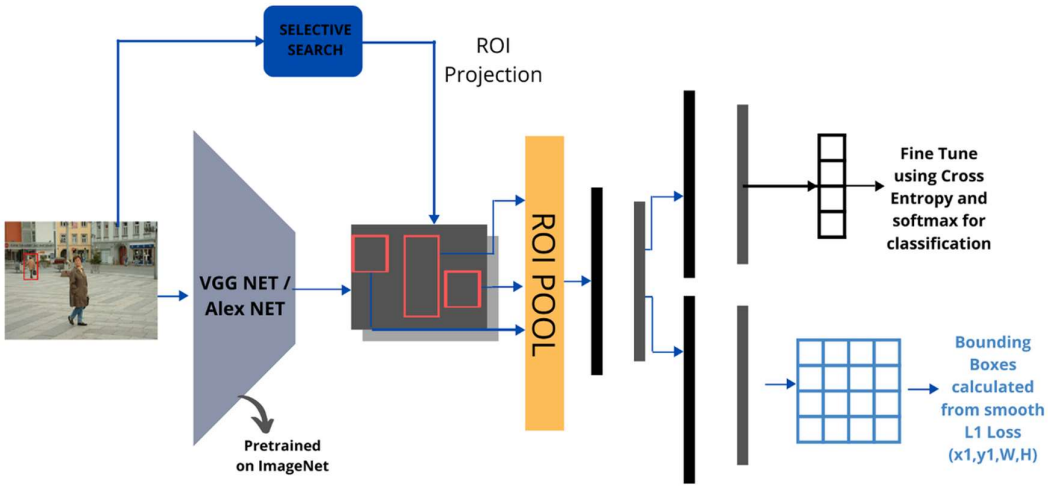


Figure 2.5 Fast R-CNN architecture

2.4.3 Faster R-CNN

Region proposal algorithm used in Fast R-CNN was the main bottleneck in the network because it was a CPU based algorithm and needed a few seconds to run. In order to fix this, another convolutional network is used to generate the region proposals called Region Proposal Network (RPN).

Faster R-CNN [5] consists of the RPN for proposing the regions and the Fast R-CNN to detect objects in the proposed region.

The region proposal network performs the following steps:

Input image is resized and given to a shared CNN both for the region proposal and object detection to generate the feature map.

We pass through the feature map produced by the last layer of the CNN using a square sliding window and each region is mapped to a lower dimensional feature. This feature is given in a box-regression layer to refine the bounding boxes and a box classification layer to classify whether there is an object of any class in the proposed region or not. Also, for each region in the sliding window bounding boxes with predefined scales and sizes called anchor boxes, are given to the next layers and for each position of the sliding window and each anchor box the RPN outputs a probability that an object exists within that anchor box in the given position.

When the regions have been generated they are fed to the Fast R-CNN detector.

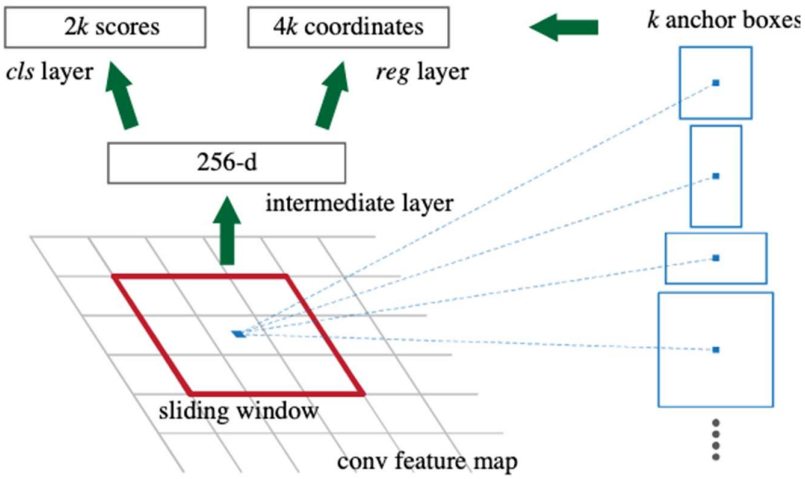


Figure 2.6 Region Proposal Network

2.4.4 Mask R-CNN

Mask R-CNN [7] extends Faster R-CNN. In the first stage, the Region Proposal Network is used, which is the network described in the Faster R-CNN section. The same architecture is used for detecting objects in the proposed regions (Fast R-CNN architecture), the output layers consist of a bounding box regressor and also a possible binary mask for each of the predefined classes.

For each RoI, binary masks for each class are generated and in this way classes won't compete for each pixel. Instead, each class generates a binary mask for the given region and the binary mask of the class that has the highest score in the classification layer is used.

However, since Mask R-CNN needs pixel-level specificity, the RoI pool used in Fast R-CNN needed some adjustment because some pixels were lost when pooling is used in the proposed regions because the size of the proposed region most of the times wasn't divisible by the size of the pooling filter. For this reason, RoI Align is used instead of RoI Pooling. In RoI Align, the grid is again splitted into blocks and in each block bilinear interpolation is used to compute the value of each block.

2.4.5 You Only Look Once v3 (YOLOv3)

YOLOv3 [8] comes with a different approach compared to RCNN when processing the input image. Instead of dividing the image into candidate regions, it looks at the entire image in order to model the context as a whole and predict the bounding boxes with more accuracy. The detection steps are:

1. Split the image into a grid. Each cell predicts bounding boxes and confidence that an object exists in that cell, P(Object). Also each cell predicts the probability of belonging to each predefined class (e.g P(Car), P(Person), P(Cat) etc.)
2. Conditional Probability for each class is calculated: P(Car | Object), P(Person | Object), P(Cat | Object) etc.
3. Bounding boxes and predictions are combined
4. Non-maximum Suppression (NMS) algorithm is used to remove merge bounding boxes of the same object and eliminate duplicate detections.

5. For each cell two bounding boxes are predicted with their confidence and their probability of belonging to each of the predefined classes. These predictions are made with a convolutional neural network.

During training we input the annotated bounding boxes and classes of the images and check the predictions of the cell in the center of the image. After that, we select the predicted bounding box that is closer to the annotated bounding box out of the two of that cell and increase the confidence and decrease the confidence of the other boxes. For the cells that don't have any ground truth values decrease the confidence of the cells and do not adjust the the class probabilities or the bounding boxes coordinates.

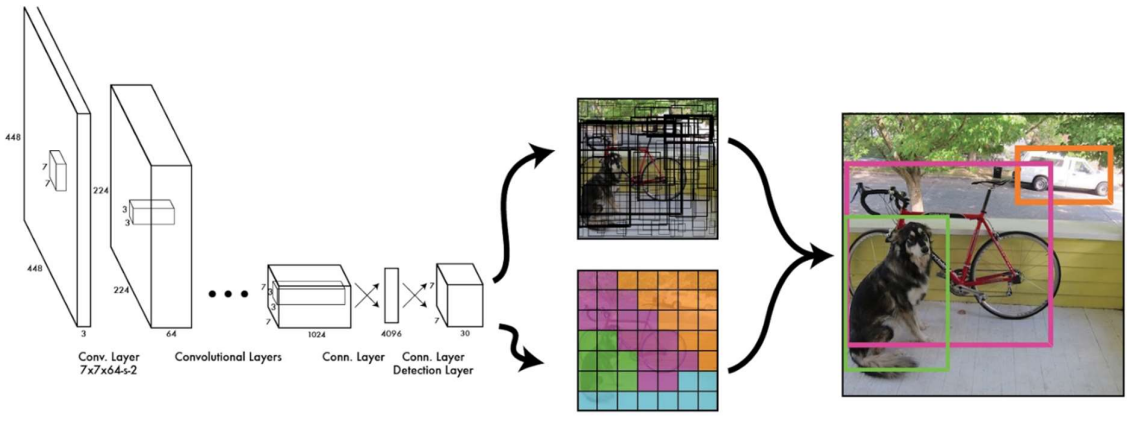


Figure 2.7 YOLOv3 architecture

2.4.6 Feature Pyramid Network (FPN)

The goal of an FPN [20] is to extract semantically stronger features by combining weak with strong semantics. As stated before, CNNs extract higher level semantics in each layer. CNNs with pyramid structure commonly used in computer vision can be exploited to extract semantically stronger features from an image. The pyramid consists of multiple steps and in each step there are multiple layers that output features of the same size. As we proceed in steps, the number of features outputted are halved. The FPN consists of two pathways: a bottom-up and and a top-down pathway. The output of the bottom-up is the input to the top-down. There are lateral connections that combine the two pathways. In order to halve the size of the features in each step during the bottom-up pathway, the stride of the convolution is doubled. At each step of the top-down pathway the features are doubled with nearest neighbor upsampling and then matrix addition is performed with the respective step in the bottom-up step. The result of this addition in each level is the output of the network.

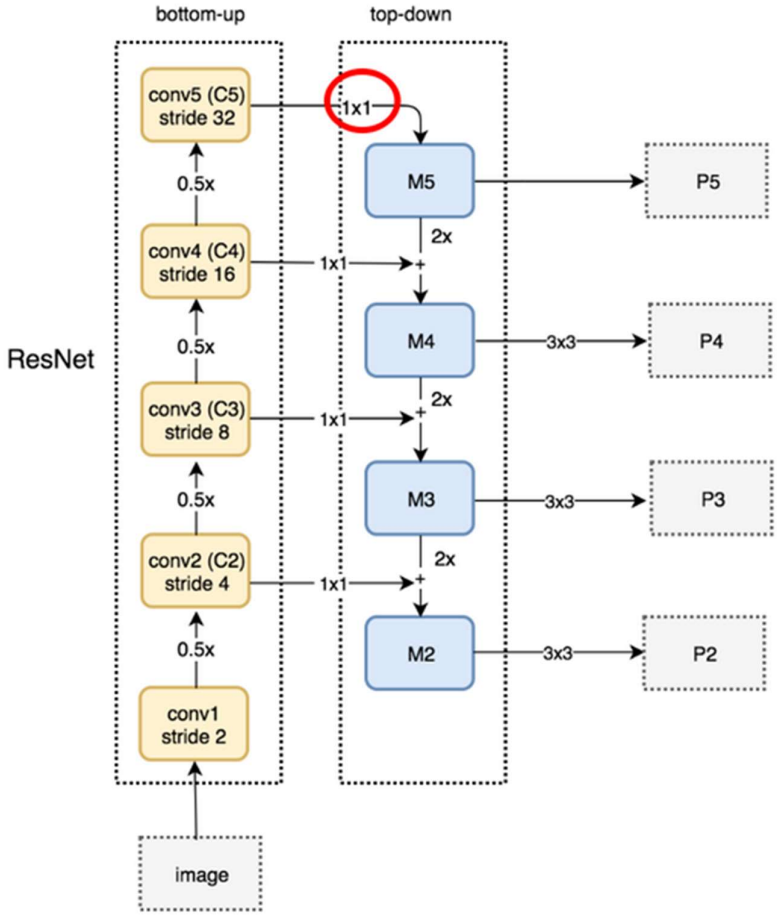


Figure 2.8 Feature Pyramid Network

2.5 Hash functions

2.5.1 Introduction

Hashing algorithms input a variable size input and map them to fixed size values. Cryptographic hash functions have the property of avalanche effect. With a small change in the input, the hash function produces a completely different value with the aim to avoid collisions between the same input. On the contrary, in multimedia we want the cryptographic hash functions in similar inputs to produce a similar output.

2.5.2 Difference Hashing

Difference hashing [21] is an image hashing algorithm that produces the same or a similar hash even if the following image parameters change: aspect ratio, brightness, contrast.

Given an image as input, the steps to create the image's hash are:

1. Convert the image to grayscale

   This step is done to allow us to perform a faster hashing since we are left with only one channel, the gray channel and also have more tolerance in images that have slightly altered colors.

2. Resize to 9x8 pixels

   This way, we ignore the aspect ratio of the images.

3. Check the difference in brightness between adjacent pixels

   Given as input an image I that has been processed with steps 1 and 2 we apply the following pseudocode:

```
let result = 0
for i=1 to 8
        for j=1 to 8
                let temp = 1 if I[x] > I[x+1] else 0
                result = result << 1 + temp
```

   9x8 array dimensions are chosen because the operations on the neighbouring pixels will be performed resulting in an 8x8 = 64 bit hash.

4. In order to compare two images, we can use the result obtained in step 3 for each image and compute their hamming distance. A good threshold for the hamming distance is 10 [21].

3. Technologies and Architecture

3.1 Introduction

We will now discuss the technologies used for designing and building the system. The basic components of the system are a web server for handling user authentication, the front-end for the documentation of the endpoints, the front-end and back-end for the authentication transactions analytics dashboard and a mobile application which is used as a second factor. Various tools have been used to design and implement these components.

3.2 Technologies selection criteria

All the technologies involved were chosen carefully which should satisfy some basic requirements.

These requirements are:

I. State-of-the-art technologies with a huge community supporting it
II. Maintainable technologies so that if a fault is discovered, it can be easily corrected without investing a lot of resources.
III. Simple framework with small learning curve
IV. Reliability
V. Portability
VI. Mechanisms that will provide security
VII. Back-end should be built on a programming language that is popular, which will also support statistical and machine learning packages for the biometric authentication
VIII. Cross-Platform development for the mobile application so that we can do our development and maintenance on a single framework which will be more efficient and cost effective

3.3 Technology stack

The technology stack is as follows:

I. **Django:** A framework used for web-development written in Python. Django follows the Model View Template pattern which will be explained in the next section. Because of the simplicity of this framework, developers do not need to dedicate a lot of time to master it and development can be fast and practical.
   Django is used for developing authentication and dashboard back-end
II. **HTML combined with Django Template Language:**
   HTML is a markup language and it is the one that is supported by browsers for frontend development. Usually it is accompanied with CSS which determines its element style.
   It is used for dashboard front-end and authentication widgets
III. **CSS:**
   It describes how each HTML element should appear. HTML elements have several attributes including their type, id, class and many. A selector is used to describe which HTML elements will be affected and what kind of design properties will have. Some properties involve the color, the size and the dimension of the element.
IV. **Flutter:**

Flutter is an open-source cross-platform development framework developed by Google. It is written in Dart, a language developed by Google as well. The main idea of Flutter is its reactive architecture. In a reactive architecture, there are two main components: the user interface (called Widget in Flutter) and the state variables attached with the user interface. When a variable related with the state of the user interface changes, the whole user interface rebuilds with the new values of the state variables. Flutter takes care of the efficiency of the UI building giving near-to-zero rendering time. Flutter is used to develop the mobile application used as a second factor both for iOS and Android.

V. Tensorflow, Keras, PyTorch:

Tensorflow [27], Keras (which is built on top of Tensorflow)[29] and PyTorch [26] are machine learning libraries available in Python which offer training and inference of deep neural networks. Also, Detectron2 [28] is a library developed by Facebook AI Research (FAIR) which offers various state-of-the-art object computer vision algorithms and is used in our system.

VI. RabbitMQ:

RabbitMQ is a messaging broker, which means a platform which allows different applications to communicate by sending and receiving messages. The messages are saved until they are received from an application. It is used for submitting asynchronously the voice and face recognition tasks into a queue which can be executed any time. When the task is submitted, an ID of the task is returned, and the browser can check whenever it wants the status of the task. This also allows tools for monitoring the task's status and checking for any failures.

VII. Celery:

Celery communicates via messages, usually using a broker to mediate between clients and workers. To initiate a task the client adds a message to the queue, the broker then delivers that message to a worker. A Celery system can consist of multiple workers and brokers, giving way to high availability and horizontal scaling. Celery requires a message broker to send and receive messages. In our case RabbitMQ is used as a message broker.

VIII. Flower:

Flower is a web based tool for monitoring and administering Celery clusters. It provides real-time monitoring using Celery Events. Some of its features areTask progress and history

Ability to show task details (arguments, start time, runtime, and more)

Graphs and statistics

IX. **PostgreSQL:**

PostgreSQL is a relational database which can be used in combination with Django. It uses Structured Query Language (SQL) to safely store and fetch data.

X. **Apache:**

Apache is the web server responsible for listening for web requests and forwarding them using its wsgi module to the Django web application to process them. Additionally Apache comes with the HTTPS module for secure communication between the server and the clients. It is easily configurable and allows for Http Strict Transport Security (HSTS)

XI. **Docker**

Docker wraps code into a unit of software called containers. Because of the smart efficient use of OS resources, the containers are lightweight, standalone and portable. The containers are platform independent and can be built quickly with a few commands.

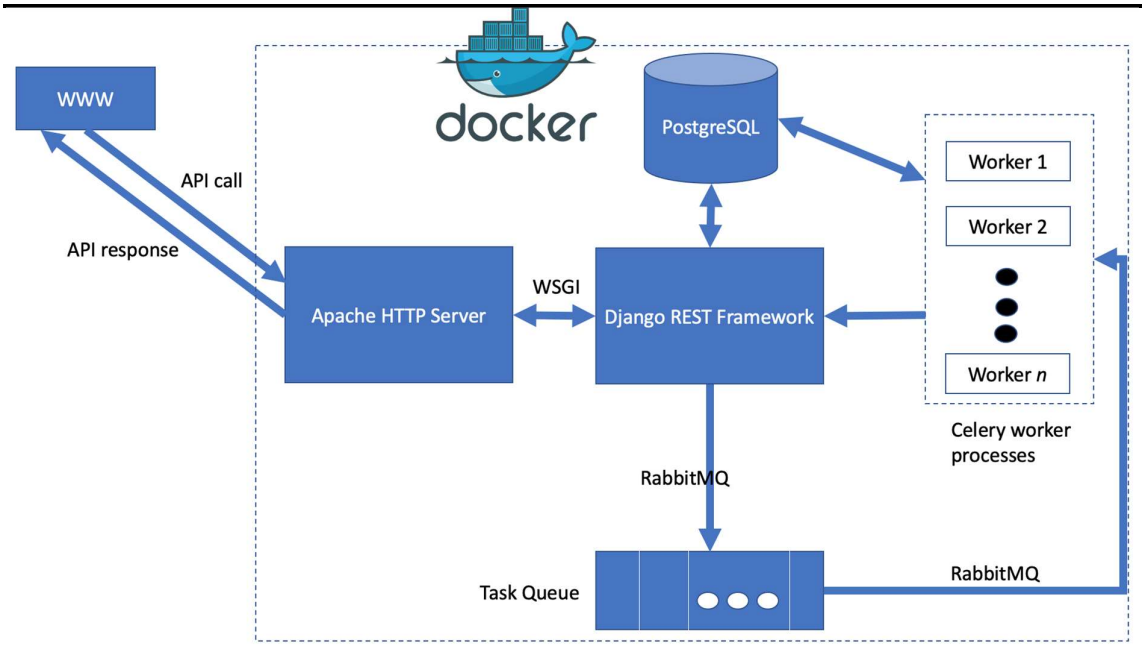


Figure 3.2 System flow during an HTTP request

3.4 Model View Template (MVT)

For developing an application, we use a specific framework which provides a specific way to build and run it. As noted above, django uses the Model View Template pattern. This pattern is split into three components: the model, the view and the template.

Model:

The model represents the structure of the data and the relationship between them. They can be thought of as the database schemas. In fact, django uses the models to create the database schema. The model data are formatted and received from the views.

View:

The views receive data as requests along with the request method from the client, processes the requests and formats it to correspond with the model structure so it can be saved in the database. It also fetches data from the database to be passed to the template for viewing.

Template:

Template is what is going to be presented to the client. They are used to generate the HTML code in an efficient and convenient way along with the static files which are the design of the website elements, interaction handling and images.

3.5 System architecture

In this section we will present how the components are interconnected within the authentication platform

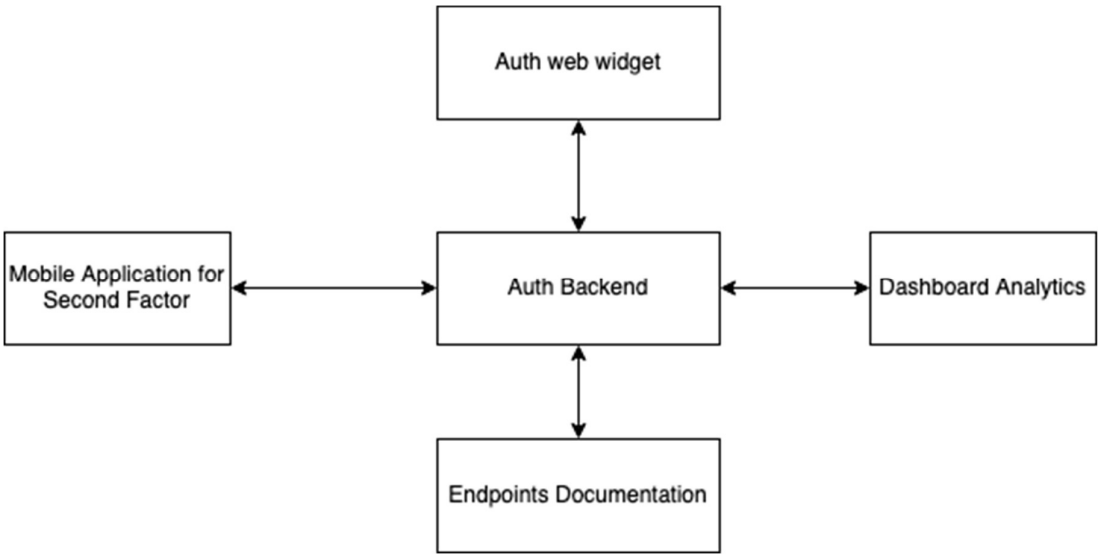


Figure 3.1 The interconnection between components

Every component communicates with the authentication backend. The reasons for the interaction of each component with authentication backend will be now explained:

Authentication web widget:

Communication with the authentication backend to initialize the widget, submit the one-time password, get a success/failure response and poll for the status of push-notifications and qr scans of the mobile application. The status for them is completed only when the mobile application sends a request to the authentication backend which contains whether or not the user has accepted the login request.

Dashboard Analytics:

Communication with the authentication backend to fetch the history of authentication transactions, the devices of each user, and any actions on each account such as block or device logout.

Endpoints Documentation:

Communication with the authentication backend to fetch metadata and serializers of each implemented endpoint that describe the accepted format and method that each request must have and the code and format of each response in every case.

Mobile Application:

Communication with the authentication backend to fetch each account's transactions, enroll an account to the device and accept/reject requests.

3.6 Database architecture:

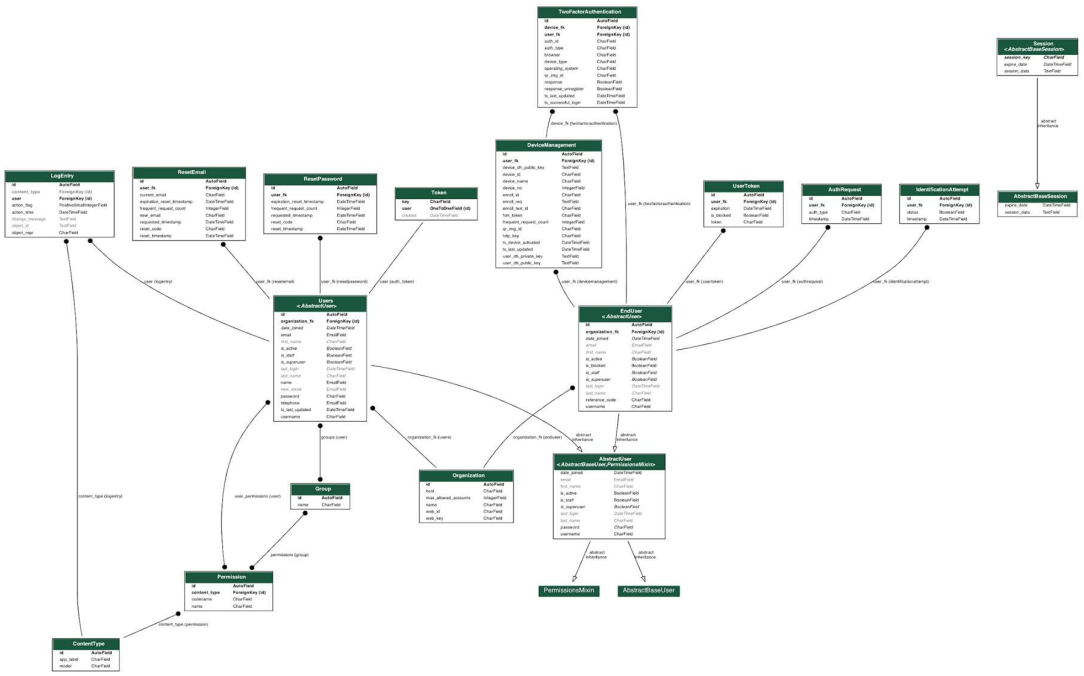


Figure 3.2 The entity relationship diagram

Now we will explain the purpose of each table along with their respective fields.

I. **TwoFactorAuthentication:**

Responsible for logging all the authentication requests with their details.

Fields:

user_fk: A reference to the EndUser table who has made the authentication request

device_fk: A reference to the mobile DeviceManagement who has made the authentication request

auth_type: The type of the authentication. Can be either be PUSH, QR or TOTP

response: The response to the authentication request. Can be either True or False

ts_successful_login: The timestamp when the user has responded to the authentication request. Null if he hasn't

ts_last_updated: The last timestamp in which this row has been updated.

operating_system: The operating system from which the user has issued the authentication request

browser: The browser from which the user has issued the authentication request

device_type: The type of the device from which the user has issued the authentication request. Can be mobile, tablet or pc

II. **IdentificationAttempt**

A table used to log all the identification attempts made during the biometric continuous authentication. At facial authentication, a photo of the user is taken by the webcam at a regular interval and the result of the face authentication is stored in this table.

Fields:

user_fk: A reference to the EndUser table who has made an identification attempt

timestamp: The timestamp of the identification attempt

status: The status of the identification attempt. True if the user was successfully found else False.

III. **EndUser**

A table which holds the user's data.

Fields:

username: The username of the user

password: The hashed password of the user. PBKDF2 algorithm with a SHA256 hash is used

reference_code: The reference code of the account which is used to enroll additional devices to the account. It is not needed when the first device enrolls.

organization_fk: A reference to the organization which the user belongs to.

is_blocked: Whether the user is blocked or not. Can be True or False.

IV. **AuthRequest**

A table used to count all the authentication requests made by a user. It is used to restrict spam requests.

Fields:

user_fk: A reference to the EndUser table who has made the authentication request

auth_type: The type of the authentication. Can be either be PUSH, QR or TOTP

timestamp: The timestamp in which the authentication request has been made

UserToken

A table which maps the authentication tokens with an account. Authentication tokens are used in the header of the request. If they are invalid or expired the request does not execute. This table allows for easily revoking tokens of a particular user or device. Currently Json Web Token (JWT) is used.

Fields:

user_fk: A reference to the EndUser table who owns the authentication token

token: The ID of the token

expiration: The expiration time of the token

is_blocked: Whether the token is blocked by an administrator. True or False

V. DeviceManagement

Describes the device paired with the account along with information needed for enrolling and logging in.

Fields:

device_id: The id of the device

device_name: The name of the device

device_no: The order of the device in which will appear in the list of the device in the web widget

user_fk: A reference to the user's model who enrolled with this device

enroll_id: The code which will be embedded inside the QR code during the enrollment.

enroll_text_id: The textual code which is used for enrolling the device with text.

qr_img_id: The image id of the qr code generated during the first user's login

totp_key: The secret key of the account's time-based one time password (TOTP) that is used to generate the six-digit one time password at any given time.

fcm_token: The token which is used to send a Firebase Push Notification (FPN) to the correct device.

ts_device_activated: The timestamp in which the device was activated. Null if the device is not activated.

ts_last_updated: The timestamp of the last update of this row.

3.7 Embedding authentication services

To embed our authentication methods in a web application iframes are used along with some javascript to handle success/failure.

3.7.1 Iframe

Iframe is an HTML element which is used to embed a website inside another website. A page with an iframe which renders the authentication is used after the first factor authentication.

3.7.2 Javascript

When the iframe initializes, callback functions listening for successful/failed authentication are registered. We have two different Window objects interacting, the iframe and the main website which embeds this iframe and we need a method to allow cross-origin communication. Window.postMessage() method allows for message passing between two window objects. In our case, this message passing is done during successful or failed authentication requests.

3.7.3 Callback URL for successful authentication

The server integrating the multi-factor authentication exposes a URL which will be called when the user has authenticated in order to create a session for the authenticated user and redirect him to the main page.

3.8 Security Measures

Because this system provides security to other systems this means that security is of high importance. For this reason we have taken several security measures which will be listed below:

3.8.1 Cross site scripting protection (XSS)

XSS protection is enabled in Django. This means that the Django templates using specific regular expressions, escape dangerous characters.

3.8.2 SQL injection

Django's querysets, which are the methods database handling are protected from this type of attack since queries are constructed using parameterization.

3.8.3 HTTPS

HTTPS allows encrypted communication. Without this, a man in the middle attack would be possible, and this could result in leakage of sensitive information. In order to use HTTPS in our services, we use Apache as a web server with mod_wsgi installed to deliver the incoming requests. Module wsgi (mod_wsgi) is responsible for serving incoming requests to Django.

3.8.4 HTTP Strict Transport Security (HSTS)

HSTS is a policy that enforces HTTPS communication in a website. A website that didn't contain HSTS would allow someone to view the website in two versions: the secure one which would not allow a man in the middle to view the information exchanged as plain text and an insecure version which would allow a communication where the messages could be easily viewed as plain text. This is expressed in the header of the response of the servers. The web browsers read this header and know that the only possible communication protocol is HTTPS.

3.8.5 JSON Web Token (JWT):

JWT is an open standard that provides a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed using a secret in our case (HMAC algorithm). In this way several endpoints can require authorization. Several endpoints check the header of the request for the existence and validity of the JWT. JWT is short lived and thus it is used in combination with a refresh token. This JWT is created and stored in our mobile application during the enrolling, along with a refresh token. Periodically the refresh token is posted to our authentication server to acquire a new JWT. When the refresh token expires, the account is considered expired and the user can't perform any action. The user has to re-enroll to obtain a new refresh token and a new JWT.

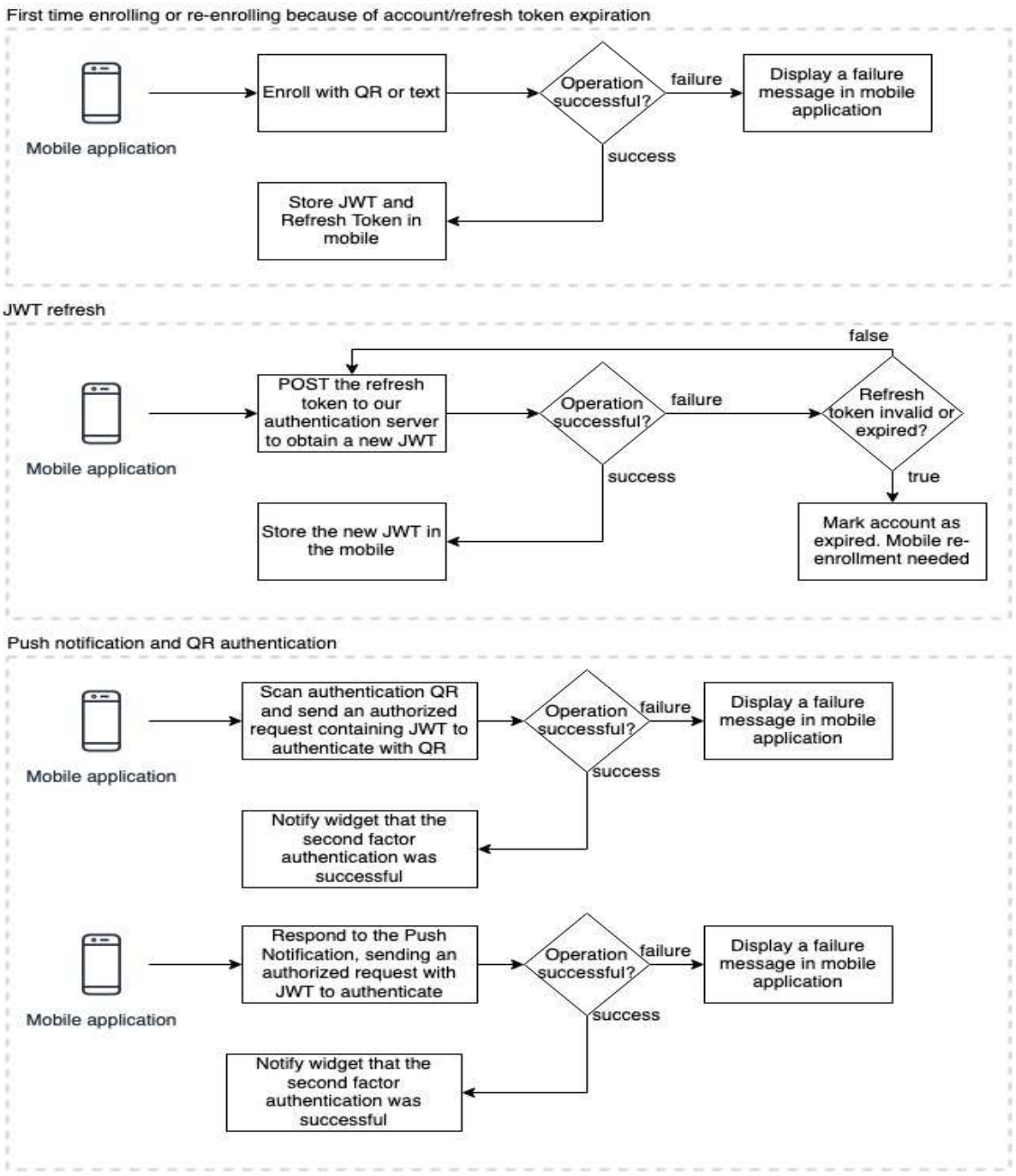


Figure 3.4 JWT creation, refresh and use on authentication

3.9 User Interface(UI)

First, the user will be guided to the system to create a demo graphical password (Figure 3.5). When he is done, he will be given a list of pictures and he will choose one which will be used as a background to create a password. The user will also have to enter a passphrase which will be an alternative way of authenticating instead of graphical user password if he wants to.

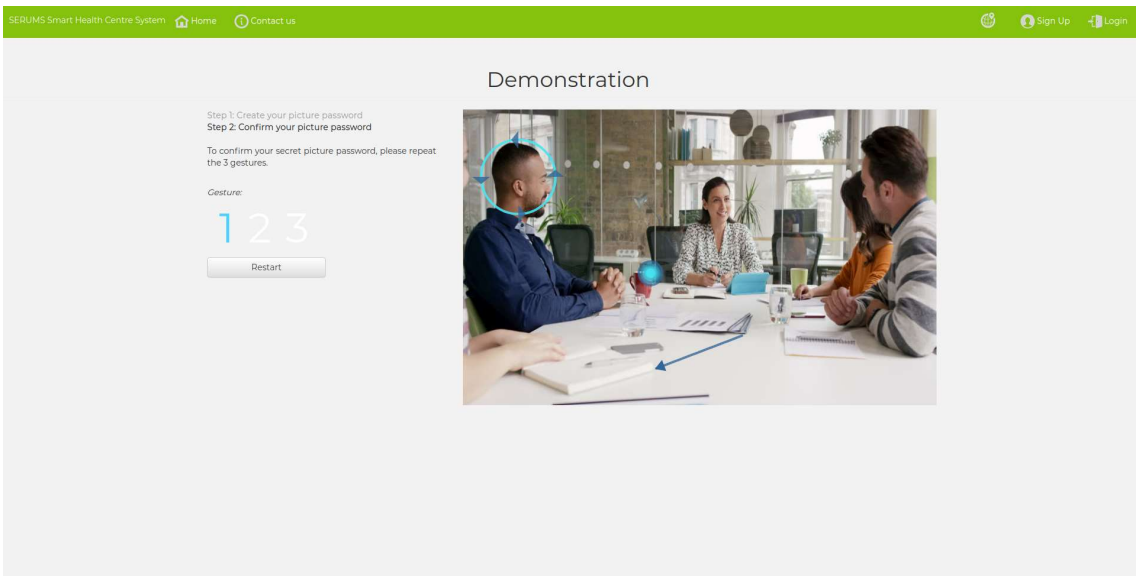


Figure 3.5 The embedded graphical user authentication

When the user has created a graphical password, the QR code enrollment screen will show up in the browser (Figure 3.6) and the user will download the application and scan the QR code to enroll to the system.

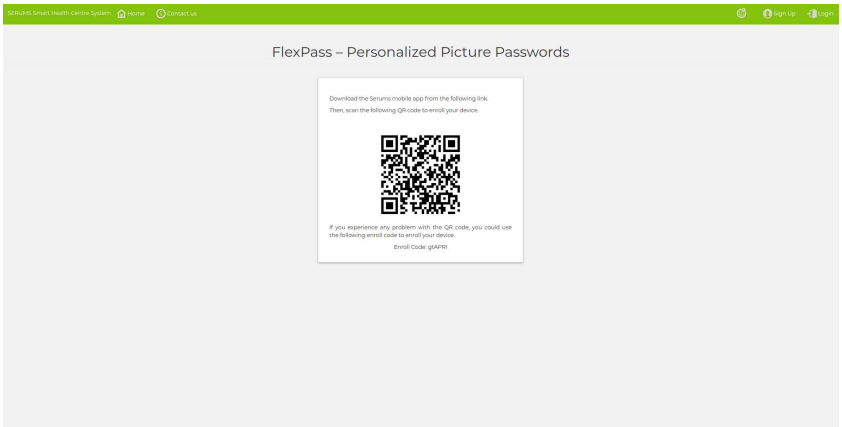


Figure 3.5 QR Code enrollment to the system

Figure 3.6 shows the authentication widget when the user has enrolled. A list of paired devices can be found. The user can select one of the listed devices to receive a push notification on a specific device to authenticate.

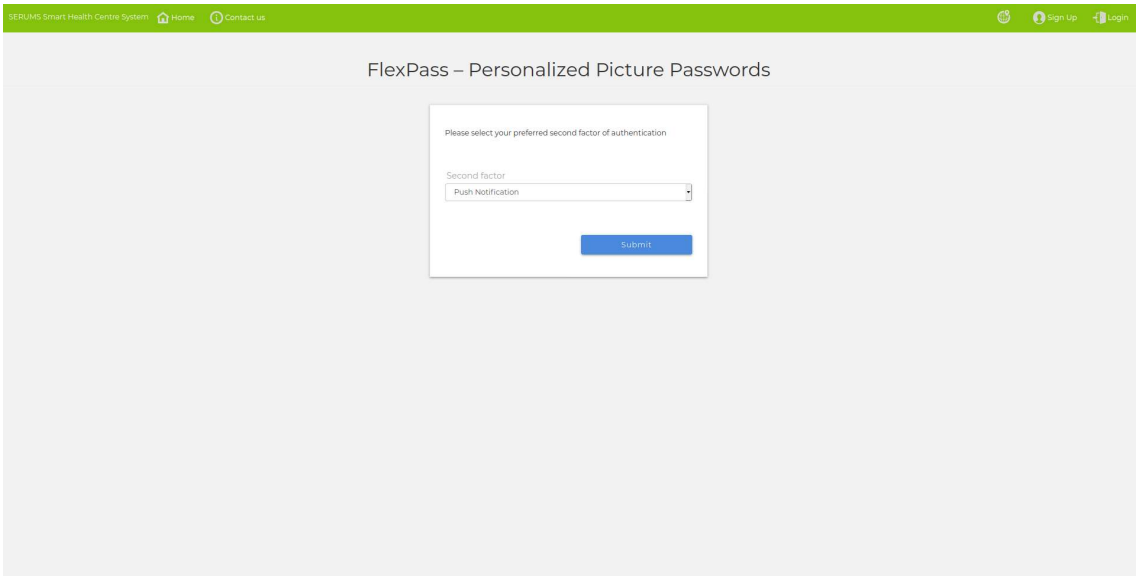


Figure 3.6 Authentication method choice

TOTP option will ask the user to enter a one-time password (Figure 3.7).

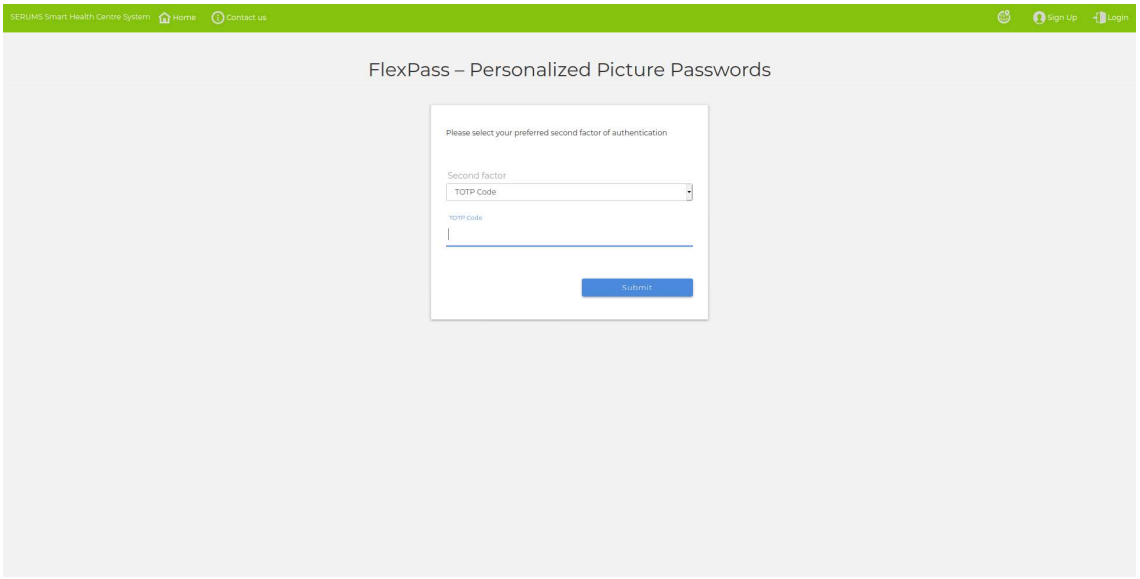


Figure 3.7 Web authentication widget for TOTP

Figure 3.8 shows a new user's screen in the mobile application.

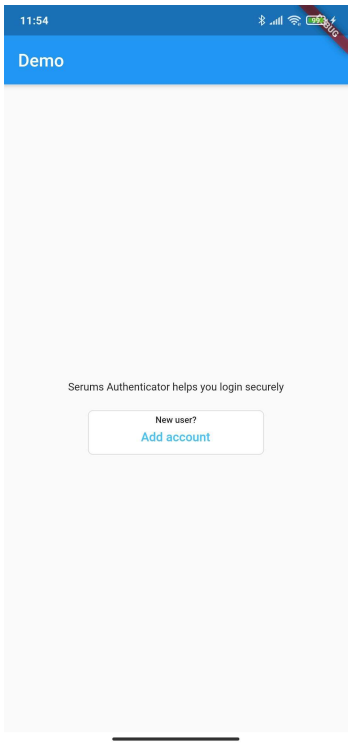


Figure 3.8 Mobile application new user screen

Figure 3.9.1 shows the authentication screen with one time password.

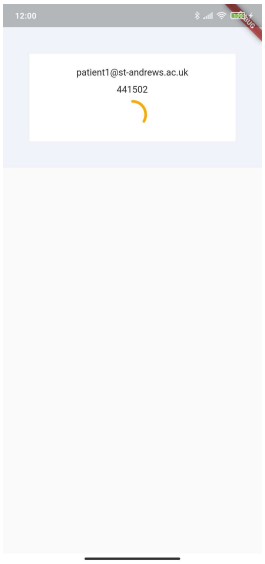


Figure 3.9.1 TOTP mobile application screen

Figure 3.9.2 shows the screen when the push notification is received and is selected by the user

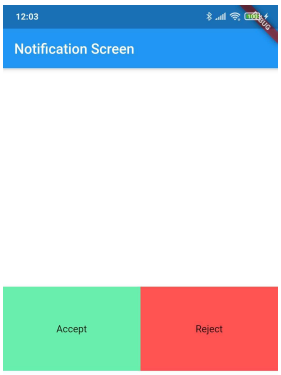


Figure 3.9.2 Push notification response screen

4. Multi-Factor Authentication

4.1 Introduction

In this chapter we will describe the requirement analysis and the design of the multi-factor authentication component.

4.2 Requirements analysis

4.2.1 Functional Requirements

Functional requirements describe how a system should work, the functions that the system supports and what is offered to the user.

The main functional requirements are:

- The user must be able to login to his system using two-factor authentication.There should be multiple methods of authentications such as QR code, Time-based One Time Passwords and Push Notifications.
- The user must be able to enroll any of his devices to the system as a second factor
- The user must be able to view the authentication history
- The user must be able to remove any account from his device
- The administrator must be able to login to the dashboard
- The administrator must be able to block a user
- The administrator must be able to delete a user
- The administrator must be able to logout all the devices from a user
- The administrator must be able to view the authentication history for all of his users.

4.2.2 Non-Functional Requirements

Non-Functional requirements describe restrictions and requirements on the implementation of a system.

The most basic non-functional requirements are:

- Performance: The platform should respond very fast, without causing the feeling of a delay. Throughput should be high, which translates to a high number of requests executed per second. There should be plenty of storage, so that data can be successfully saved in the database and the file system.
- Usability: measures how easily a user can use an interface. Nielsen[33] defines a usable interface, an interface which is easily learnable, users can accomplish their tasks efficiently, users can memorize how to proficiently accomplish their task, errors made by the users must be minimal and users must recover easily and the user interface must provide satisfaction.
- Reliability: The platform should be online most of the time with no or minimal downtimes. Mean Time Between Failure (MTBF) which is the time between two subsequent failures should be maximal and Mean Time To Recover (MTTR), which is the time for a system which had a failure to recover, should be minimal.
- Scalability: The system should be easily extendable to support more payload and new functionalities. This can be achieved with physical factors such as data storage, network bandwidth, CPU, GPU that will be able support future operations. At the same time software should be organized in such a way that it

will be easy to integrate new solutions. A modern approach for software scalability are microservices which structure a system as a collection of small services, operating independently and communicating with each other.

- Portability: how easily the software can be installed and operate into a different environment. This is a crucial nonfunctional requirement because this software is developed by a group of developers, where each developer may have a different environment and there is a need for compatibility. Also the mobile application should be portable, so that it can operate well in the different Android iOS versions.
- Security: the ability of the system to prevent unauthorized use of it's resources. A secure system increases user's trust which is of high importance for our system since it provides user authentication.

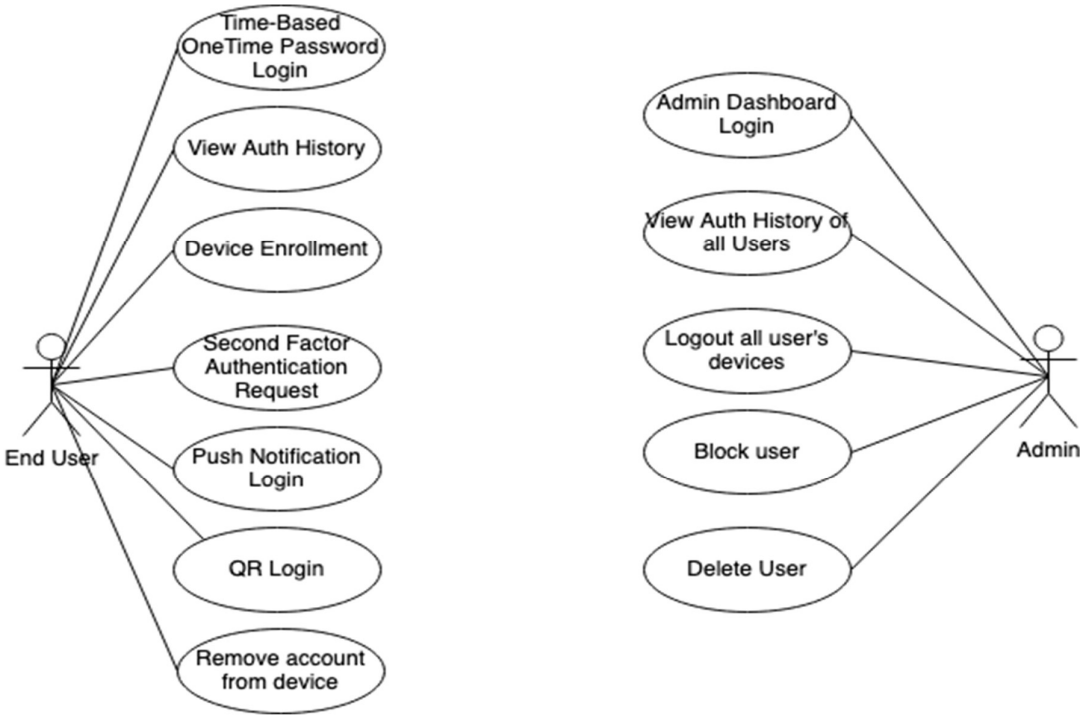


Figure 4.1 Use case diagram of Multi-Factor Authentication

4.3 System Design

4.3.1 Widget enrollment

The steps shown in the sequence diagram in the figure 4.2 will be now explained in detail.

1. Authenticate first step:
   End user will authenticate using the first factor (e.g textual password)
2. Initialize widget:
   The server which integrates the authentication widget will send a request to our authentication server to render the authentication widget.
3. Our authentication server will communicate with our authentication database. The database will check if there is a row with the user's username and organization. If nothing is found that means that the user from that particular organization/website has never enrolled before, else the user is enrolled and it will respond with the appropriate enrollment status to the .
4. If the user is enrolled our authentication server will render it's widget (which is an iframe embedded in the website where the end user is trying to authenticate) with a list of the authentication methods (QR, TOTP, Push Notification) for the user to select.
   4.1 Show auth methods:
   End user is now able to view the authentication methods and select any of them.
5. In the case the user has never enrolled the authentication server will render a QR code with an enrollment text in its iframe.
6. The user will scan the QR code from the widget using his mobile phone. The mobile phone will send a request containing the QR code to our authentication server.
7. The server will check in the database if the QR code is valid. If the QR code is valid
8. The widget will render a screen showing the authentication methods if the QR code was valid
9. If the QR code is invalid there will be an error message

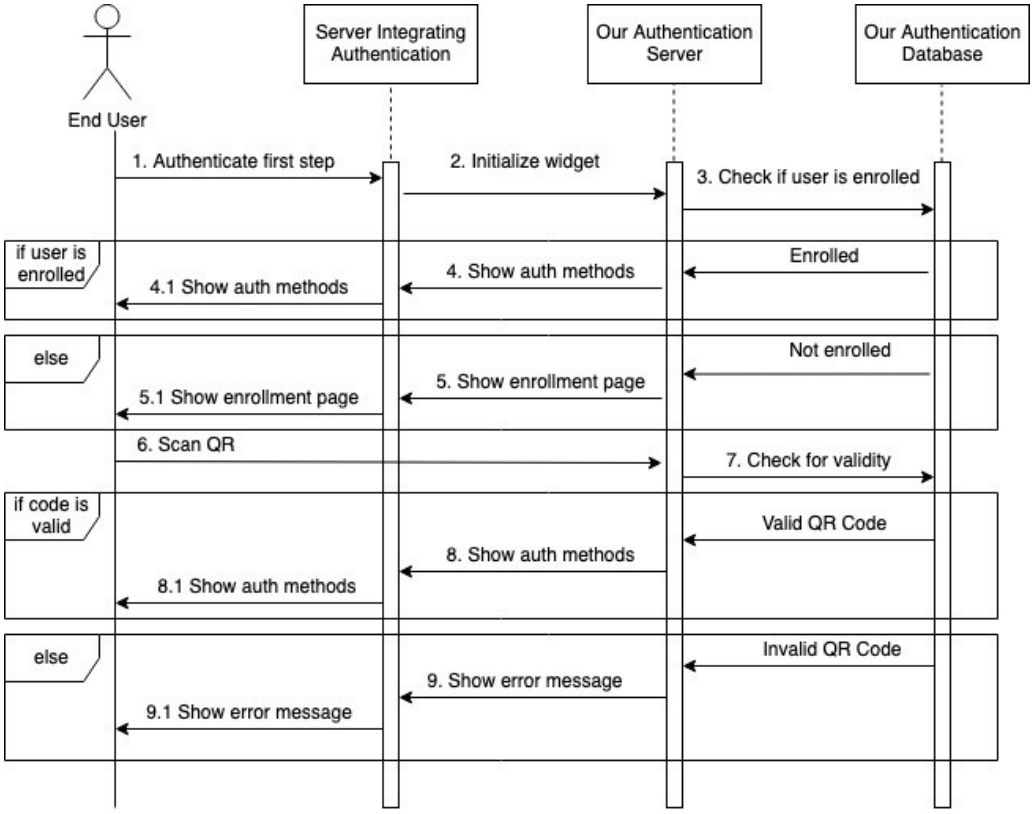


Figure 4.2 User enrollment

4.3.2 User authentication with TOTP

The steps shown in the sequence diagram in the figure 4.3 will be now explained in detail.

1. Authenticate first step:
   End user will authenticate using the first factor (e.g textual password)
2. Initialize widget:
   The server which integrates the authentication widget will send a request to our authentication server to render the authentication widget
3. Select TOTP:
   End user selects TOTP as the authentication method and enters the TOTP found in his mobile application
4. Check code:
   The TOTP is sent to the authentication server and the server fetches the secret key of the account from the database to generate the TOTP and check if it is correct.
5. Success auth:

If the TOTP entered is correct then the authentication server will respond that the authentication code is correct.

5.1 Handle success:

The server integrating the authentication methods will have to handle the successful authentication

6. Failed auth:

   A wrong TOTP was entered by the end user.

   6.1 Handle failure:

   The server integrating the authentication methods will have to handle the failed authentication

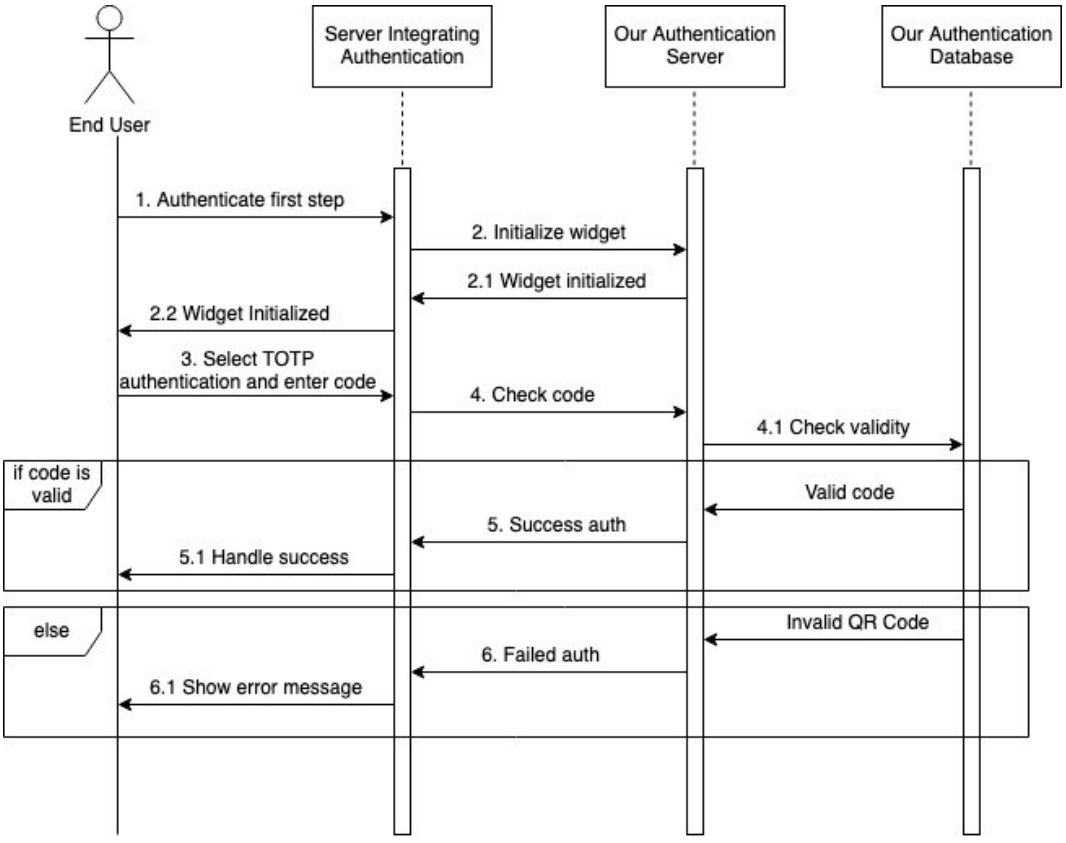


Figure 4.3 User authentication using Time-based One Time Password

4.3.3 User authentication with QR code

The steps shown in the sequence diagram in the figure 4.3 will be now explained in detail.

1. Authenticate first step:
   End user will authenticate using the first factor (e.g textual password)
2. Initialize widget:
   The server which integrates the authentication widget will send a request to our authentication server to render the authentication widget
3. Select QR authentication
   End user selects TOTP as the authentication method and enters the TOTP found in his mobile application
4. Initialize QR authentication
   A request is sent to the authentication server to initialize the QR authentication
   4.1 Generate QR code:
   A QR code is generated and stored in the database
5. Send QR code:
   The QR code is sent to the server integrating the authentication.
   5.1 Display QR code:
   The QR code is displayed at the website which integrates the QR authentication.
6. Scan QR code:
   The end user scans the QR code using the mobile application. The scanned code is sent from the mobile application to our authentication server.
7. Check validity:
   Our authentication server checks in the database to ensure that the QR code received is valid.
8. Success auth:
   If the QR code sent from the mobile application is correct then the authentication server will notify the website integrating the our user authentication for success.
9. Handle success:
   The server integrating authentication will handle the success.

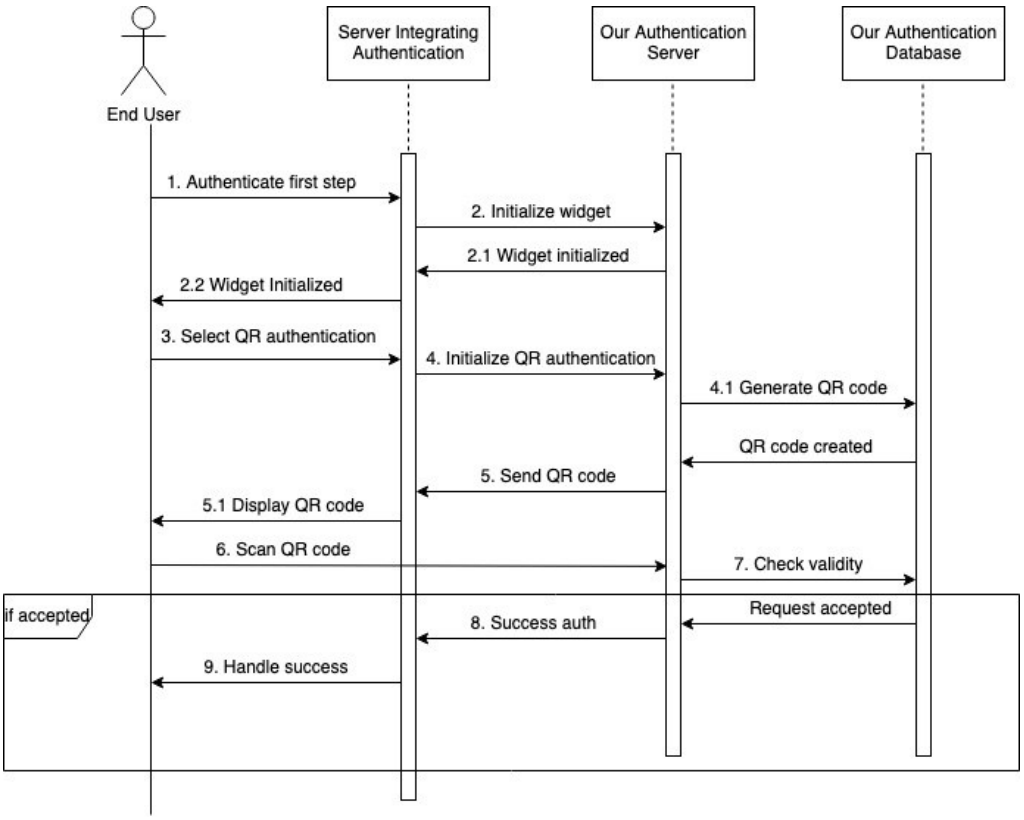


Figure 4.4 User authentication with QR scanning

5. Image Analysis Platform

5.1 Introduction

Thanks to the improvement of machine learning, computer vision has seen a lot of breakthroughs in the last decade [5, 6, 7, 8, 18]. A decade ago, object detection seemed like an intractable problem. Today object detection and label classification algorithms with over 95% accuracy show that nothing is impossible.

The purpose of this platform is to get a better insight of the pictures that are used for graphical user authentication. Using the user's eye tracking data during the password creation can be combined with the objects detected. The idea is that during the examination of the picture, the user fixates on specific areas. Objects detected can serve as potential hotspots and having a lot of eye gaze fixations on recognized objects may translate to a weak password. In this way, the system can predict that the user may potentially create a weak password before even the user starts the password creation and recommend the user to select another picture. Also, this

platform can serve as an assistive tool for creating a graphical user password strength meter. This can be achieved through the knowledge of the objects present and the shapes that the user has drawn.

In addition, this platform can fetch images from the internet, which are related to specific keywords and within a specific location, and can be more relevant to the user. Pictures in a close location or pictures containing something that he is familiar with, can have a dramatic impact on the user's memorability [22].

5.2 Main functionalities

The image analysis platform allows the user to select which algorithms he wants to run on the images. Also, a google image search is embedded along with a location based image search. In location based image search, the user enters a location and a radius within the location. Both keyword search and location search are offered by Google using a RESTful API. The user can also upload his own images for analysis. All the data can be downloaded in a JSON format at any time.

5.3 Object detection

The current object detection algorithms are: Mask R-CNN trained on COCO [23], PASCAL VOC [24] and Cityscape [25] datasets, YOLOv3 and TinyYOLOv3 (which is similar to YOLO with less convolutional layers) trained on COCO. The way the platform is built allows for easy adjustment of existing algorithms and addition of new ones. The algorithms used are implemented in Tensorflow [27] and PyTorch [26]. The models pre-trained weights are stored on the server, and in each request they are loaded so that the model can predict the objects present in the image along with their respective locations and bounding boxes.

5.4 Multi-label classification

We decided to use third party services for label detection. We use Google's Vision AI, Amazon's Rekognition and Clarifai. These services expose RESTful API services that give results in minimal time.

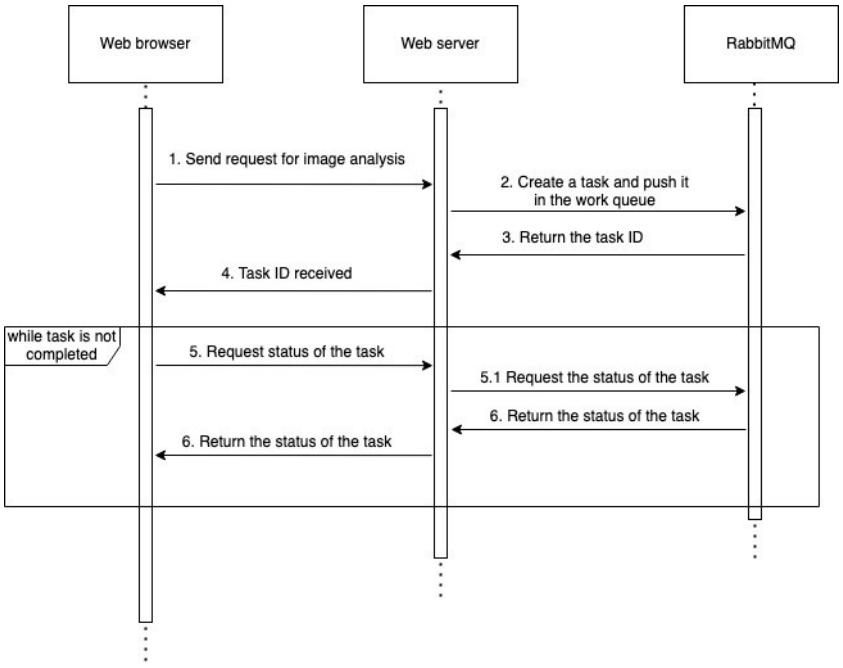


Figure 5.1 Image analysis flow between browser, server and RabbitMQ

5.5 Database Architecture

Figure 5.2 depicts the database schema.

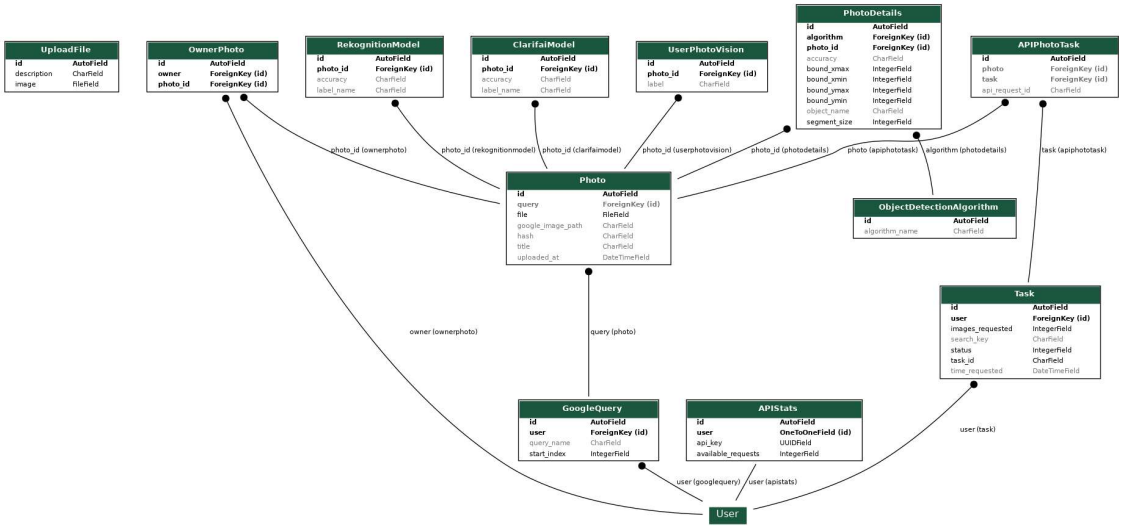


Figure 5.2 Database schema

Tables description:

1. **Photo**

   Contains details for each unique photo uploaded

   file: The path to the photo

   uploaded_at: The timestamp of the upload date

   query: A reference to the GoogleSearch table which contains the search keyword that was used to fetch this image. If this image was uploaded by the user then it will be null

   google_image_path: The image path if the image was uploaded with google search, else null

   hash: The hash of the image produced with perceptual hashing

2. **GoogleQuery**

   Contains information for each google search query

   user: A reference to User table

   query_name: The keyword used to find the user

   start_index: The last index used for the image. If the user searches twice for the same keyword, google search won't fetch the same images.

3. **OwnerPhoto**

   Describes which users can view which images. If users upload or fetch from the internet the same images, then perceptual hashing will detect it and the same image won't be analyzed twice.

   owner: A reference to the User table

   photo_id: A reference to the Photo table

4. **ObjectDetectionAlgorithm**

Describes information about each object detection algorithm

algorithm_name: The name of the object detection algorithm

5. **PhotoDetails**

   Contains analysis results for each photo

   photo_id: A reference to Photo table

   algorithm: A reference to the algorithm

   object_name: The name of the object detected

   bound_ymin: The Y coordinate of the bottom left box corner of the image detected

   bound_ymax: The Y coordinate of the top left right box corner of the image detected

   bound_xmin: The X coordinate of the bottom left box corner of the image detected

   bound_ymax: The X coordinate of the top right box corner of the image detected

6. **UserPhotoVision**

   Contains analysis results for Google Vision label classification algorithm

   photo_id: A reference to the Photo table

   label: A semantic label that the Google Vision label classification has returned

7. **RekognitionModel**

   Contains analysis results for the Rekognition label classification algorithm

   photo_id: A reference to the Photo table

   label: A semantic label that the Rekognition has returned

8. **ClarifaiModel**

   Contains analysis results for Clarifai label classification algorithm

   photo_id: A reference to the Photo table

   label: A semantic label that the Clarifai label classification has returned

9. **GraphicalPassword**

   Contains the gestures of each graphical password that was uploaded to the platform for analysis. Each row represents a gesture of the password.

   password_id: The id of the password that the gesture belongs to. Used so that we can group the gestures belonging to the same password

   algorithm: A reference to the ObjectDetectionAlgorithm table which represents the algorithm used to analyze the image for the creation of the graphical password.

   order: The order of the gesture in the password

   gestures_photo_location: The location of the processed image that contains the gestures

   gesture_type: The type of the gesture. ‘L’ for line, ‘C’ for circle and ‘T’ for tap.

   x1: The x coordinate of the gesture

   y1: The y coordinate of the gesture

   x2: If gesture is tap or circle: Null. If the gesture is a line: the x coordinate of the end of the line.

y2: If gesture is tap or circle: Null. If the gesture is a line: the y coordinate of the end of the line.

radius: If gesture is tap or line: Null. If the gesture is a circle: the radius of the circle.

is_clockwise: If the gesture is tap or line: Null. If the gesture is a circle: True if the circle has clockwise direction else False.

5.6 User Interface

Figure 5.6.1 shows the main page interface.

Analysis settings opens a modal with the enabled object detection algorithms, and the user can enable or disable anything he wants.

Clear Database button clears the database. Used for debugging purposes

Generate Json button generates and downloads a JSON file with all analysis results

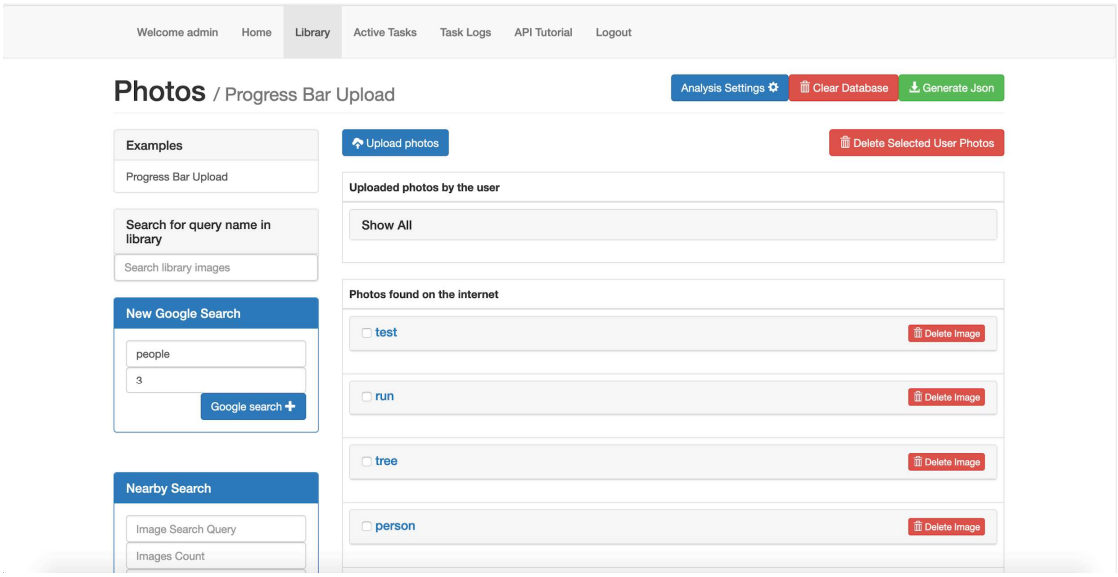


Figure 5.6.1 Image analytics component

Figure 5.6.2 shows the active tasks with their state.

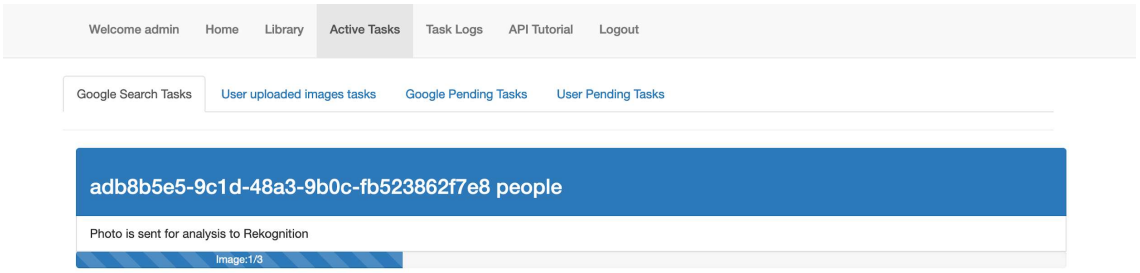


Figure 5.6.2 Active image analysis tasks with their state

Figure 5.6.3 shows the analysis results for a picture. The object name, the prediction confidence and the bounding boxes coordinates

person 0.8636796474456787% ymin:30 xmin:26 ymax:32 xmax:28

pottedplant 0.8582054376602173% ymin:28 xmin:27 ymax:49 xmax:42

pottedplant 0.8411549925804138% ymin:40 xmin:1 ymax:56 xmax:17

car 0.7520239353179932% ymin:30 xmin:22 ymax:32 xmax:28

pottedplant 0.7162237763404846% ymin:36 xmin:55 ymax:40 xmax:59

chair 0.7153794169425964% ymin:33 xmin:71 ymax:56 xmax:96

pottedplant 0.5215601921081543% ymin:24 xmin:78 ymax:41 xmax:93

car 0.5204785466194153% ymin:31 xmin:22 ymax:32 xmax:27

Figure 5.6.3 Objects detected in the image

Figure 5.6.4 Shows the number of guesses needed for the informed bruteforce attack to crack each gesture on the analyzed image.

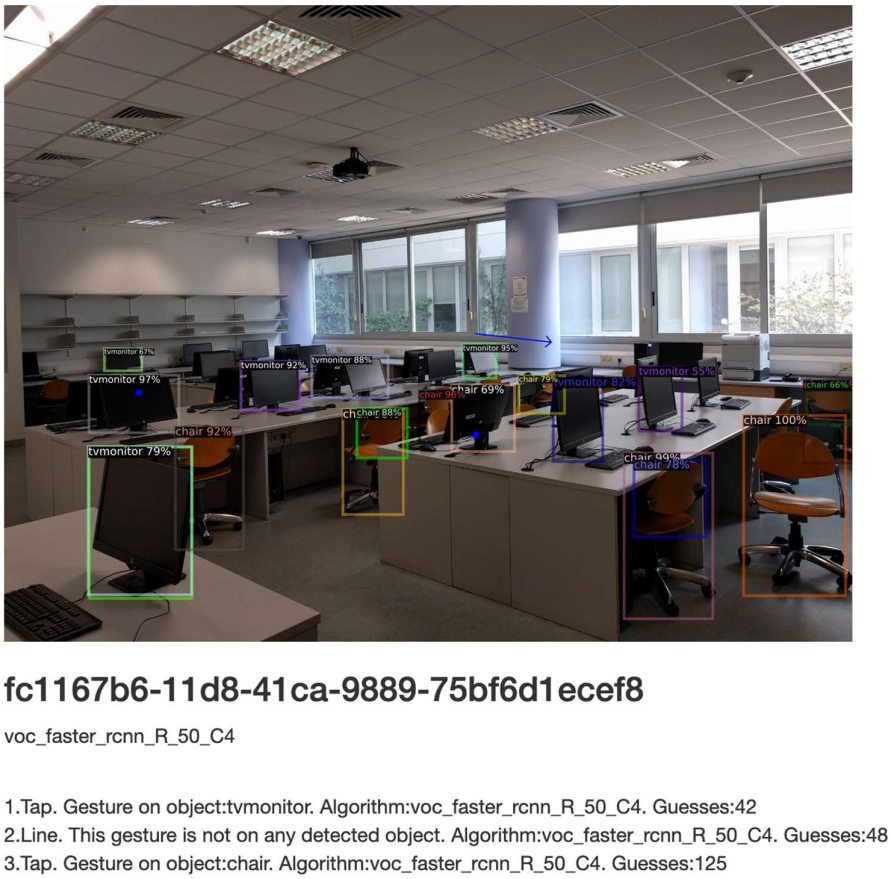


Figure 5.6.4 Graphical password with objects detected brute-force analysis

Figure 5.5.5 Shows the result of similar search. Labels of the image are used to fetch similar images from google using the search API.

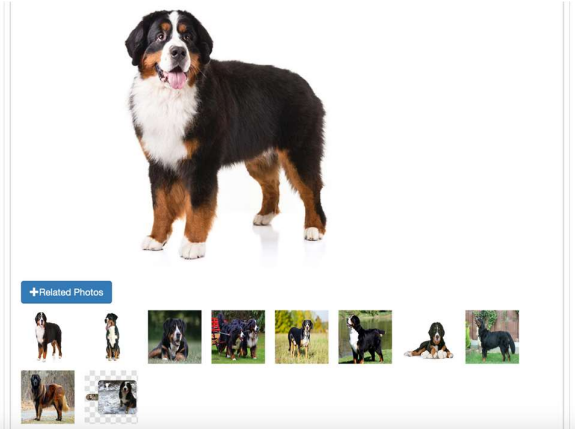


Figure 5.5.5 Similar Image Search

Figure 5.5.6 Shows the task logs. It consists of a sorted list with the task id, search keyword, number of images requested and the date.

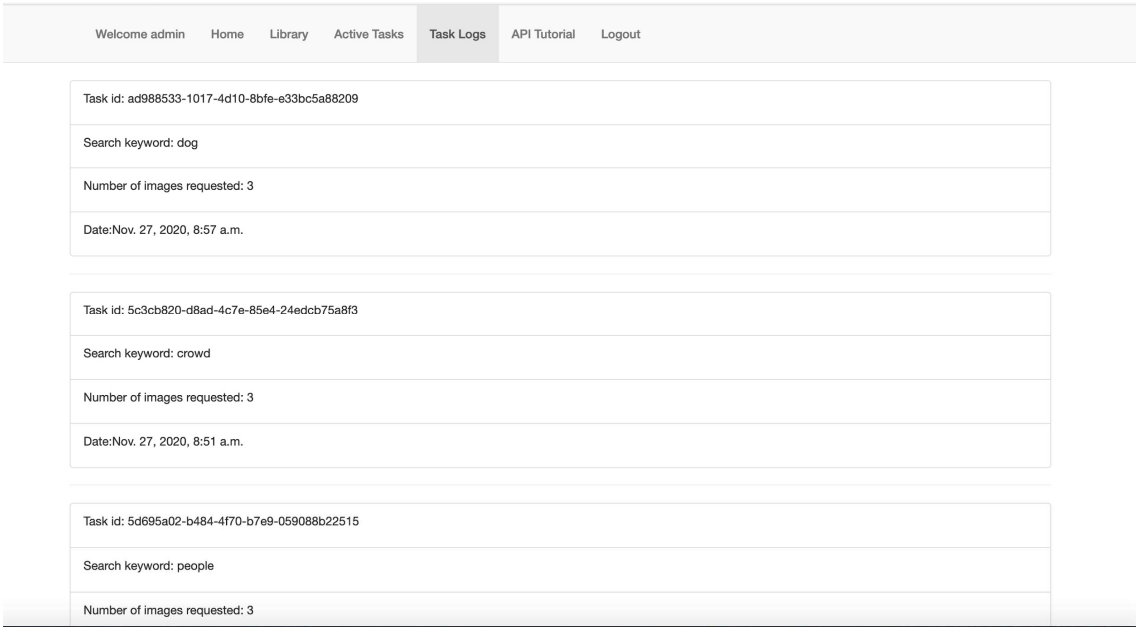


Figure 5.5.6 Task logs

Figure 5.5.7 Shows the live monitoring graphs of Flower. The graphs of succeeded tasks, failed tasks, tasks pending in the queue for execution and the execution time needed for each task are shown.

Figure 5.5.7

Figure 5.5.8 Shows the logs of the tasks that have run. The name, state, arguments, date received and date started are shown.

Figure 5.5.8 Task logs

Chapter 6

6. Evaluation

6.1 Graphical password platform evaluation

6.1.1 Key evaluation questions

Q1. How does the mechanism that proposes regions using object detection algorithms for informed graphical brute-force attack compares to an uninformed graphical brute-force attack?
Q2. How often do users select regions where objects are detected by the object detection algorithm for their graphical password?

6.1.2 Evaluation setup

In order to test how well the graphical user password platform performs, we developed a Web-based PGA-like graphical authentication scheme (Figure 6.1) in which users can draw three gestures that can be of type tap, circle and line. They can draw a gesture of a specific shape multiple times or not at all. Free line gestures are not permitted and are converted in one of the permitted gesture types. The location, the order and the direction of the gestures matter.

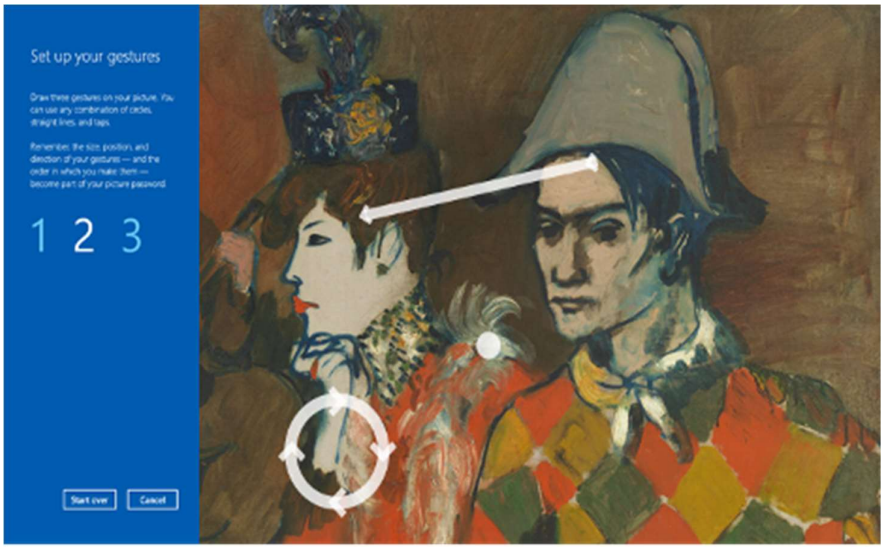


Figure 6.1 Picture Graphical Authentication

The image is split into segments and gesture segments are recorded instead of pixels. To calculate the segment size, the longest side of the image is divided by 100. Then the shortest side is divided by the segment size. In this way a grid of 100 squares is created and the segments of the gestures are recorded instead of the pixels. A margin of error of 36 segments around each gesture are also permitted.

Before registering a password, the user is presented with an example image and is required to create a demo password in order to get familiar with the system.

To some users a set of images of sceneries from their everyday life was given to them and they were requested to choose one to use as background for their graphical password. The rest of the users were given generic images which were not related to their everyday life.

They were also told that they would need to memorize the password in order to login another day to complete a questionnaire. In this way users are engaged to create a password that they will easily memorize. Furthermore the images used as background had similar number of hotspots and complexity. In order to calculate the number of hotspots a combination of saliency maps and saliency filters were used [32]. These hotspots draw the individual's attention making them candidate regions for the user's gestures. Also, the complexity is calculated using entropy estimators.

A total of 36 participants, (20 females) participated ranging in age between 20 to 32 years old (m=21.58, sd=2.25). To avoid biases, participants had no relationship with the researchers and no prior experiences with PGA-like authentication mechanisms. Participants were undergraduate students within the university and consisted both of people with and without technical background.

6.1.2 Analysis of results

After the case study we used various object detection algorithms on the images which were trained on different datasets in order to extract potential hotspots of the image. Depending on the image, the algorithm and the dataset used during training, the hotspots extracted varied.

The algorithms were trained on two widely used computer vision datasets: Common Objects in Context (COCO) and PASCAL Visual Object Classes (VOC).

We then employed a brute-force algorithm that starts searching for the password gestures location in the regions where objects were found by the object detection algorithm. Because of the huge amount of combinations that there can exist in a graphical password that follows our guidelines the brute-force is simplified. It only tries to find the location of the three gestures. In case of a circle, the center of the circle is considered, in case of a line, the location of the beginning of the line and in case of a tap, its location is considered. If no password combination is found in any of the suggested regions, the brute-force starts from the upper left corner of the image and tries to find the gestures location up to the bottom right corner.

In figure 6.1 we can see boxplots for the number of guesses needed for each algorithm. From the boxplot we can deduce that in most of the cases in informed brute-force less attempts are needed to find the location of the gesture. This answers the key evaluation question Q1.

As we can see, for all the algorithms the first quartile is less than 30. These are cases where the gestures are on objects detected by the algorithm. The third quartile varies with each algorithm and these are cases where the gesture is not on any of the bounding boxes suggested by the object detection algorithms. The number of objects an algorithm detects plays a considerable role in this number. While there is a bigger chance to find the gesture when more and bigger bounding boxes are suggested, if the gesture is not present in one of those regions, the brute-force algorithm will have to do more guesses to find the location of the gesture. In our case, images contained a lot more items that belonged to the PASCAL VOC dataset (i.e., chair, monitor) that was used to train MASK RCNN (grey boxplot) and this is why its third quartile is bigger than the rest. The yellow boxplot which represents MASK RCNN trained with cityscapes dataset, did not find any objects in the images and thus the brute-force started immediately from top left to bottom right. In fact, the first quartile is because of some gestures located in the top left of the image.

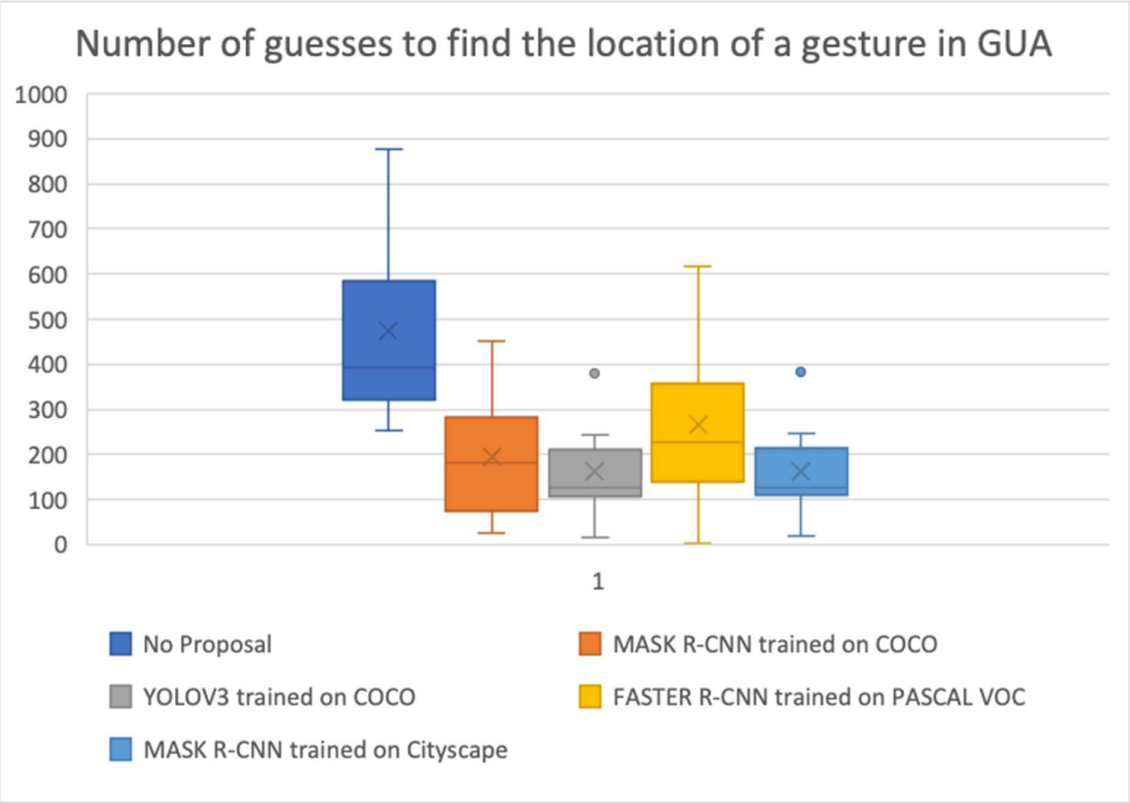


Figure 6.2 Number of guesses to find a gesture's location

Looking in Figure 6.3, as the number of objects detected increases, the average number of guesses increases because the brute-force will search in more places in the image.

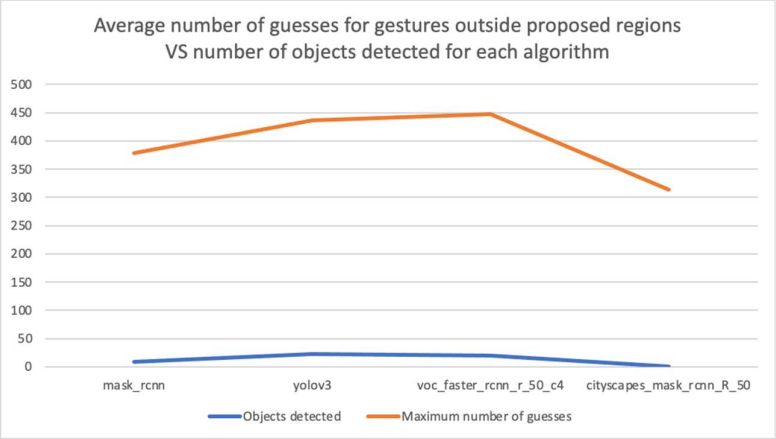


Figure 6.3 Average number of guesses for gesture outside proposed regions VS object count detected for each algorithm

In figure 6.4, we can see that the number of guesses for gestures on hotspot are way less than the number of guesses for gestures not on any hotspot suggested by the object detection algorithm.

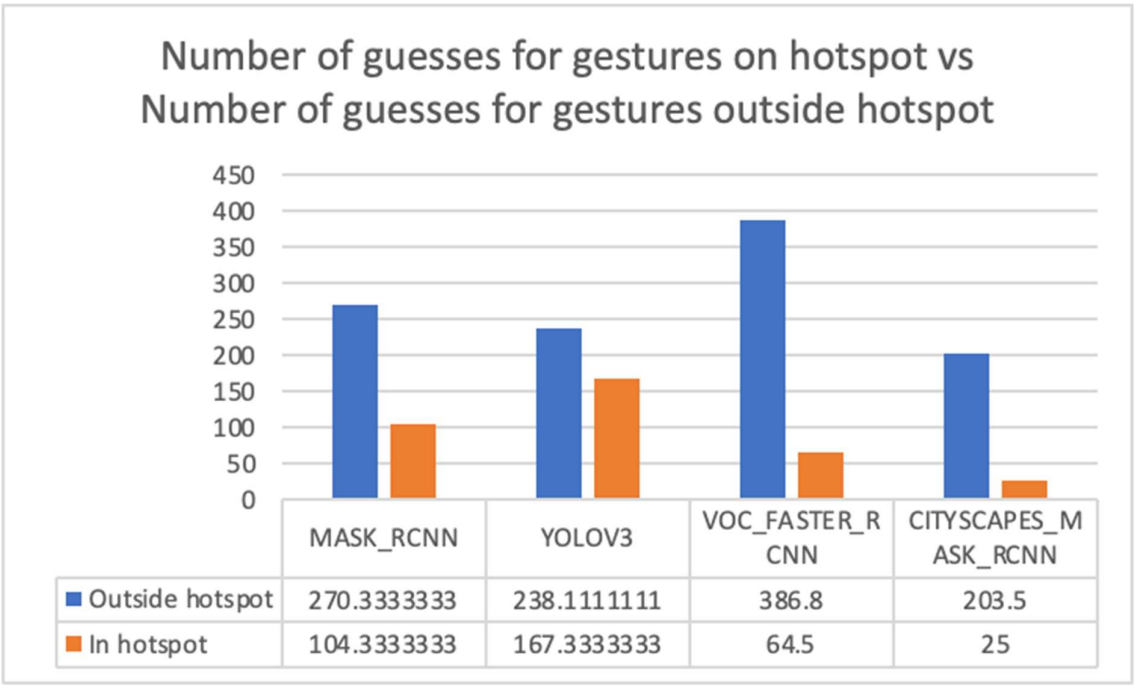


Figure 6.4 Number of guesses for gestures on hotspot VS Number of guesses for gestures outside hotspot

In Figure 6.5 the percentage of objects detected that were also used by the user to create a graphical password is shown. Different algorithms on different pictures are presented (pictures 1, 2, 3). This percentage is dependent on the content of the image, the data set used to train, the location where the user selected the gesture and the object detection algorithm. This answers the key evaluation question Q2.

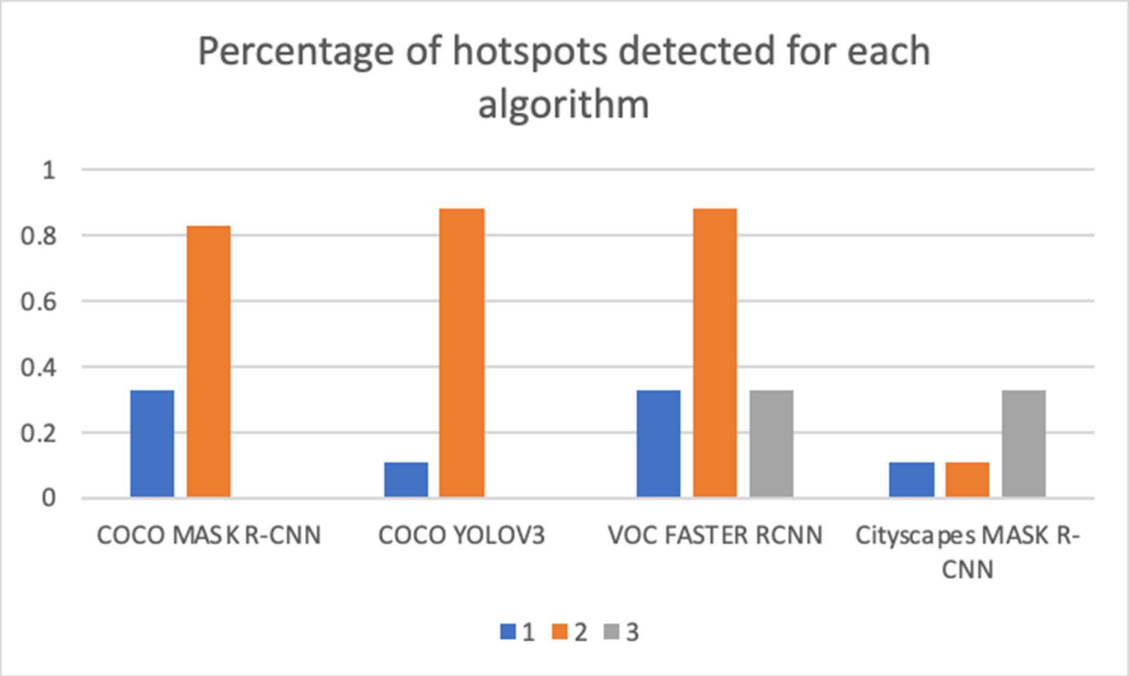


Figure 6.5 Percentage of hotspots detected for each algorithm

6.2 Multi-Factor Authentication evaluation

6.2.1 Key evaluation questions

Q1. What percentage of participants accomplished to enroll in the system?

Q2. What percentage of participants were unauthorized to enter to the system but were incorrectly accepted?

Q3. What percentage of participants were authorized to enter the system but were incorrectly rejected?

6.2.2 Evaluation setup

For the purposes of SERUMS H2020, users were invited to interact with our developed MFA combined with a Graphical User Authentication Mechanism. Users came from 3 institutions: Zuyderland Medical Center (ZMC), Fundacio Privada Clinic Per a la Recerca Biomedica (FCRB) and University of St Andrews (USTAN). The user composition was as follows: 21 users from ZMC, 21 users from FCRB and 32 users from USTAN.

The integration capabilities of the push notification method were assessed based on hardware compatibility, software compatibility, systems interoperability, and vendor independency. No issues were reported during the period of the conducted studies.

6.2.3 Findings

To answer the key evaluation question Q1 we calculated Failure To Enroll (FTE). It is defined as the number of failed enrollment attempts to the system over the total enrollment attempts. Failed enrollment attempts were 0 and total enrollment attempts were 18, so FTE was 0/18 which translates to 0% failed enrollment attempts.

To answer the key evaluation question Q2 we calculated False Acceptance Rate (FAR). It is defined as the percentage of identification instances in which unauthorized persons are incorrectly accepted over the total number of enrollment attempts. No unauthorized people were incorrectly accepted, thus the FAR metric was 0%.

To answer the key evaluation question Q3 we calculated False Reject Rate (FRR). It is defined as the percentage of incorrectly rejected access attempts by an authorized user over the total number of enrollment attempts. No access was given to the system incorrectly, so FRR was 0%.

7. Conclusions and Future Work

7.1 Conclusions

The purpose of this thesis project was to design and develop a multi-factor authentication mechanism and evaluate how well it can be combined with graphical user authentication by evaluating the system's usability and security. Our second goal was to better understand how background images affect users graphical password choice by developing tools for image analysis. This will help in the development of a graphical password strength meter that will give feedback to the user about his password strength and engage him to create a stronger password.

In order to evaluate the system we invited people to interact with the whole system, we've collected user's interaction data and analyzed them.

7.2 Limitations

The first limitation concerns the setting where the evaluation ran. The study was conducted in a lab environment. While there was no external noise that would draw the user's attention, this wasn't a realistic setting. We speculate that the user's graphical password choices may vary in a live environment. Also the time needed to authenticate through this system is expected to increase more.

The second limitation has to do with the object detection algorithms. Even though there has been a big progress in the computer vision algorithms they still have some limitations. To begin with, computer vision algorithms are trained on datasets that contain a labeled set of classes. Thus, the algorithm is able to detect only objects that it is trained on. This is a problem, because many objects are not detected by some object detection algorithms but are detected by others. Consequently, many users when creating a graphical password tend to choose regions where objects exist but the object detectors are unable to spot them. This leads to a huge number of guesses in order to crack the password, while the region selected by the user may be in a predictable region. Also, sometimes there are regions where the algorithms falsely spot objects.

7.3 Future Work

As stated above, our main goal is to create a graphical password strength meter that will be able to calculate the password strength by measuring the number of guesses needed to crack a password.

The following are suggested to make a more realistic strength meter:

- Consider a more realistic graphical brute-force attack by considering all the parameters of the gestures such as the radius of the circle and its direction and the end of the line gestures. Also, instead of trying to crack each password individually, try to crack the combination of them. However, these extra parameters will be time costly. In order to consider these extra parameters we will need to develop approximation algorithms to estimate the number of guesses using combinatorics.
- Find a policy to limit and prioritize the suggested regions. The bruteforce attack should be wise enough to know which detected objects draw the user's attention.
- Investigate how human factors affect the selection of regions. Human factors can be used as input to the informed brute-force attack.
- Train custom object detection models on specific classes of items where people are more likely to select them. An alternative would be to train a regressor which inputs regions of an image and outputs the probability that someone will select that region.

References

[1] Bonneau, J., Herley, C., van Oorschot, P., & Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Proceedings of the Symposium on Security and Privacy (SP 2012), IEEE Computer Society, 553-567

[2] Jeffrey Jay Johnson, Steve Seixeiro, Zachary Pace, Giles van der Bogert, Sean Gilmour, Levi Siebens, and Kenneth Tubbs. 2014. Picture Gesture Authentication. Retrieved from https://www.google.com/patents/US8910253

[3] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[4] Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T. *et al.* Selective Search for Object Recognition. *Int J Comput Vis* **104,** 154–171 (2013). https://doi.org/10.1007/s11263-013-0620-5

[5] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick; *Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017*, pp. 2961-2969

[7] R. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.

[8] Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*

[9] Elizabeth Stobert and Robert Biddle. 2013. Memory retrieval and graphical passwords. In *Proceedings of the Ninth Symposium on Usable Privacy and Security* (*SOUPS '13*). Association for Computing Machinery, New York, NY, USA, Article 15, 1–14. DOI:https://doi.org/10.1145/2501604.2501619

[10] L. Huestegge and L. Pimenidis. 2014. Visual Search in Authentication Systems Based on Memorized Faces: Effects of Memory Load and Retention Interval. *International Journal of Human-Computer Interaction* 30, 7: 604–611. https://doi.org/10.1080/10447318.2014.907464

[11] Katherine M. Everitt, Tanya Bragin, James Fogarty, and Tadayoshi Kohno. 2009. A comprehensive study of frequency, interference, and training of multiple graphical passwords. In *Proceedings of the 27th international conference on Human factors in computing systems* - CHI 09, 889. https://doi.org/10.1145/1518701.1518837

[12] Weizhi Meng, Wenjuan Li, Lijun Jiang, and Liying Meng. 2016. On Multiple Password Interference of Touch Screen Patterns and Text Passwords. In *Chi '16*, 4818–4822. https://doi.org/10.1145/2858036.2858547

[13] Sonia Chiasson, Alain Forget, Robert Biddle, and P. C. van Oorschot. 2009. User interface design affects security: Patterns in click-based graphical passwords. I*nternational Journal of Information Security* 8, 6: 387–398. https://doi.org/10.1007/s10207-009-0080-7

[14] Katsini, C., Fidas, C., Raptis, G.E., Belk, M., Samaras, G. and Avouris, N., 2018, April. Influences of human cognition and visual behavior on password strength during picture password composition. In *Proceedings of the 2018 CHI conference on human factors in computing systems* (pp. 1-14).

[15] Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1985. *Learning internal representations by error propagation* (No. ICS-8506). California Univ San Diego La Jolla Inst for Cognitive Science.

[16] McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133 (1943). https://doi.org/10.1007/BF02478259

[17] Srivastava, R.K., Greff, K. and Schmidhuber, J., 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

[18] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[19] Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[20] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

[21] Difference Hashing, retrieved from http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html

[22] Constantinides, A., Belk, M., Fidas, C. and Samaras, G., 2018, July. On Cultural-centered Graphical Passwords: Leveraging on Users' Cultural Experiences for Improving

Password Memorability. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization* (pp. 245-249).

[23] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

[24] Everingham, M., Van Gool, L., Williams, C.K., Winn, J. and Zisserman, A., 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*(2), pp.303-338.

[25] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. and Schiele, B., 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3213-3223).

[26] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. and Desmaison, A., 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems* (pp. 8026-8037).

[27] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (pp. 265-283).

[28] Wu, Y., Kirillov, A., Massa, F., Lo, W.Y. and Girshick, R., 2019. Detectron2.

[29] A. Adams and M. Sasse, “Users Are Not The Enemy,” Commun. ACM, vol. 42, no. 12, pp. 41–46, 1999.

[30] Paivio, A., Rogers, T. B., & Smythe, P. C. (1968). Why are pictures easier to recall than words?. Psychonomic Science, 11(4), 137-138.

[31] Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine learning*, *20*(3), pp.273-297.

[32] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. 2012. Saliency filters: Contrast based filtering for salient region detection. *IEEE Conference on*

Computer Vision and Pattern Recognition, 733-740. DOI:https://doi.org/10.1109/CVPR.2012.6247743

[33] Nielsen, J., 1994, April. Usability inspection methods. In Conference companion on Human factors in computing systems (pp. 413-414).

Appendix A

Installing and running the server

First enter into **apps/website** folder and enter the following commands

```
apt-get update
apt-get install docker docker-compose
docker-compose build
```

Now the container is built. In order to start the server and install database migrations, execute the following commands:

```
docker-compose up
docker ps
docker exec -ti <web-container-id> /bin/bash
cd apps/website
python3 manage.py makemigrations
python3 manage.py migrate
exit
```

To stop the server enter the following command in **apps/website**

```
docker-compose stop
```

To start the server execute start the container, find the container id, enter in and start celery with the following commands

```
docker-compose up
docker ps
docker exec -ti <web-container-id> /bin/bash
```

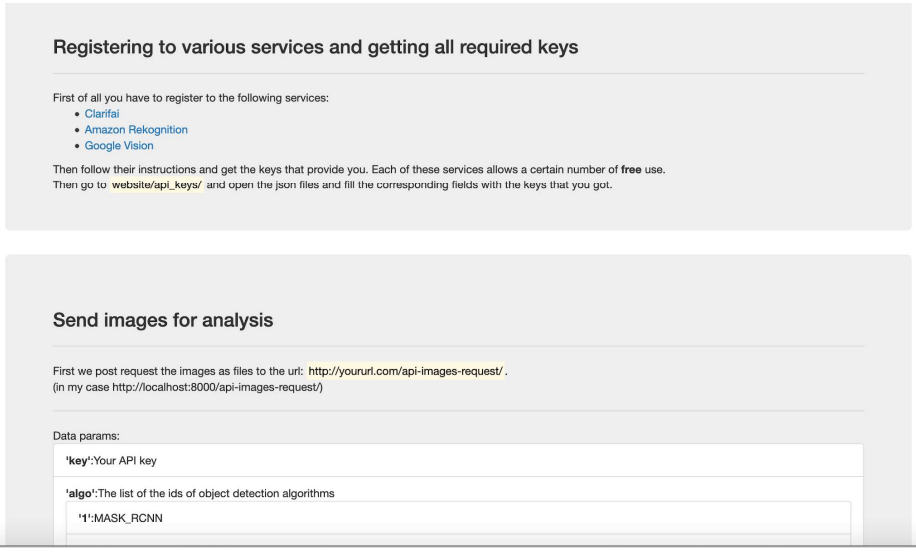
```
./init_celery.sh
```

If everything goes well, you can navigate to **localhost:8000** and view the homepage

Appendix B

Apart from the user interface for analyzing the images and passwords, the RESTful API is also exposed.

Below the API documentation is shown.



Registering to various services and getting all required keys

First of all you have to register to the following services:

- Clarifai
- Amazon Rekognition
- Google Vision

Then follow their instructions and get the keys that provide you. Each of these services allows a certain number of **free** use.
Then go to `website/api_keys/` and open the json files and fill the corresponding fields with the keys that you got.

Send images for analysis

First we post request the images as files to the url: `http://yoururl.com/api-images-request/` .
(in my case http://localhost:8000/api-images-request/)

Data params:

'key':Your API key

'algo':The list of the ids of object detection algorithms

'1':MASK_RCNN

'3':YoloV3
'4':TinyYoloV3

Also send your **jpg** images in **binary** format.

Server Response:

If all the things posted are valid , then that means that the images have been uploaded to the server and the analysis has began.
You will receive in the key **'request_url'** a url where you will post again with param **'key'** your API key to view the progress. (*16e17272-6f15-44d0-a86e-a3000d025685*).
The server will respond with the following keys:

'completed_tasks': All the tasks that have been completed
'all_tasks' : All the tasks that have been analyzed or will be analyzed

You can periodicaly post request to this url to see the progress
When all tasks are completed **('completed_tasks'=='all_tasks')** , you will receive the key **'images'**:

Sample json server response when all images have been completed ('images' key)


```
  {
"images": [
    {
        "MASK_RCNN": [
            {
                "accuracy": "0.78294176",
                "bound_xmax": 571,
                "bound_xmin": 47,
                "bound_ymax": 47,
                "bound_ymin": 47,
                "object_name": "person",
                "segment_size": 1
            },
            {
                "accuracy": "0.9946597",
                "bound_xmax": 543,
                "bound_xmin": 203,
                "bound_ymax": 404,
                "bound_ymin": 59,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
        "RetinaNet": [
            {
                "accuracy": "0.9761391",
                "bound_xmax": 396,
                "bound_xmin": 54,
                "bound_ymax": 555,
                "bound_ymin": 183,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
        "TinyYoloV3": [],
        "YoloV3": [
            {
                "accuracy": "0.9975961446762085",
                "bound_xmax": 553,
                "bound_xmin": 179,
                "bound_ymax": 405,
                "bound_ymin": 44,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
        "amazon_rekognition": [
            {
                "accuracy": "93.61317443847656",
                "label_name": "Canine"
            },
            {
                "accuracy": "93.61317443847656",
                "label_name": "Mammal"
            },
            {
                "accuracy": "93.61317443847656",
                "label_name": "Animal"
            },
        ],
        "clarifai": [
            {
                "accuracy": "99",
                "label_name": "dog"
            },
            {
                "accuracy": "98",
                "label_name": "cute"
            },
            {
                "accuracy": "98",
                "label_name": "pet"
            },
            {
                "accuracy": "97",
                "label_name": "animal"
            }
        ],
        (same goes for the rest of the images)
    }
  ]
}
```

Sample python script requesting 3 images with search key 'dog'

```
import requests
import os
import json
import time
REQUEST_URL = "http://localhost:8000/google-search-query/"
API_KEY = '16e17272-6f15-44d0-a86e-a3000d025685'
images = []

#here we request 3 images analyzed with MASK_RCNN and RetinaNet from google with the keyword dog
r = requests.post(REQUEST_URL,data={'key':API_KEY,
                                    'num_images':3,
                                    'algorithms':['1','2'],
                                    'search_key':'dog'})
result_dict = json.loads(r.content.decode())
print(result_dict)
"""server has responded with a request url, now you can periodicaly send a request to that
request url to see the progress of the analysis"""
if result_dict['is_valid']:#if the request was valid
    request_url = result_dict['request_url']
    while True:
        r = requests.post(request_url,data={'key':API_KEY})
        result_dict = json.loads(r.content.decode())
        if result_dict['state']=='SUCCESS' or result_dict['state']=='FAILURE':
            #break if analysis has finished
            break
        print(result_dict)
        #sleep 10 seconds and retry if not all the images have not been analyzed yet
        time.sleep(10)
    #Finaly you've got the results.
    for image in result_dict['images']:
        for key,val in image.items():
            print(key+":")
            print(val)
```

Request images from google

You can give a keyword and get back images found in google with their analysis details

First we post request the images as files to the url: **http://yoururl.com/google-search-query/**.
(in my case http://localhost:8000/google-search-query/)

Data params:

- **'key'**:Your API key
- **'search_key'**:The search key that will be searched
- **'num_images'**:How many images you want
- **'algorithms'**:The list of the ids of object detection algorithms
  - **'1'**:MASK_RCNN
  - **'2'**:RetinaNet
  - **'3'**:YoloV3
  - **'4'**:TinyYoloV3

Server Response:

a3000d025685).
The server will respond with the following keys:

'state': The state of the analysis.Possible states:

- **PENDING**:The whole task has not yet started.
- **PROGRESS**:The whole task is in progress
- **SUCCESS**:The whole task has finished successfuly
- **FAILURE**:Something went wrong during the whole process

'details' : Details about the current image that is being analyzed

- **'current'**:How many images have been analyzed so far
- **'total'**:The total number of images requested
- **'current_algo'**:How many object detection algorithms have been used so far
- **'total_algos'**:The total number of object detection algorithms that will be used
- **'note'**:A note regarding the current state of the analysis

You can periodicaly post request to this url to see the progress
When all tasks are completed **('state'=='SUCCESS' or 'state'=='FAILURE')** , you will receive the key **'images'** that contains all the analysis results.

It's exactly the same as the user image request


```
  {
"images": [
    {
        "MASK_RCNN": [
            {
                "accuracy": "0.78294176",
                "bound_xmax": 571,
                "bound_xmin": 194,
                "bound_ymax": 394,
                "bound_ymin": 47,
                "object_name": "person",
                "segment_size": 1
            },
            {
                "accuracy": "0.9946597",
                "bound_xmax": 543,
                "bound_xmin": 203,
                "bound_ymax": 404,
                "bound_ymin": 59,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
        "RetinaNet": [
            {
                "accuracy": "0.9761391",
                "bound_xmax": 396,
                "bound_xmin": 54,
                "bound_ymax": 555,
                "bound_ymin": 183,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
            {
                "accuracy": "0.9975961446762085",
                "bound_xmax": 553,
                "bound_xmin": 179,
                "bound_ymax": 405,
                "bound_ymin": 44,
                "object_name": "dog",
                "segment_size": 1
            }
        ],
        "amazon_rekognition": [
            {
                "accuracy": "93.61317443847656",
                "label_name": "Canine"
            },
            {
                "accuracy": "93.61317443847656",
                "label_name": "Mammal"
            },
            {
                "accuracy": "93.61317443847656",
                "label_name": "Animal"
            },
        ],
        "clarifai": [
            {
                "accuracy": "99",
                "label_name": "dog"
            },
            {
                "accuracy": "98",
                "label_name": "cute"
            },
            {
                "accuracy": "98",
                "label_name": "pet"
                "label_name": "animal"
            }
        ],
        (same goes for the rest of the images)
    }
  ]
}
```


Appendix C

Technologies used:

Flutter - https://flutter.dev/

Django - https://www.djangoproject.com/

RabbitMQ - https://www.rabbitmq.com/

Docker - https://www.docker.com/

Celery - https://docs.celeryproject.org/en/stable/

Tensorflow - https://www.tensorflow.org/

Keras - https://keras.io/

PyTorch - https://pytorch.org/