AcCAPTCHA

Edoardo De Matteis

Contents

1 Introduction

Nowadays when surfing the web CAPTCHA tests are everywhere, to the point we are used to get asked to prove our humanity and something is off when it does not happen. Furthermore, technology moves forwards and tests are becoming increasingly harder, while it may be a nuisance for the abled it can often be a barrier for people with disabilities. It is a priority to develop reliable CAPTCHA tests designed to be accessible to everyone.

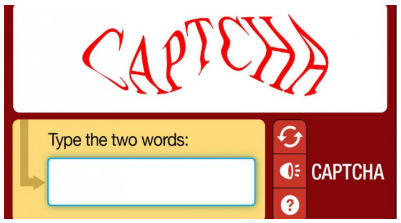
The word CAPTCHA stands for *Completely Automated Public Turing test to tell Computers and Humans Apart*. In the year 2000 during a speaking at Carnegie Mellow University held by YAHOO!, at the time one of the world's biggest companies, they listed the ten most relevant problems they did not know hot to solve; among these issues stood out that spammers would write programs to automatically generate mails, and use them for malicious purposes [11, 5]. The world needed a test to distinguish humans from computers such that any human, regardless of age, gender, education or language could pass it. To make things even more difficult, the CAPTCHA test is born from a paradoxical idea: any computer should be able to grade it but should not be able to pass it, like if a teacher could grade students' exams without knowing the subject.

CAPTCHA The idea exposed in [1] is based on the fact that humans are extremely good at pattern and optical character recognition i.e., reading, and we generally learn to do it since we are kids and can do it under many circumstances like bad lighting and other. A text is took and warped, stretched or generally modified in some way such tha computer vision techniques at the time could not read it, yet the computer with the gold truth just has to match it with the user's input. Once released CAPTCHA has been used to authenticate access to YAHOO! mail services.

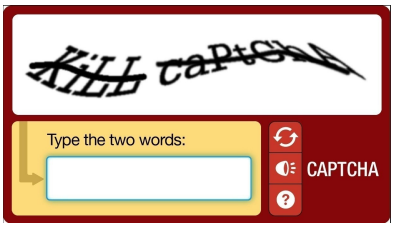
reCaptcha All the character input by humans into CAPTCHA gave new data that made computers smarter. The new versione, reCAPTCHA [2], uses two words: one generated so the compute knows the answer and the other pulled from a book or originally a *The New York Times*' old article, both of which are distorted. When a human gets the first word right the system guesses the second one is probably right and distributes it across different users, if there is consensus then it is approved. So many tests were done that something along one year of NY Times articles was digitalized every 4 days, this led Google to acquire reCAPTCHA in 2009 and exploit it to digitalize texts in its archive. When you do that enought times you get a big enough dataset of distorted characters that computers can effectively learn how to read them, in a way CAPTCHA taught machines how to read highly warped texts, in a 2014 machine learning study Google stated how humans have a 33% accuracy on recognizing CAPTCHA's warped characters while a trained machine has 99.8% [10].

reCAPTCHAv2 Given the huge potential reCAPTCHA showed to have, Google exploited it to have human users classify images and build up a data, that were later used to improve Google Maps [11]. Again, computers eventually became good enough on images that reCAPTCHAv2 is not enough anymore.

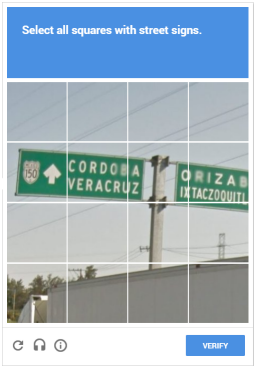
NoCAPTCHA reCAPTCHAv3 Computers changed and tests have to change as well, NoCAPTCHA reCAPTCHAv3 [9] are behavioral-based: a background test is always running and if it detects some bot-like behavior e.g., clicking around or writing large chunks of text in a very short time, it makes you take a standard test. Alternatively, instead of a test it aks for authentication, based on the idea that is possible to tell you are not a robot given that is possible to tell who you are.



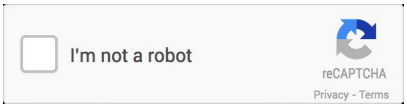
(a) CAPTCHA.



(b) ReCAPTCHA.



(c) ReCAPTCHAv2.



(d) NoCAPTCHA.

Figure 1: Visualizationof the different CAPTCHA tests presented in this historical review.

The CAPTCHA problem is far from solved and, realistically, at some point the current CAPTCHA test will be obsolete. A problem present in all the test instances we saw is accessibility e.g., no deafblind person could pass one of them. Speaking from personal experience, while surfing the web I noticed that there are very few options for disabled people: once I found the possibility to select an audio test but still the default selection was a classical visual CAPTCHA.

2 Methodology

I propose AcCAPTCHA, a starting point for an accessible CAPTCHA system, such that the user selects what they can do and are then subjected to a corresponding test. The system is accessible through a browser[1], as it would be realistically desired, and currently supports the following tests:

Text recognition A classical text recognition, as in CAPTCHA and ReCAPTCHA, in which the user reads a warped text and has to input it to the system, see figure 28. Data were collected from [12, 7] and we perform a simple text matching.

Image classification One of Luis von Ahn's original ideas for CAPTCHA was to show some image to the user and have them describe it, yet humans proved to be very bad; as an example, given an image of a car people would also say something like "tires", "vehicle", "automobile" and so on, and they would all be correct. Taking inspiration from that I set the up a closed-response object classification task as in figure 29, images are took from the *Natural Images* dataset used in [8]

Finger counting The two previous tasks are highly dependend on sight and do not add any accessibility to the system, thus we include a behavioral task: the user is shown a number and a rectangle on the screen and is asked to replicate it with their fingers (figure 31), if they do it correctly for a customizable interval of time then the test is passed. The *MMPOSE* toolkit [6] implements hand pose recognition and even though the model has good performances (AUC of 84% and PCK@0.2 to 81%) there are still issues with too high or low illumination, a known problem in computer vision. The way in which we count finger is based on keypoints position i.e. when a finger's phalanx keypoint is over its respective knuckle we increment our counter by one. Doing that we assume the user adopts the western way of finger counting, cultural differences make it impossible to have a universal system: in many regions of Asia the thumb acts as a pointer counting on phalanxes (figure 2), particular to the Korea Peninsula is *chisanbop* which allows to count up to 99 based on the assignment shown in figure 3. In western countries things are a bit easier to address and each finger is associated to a unit, even though it was not always like that (figure 4), there are still some "regional" variations 5 that the system can take care of.

Word reading This task makes use of voce, the system randomly selects a customizable number of words from [4] and the user reads them, see figure 30. The system uses *Google Cloud Speech API* [3], mainly because is the easiest one to use and does not need us to authenticate for an API token. Yet be wary that it is not suitable for non-private use cases, and it does not perform good when the user has a non-english accent e.g., italian in our case.

[1] It is per now not on the internet and can be accessed only locally.

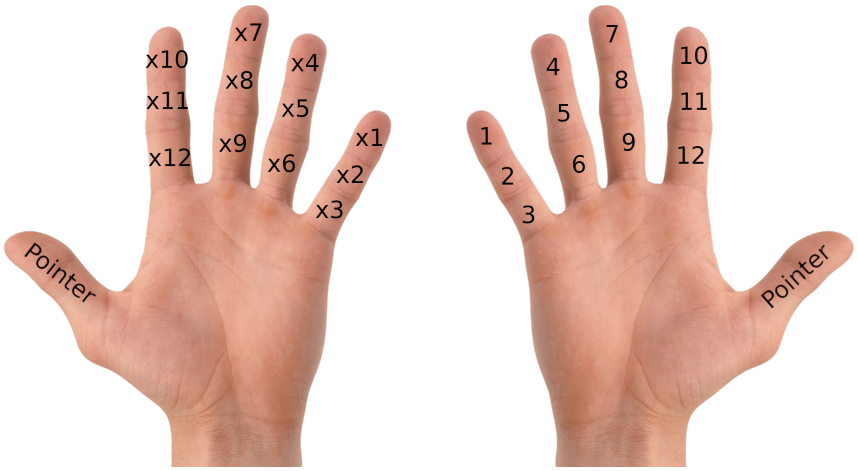
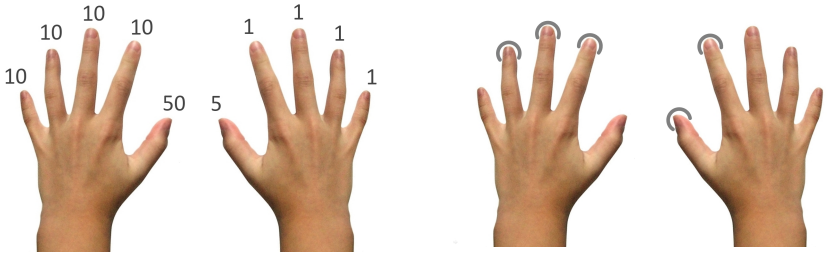


Figure 2: In some regions of Asia finger counting is done on base-12. One hand keeps track of the number and the other acts as a multiplier, up to 144.



(a) How to count in chisanbop.

(b) Thirty-six in chisanbop.

Figure 3: Chisanbop numbering system.

When a user accesses the system they have to configure the system according on which tests they can or want to be submitted to, as shown in figure 27, by default all chechboxes are not checked. The way it is done is associating each test to a propositional logic conjuction of checkbox flags, then one test is randomly selected from the set of tests whose formula is true.

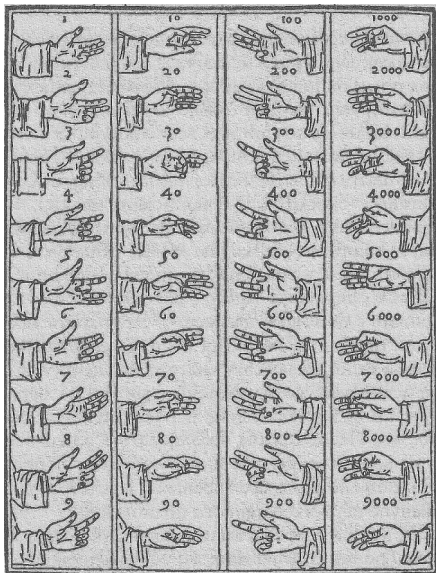


Figure 4: Finger numbering system to count from 1 to 9999 from *Summa de arithmetica*, Luca Pacioli, 1494.

(a) Number three as done in Great Britain.

(b) Number three as done in continental Europe.

Figure 5: Different variations of the number three in Europe, as shown in Quentin Tarantino's *Inglorious Basterds*.

3 Architecture

We take a look at how the system is designed.

3.1 Actors

The actors in the system are the ones in table 1.

Table 1: Actors in the system.

Name	**Description**
User	The test subject that uses the system.
MMPOSE	The MMPOSE external system used for hand-pose estimation.
Google Speech API	The Google API used for speech-to-text conversion.

3.2 Requirements

Requirements are classified based on their priority: *mandatory* or *recommended* and *desirable*.

Table 2: *Configure system* requirement.

Id	REQ01F
Usecase	Configure system
Type	Functional
Priority	Mandatory
Description	The user should be able to manage which tests will be performed, based on their accessibility requirements.

Table 3: *Text recognition* requirement.

Id	REQ02F
Usecase	Text recognition
Type	Functional
Priority	Mandatory
Description	The user should be able to be subjected to a CAPTCHA-like automated test.

Table 4: *Image classification* requirement.

Id	REQ03F
Usecase	Image classification
Type	Functional
Priority	Mandatory
Description	The user should be able to be subjected to an image classification automated test.

Table 5: *Finger counting* requirement.

Id	REQ04F
Usecase	Finger counting
Type	Functional
Priority	Recommended
Description	The user should be able to a behavioral finger countng automated test.

Table 6: *Word reading* requirement.

Id	REQ05F
Usecase	Word reading
Type	Functional
Priority	Recommended
Description	The user should be able to be subjected to word reading speech-to-text automated test.

Table 7: *Word reading* requirement.

Id	REQ06F
Usecase	Hearing
Type	Functional
Priority	Desirable
Description	The user should be able to be subjected to a word hearing automated test.

Table 8: *Web access* requirement.

Id	REQ07NF
Usecase	Web access
Type	Non functional
Priority	Desirable
Description	The user should be able to easily access the system through the web.

Table 9: *Client access* requirement.

Id	REQ08NF
Usecase	Client access
Type	Non functional
Priority	Desirable
Description	The user should be able to easily access the system through a client.

Table 10: *Portability* requirement.

Id	REQ09NF
Usecase	Portability
Type	Non functional
Priority	Desirable
Description	The system should work both on desktop and mobile, on the main browsers: • *Google Chrome* 105.0.5195.102 • *Microsoft Edge* 100.0.1185.50 • *Mozilla Firefox* 104.0.1 • *Opera* 90.0.4480.80 • *Safari* 15.6.1

Table 11: *Performance* requirement.

Id	REQ10NF
Usecase	Performance
Type	Non functional
Priority	Desirable
Description	The system should be fast enough to be easily usable by the user.

Table 12: *Customization* requirement.

Id	REQ11NF
Usecase	Customization
Type	Non functional
Priority	Desirable
Description	The system should allow the user to select the appearance of the system, more precisely the following color-palette should be available: • Light theme. • Dark theme. • Black and white. • Protanopia. • Deuteranopia. • Tritanopia.

Table 13: *Security* requirement.

Id	REQ11NF
Usecase	Security
Type	Non functional
Priority	Mandatory
Description	The system's tests hould be strong against bot attempting at them.

3.3 Use cases

We show in figure 6 the usecase diagram, tables 14-25 describes usecases and figures 7-18 are the respective activity diagrams.

3.3.1 Usecase diagram

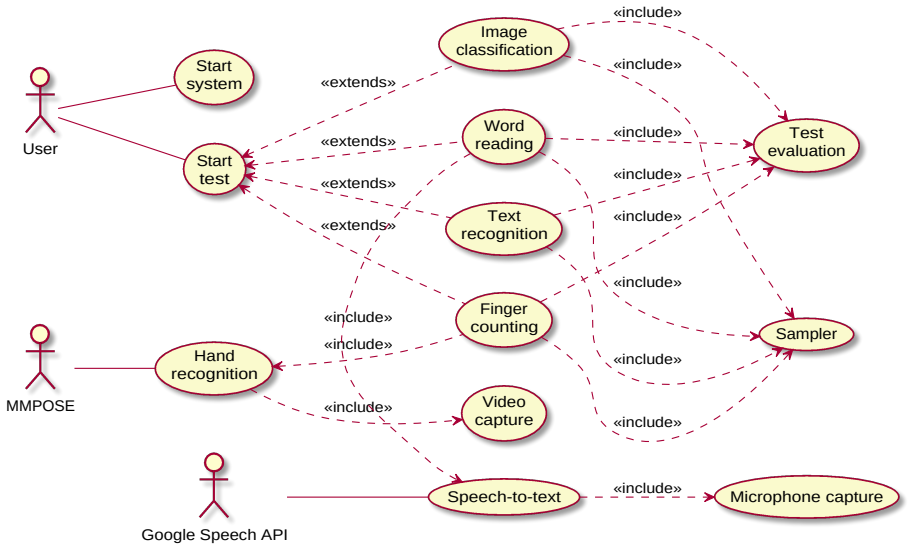


Figure 6: Usecase diagrams

3.3.2 Usecase descriptions

Table 14: *Start system* usecase.

Id	UC01
Usecase	Start system.
Description	The system starts.
Actor	User
Precondition	The system has not started yet.
Events	1. The user starts the client. 2. The user presses the ”start” button.
Postcondition	The system has started.

Table 15: *Start test* usecase.

Id	UC02
Usecase	Start test.
Description	The user selects which devices of modes they want to use and gets presented a test.
Actor	User
Precondition	The system has started.
Events	1. The user selects which checkboxes they want to be true. 2. The user presses the ”submit” button. 3. The system shows a test to the user.
Postcondition	The user has been submitted to a test.

Table 16: *Text recognition* usecase.

Id	UC03
Usecase	Text recognition.
Description	The system performs a text recognition test.
Actor	-
Precondition	The user has expressed to want a test to be started.
Events	1. The system shows a CAPTCHA-like image to a user. 2. The user selects writes the perceived text into the box. 3. The user submits the answer. 4. The system evaluates the output. 5. The system redirects the user to the home page.
Postcondition	The user has passed or not the test.

Table 17: *Image classification* usecase.

Id	UC04
Usecase	Image classification.
Description	The system performs an image classification test.
Actor	-
Precondition	The user has expressed to want a test to be started.
Events	1. The system shows an image to a user. 2. The user selects a class. 3. The user selects submit. 4. The system evaluates the output. 5. The system redirects the user to the home page.
Postcondition	The user has passed or not the test.

Table 18: *Finger counting* usecase.

Id	UC05
Usecase	Finger counting.
Description	The system performs an finger counting test.
Actor	-
Precondition	The user has expressed to want a test to be started.
Events	1. The system shows a number and to the user. 2. The system opens the webcam with a coloured region-of-interest. 3. The user shows the number with their hand inside the RoI. 4. The system waits for a custom number of correct guesses. 5. The system redirects the user to the home page.
Postcondition	The user has passed or not the test.
Alternative events	4. The user does not show the correct number exceeding the timeout.

Table 19: *Word reading* usecase.

Id	UC06
Usecase	Word reading.
Description	The system performs a word reading test.
Actor	-
Precondition	The user has expressed to want a test to be started.
Events	1. The system shows some words to a user. 2. The user presses the "Record" button. 3. The user reads the words aloud. 4. The system converts the speech to text. 5. The system matches the converted text with ground truth words. 6. The system redirects the user to the home page.
Postcondition	The user has passed or not the test.

Table 20: *Test evaluation* usecase.

Id	UC07
Usecase	Test evaluation.
Description	The system evaluates the test performed by the user.
Actor	-
Precondition	The user has performed a test.
Events	The system compares the user input with the ground truth and returns a value.
Postcondition	-

Table 21: *Sampler* usecase.

Id	UC08
Usecase	Sampler.
Description	The system randomly samples some data that should be presented to the user.
Actor	-
Precondition	The user has performed a test.
Events	The system samples data randomly and returns it.
Postcondition	-

Table 22: *Hand recognition* usecase.

Id	UC09
Usecase	Hand recognition.
Description	MMPOSE recognized an hand.
Actor	MMPOSE
Precondition	The webcam is activated.
Events	MMPOSE recognizes an hand skeleton and returns it.
Postcondition	-

Table 23: *Video capture* usecase.

Id	UC10
Usecase	Video capture.
Description	The camera records a video and shows it.
Actor	-
Precondition	The webcam is activated.
Events	The camera shows the live recorded video.
Postcondition	-

Table 24: *Speech-to-text* usecase.

Id	UC11
Usecase	Speech-to-text.
Description	The Google Speech API converts a recorded speech to text.
Actor	Google Speech API
Precondition	The microphone is activated.
Events	The Google Speech API converts speech to text and returns it.
Postcondition	-

Table 25: *Microphone capture* usecase.

Id	UC12
Usecase	Microphone capture.
Description	The microphone records a voice.
Actor	-
Precondition	The microphone is activated.
Events	The microphone records a voice.
Postcondition	-

3.3.3 Activity diagrams

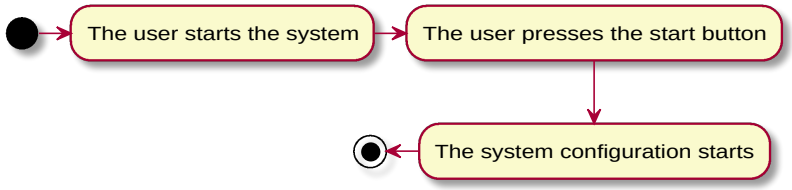


Figure 7: *Start system* activity diagram.

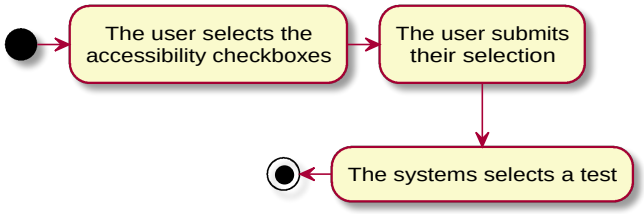


Figure 8: *Start test* activity diagram.

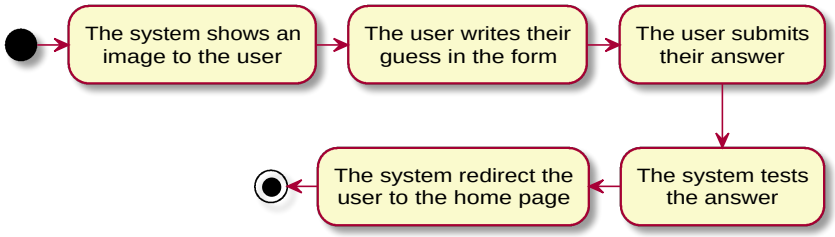


Figure 9: *Text recognition* activity diagram.

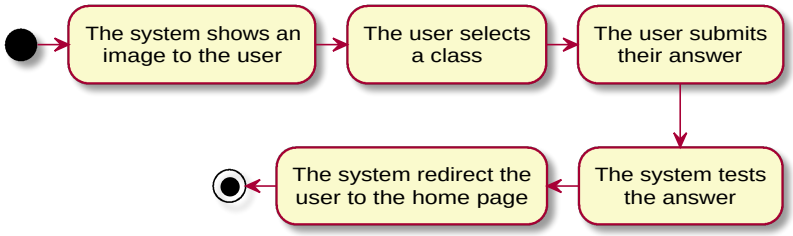


Figure 10: *Image classification* activity diagram.

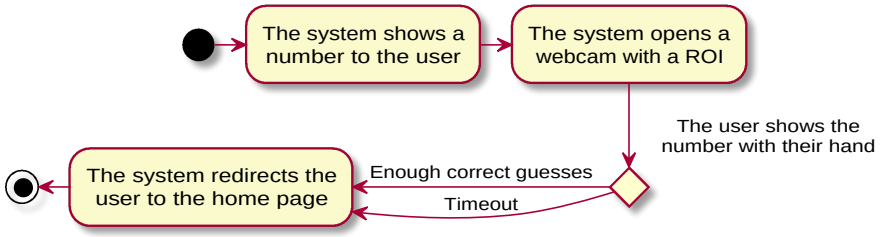


Figure 11: *Finger counting* activity diagram.

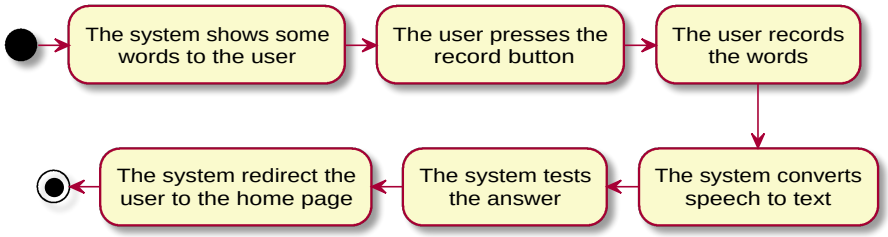


Figure 12: *Word reading* activity diagram.

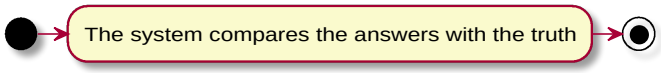


Figure 13: *Test evaluation* activity diagram.

Figure 14: *Sampler* activity diagram.

Figure 15: *Hand recognition* activity diagram.

Figure 16: *Video capture* activity diagram.

Figure 17: *Speech-to-text* activity diagram.

Figure 18: *Microphone capture* activity diagram.

3.3.4 Class diagram

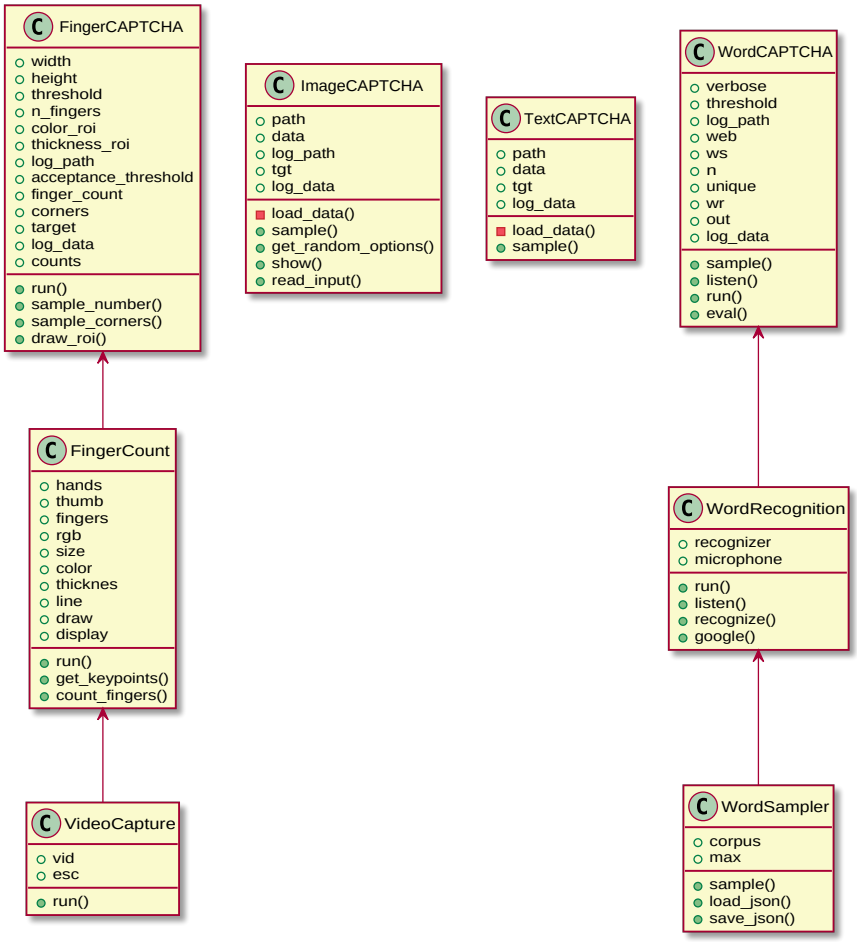


Figure 19: Class diagram.

3.3.5 Sequence diagrams

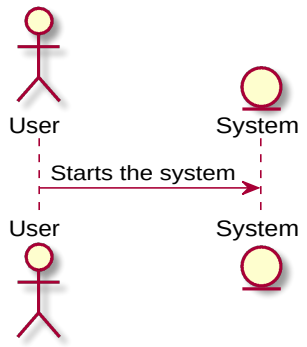


Figure 20: *Start system* sequence diagram.

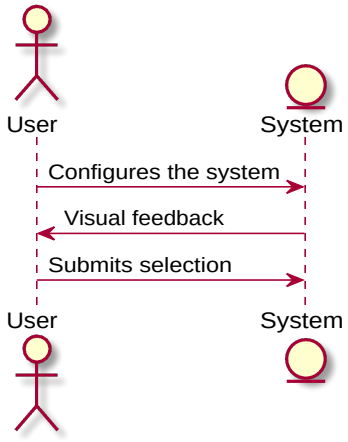


Figure 21: *Start test* sequence diagram.

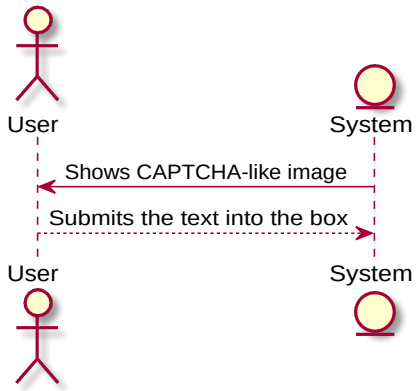


Figure 22: *Text recognition* sequence diagram.

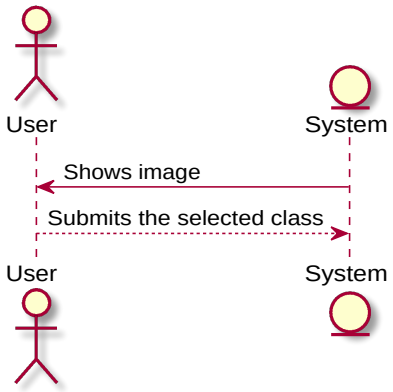


Figure 23: *Image classification* sequence diagram.

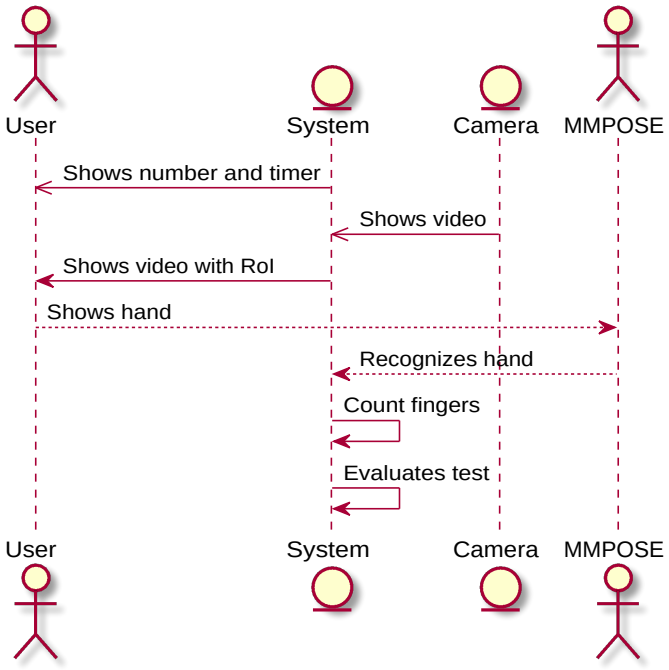


Figure 24: *Finger counting* sequence diagram.

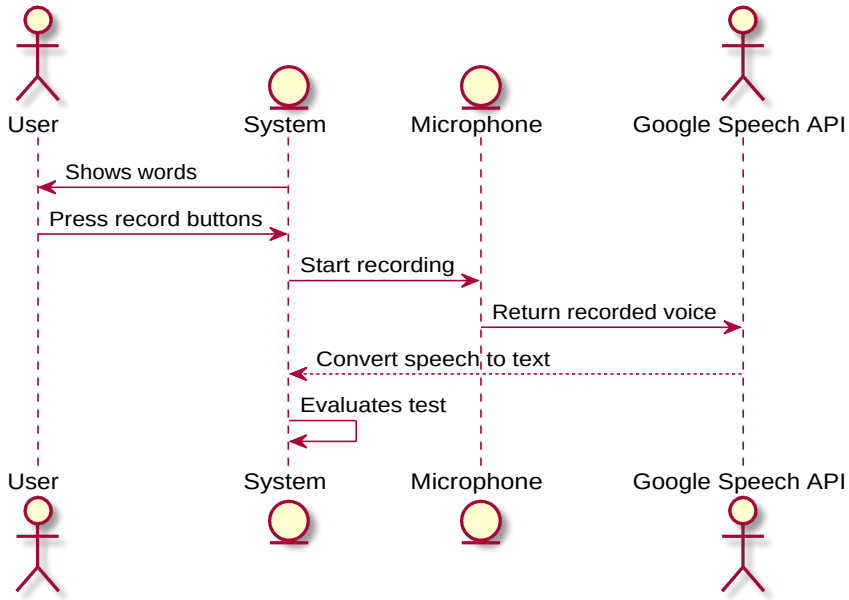


Figure 25: *Word reading* sequence diagram.

4 Interface

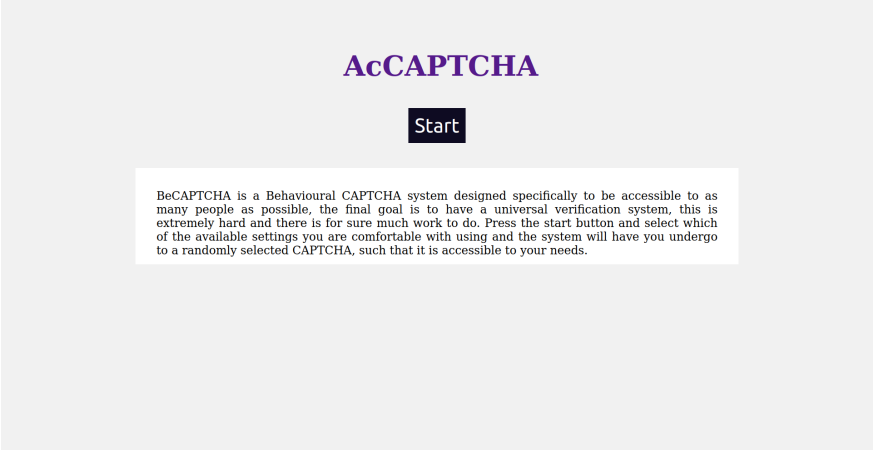


Figure 26: Home page.

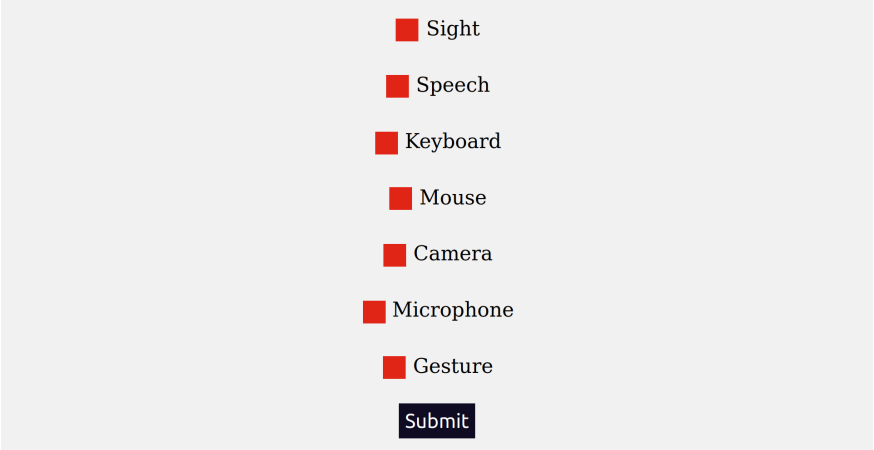


Figure 27: Accessibility selection window.

Figure 28: Text recognition task.

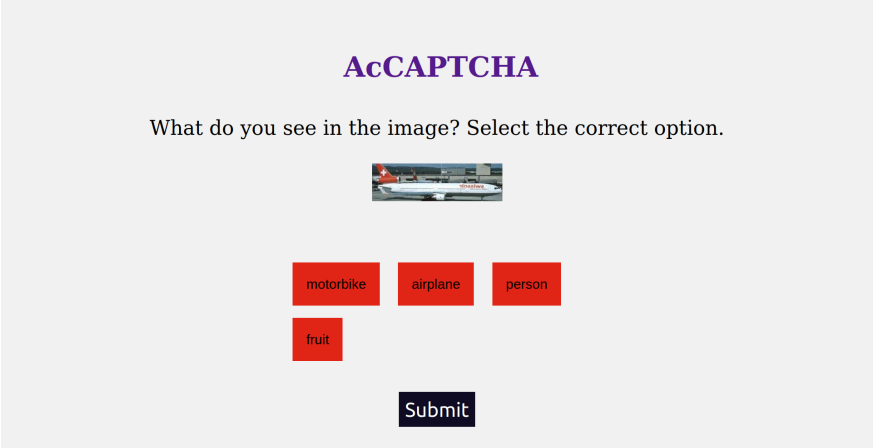


Figure 29: Image classification task.

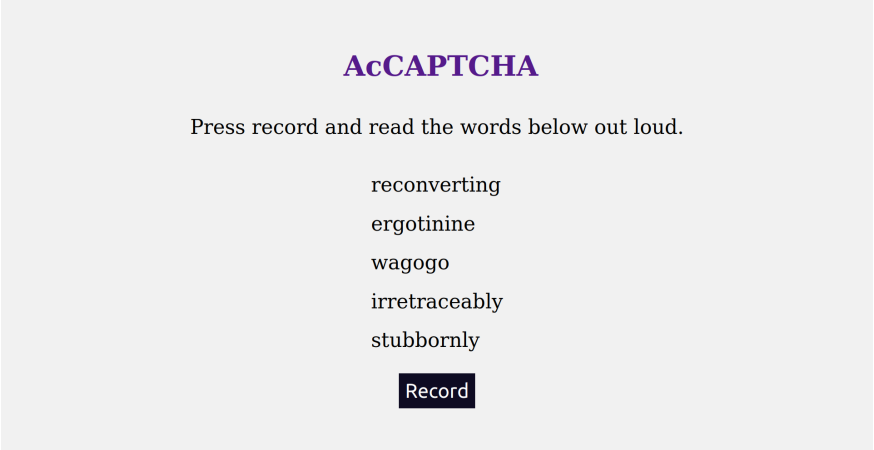


Figure 30: Word reading task.

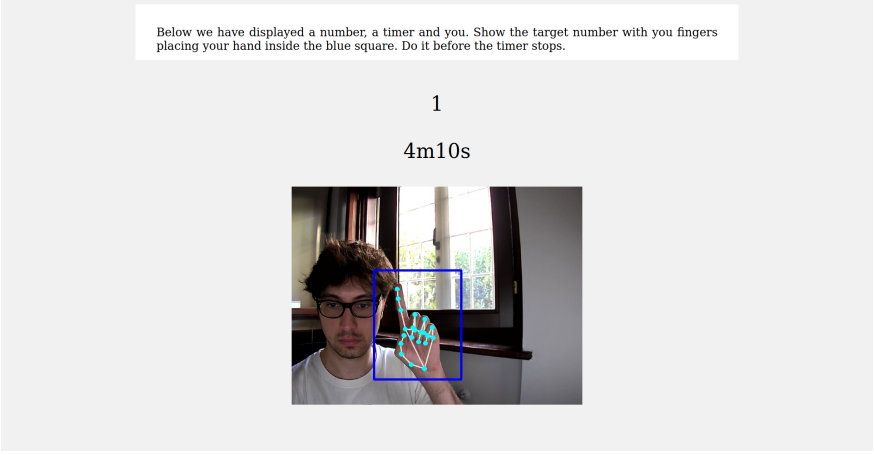


Figure 31: Finger counting task.

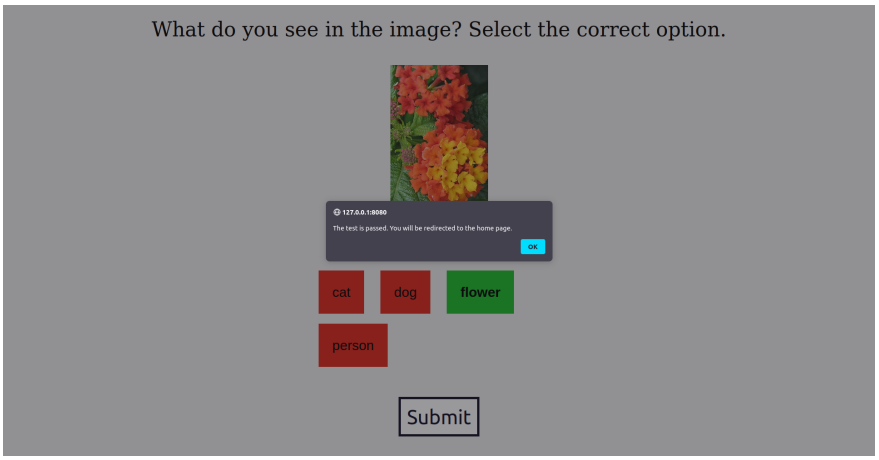


Figure 32: Succed alert on the image classification task.

References

[1] Luis von Ahn et al. "CAPTCHA: Using Hard AI Problems for Security". In: *Advances in Cryptology — EUROCRYPT 2003*. Ed. by Eli Biham. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 294–311. ISBN: 978-3-540-39200-2.

[2] Luis Ahn et al. "reCAPTCHA: Human-based character recognition via Web security measures". In: *Science (New York, N.Y.)* 321 (Sept. 2008), pp. 1465–8. DOI: 10.1126/science.1160379.

[3] Google APIs. *Cloud Speech API*. https://github.com/googleapis/python-speech. 2022.

[4] dwyl. *List Of English Words*. https://github.com/dwyl/english-words. 2022.

[5] Josh Dzieza. *Why CAPTCHAs have gotten so difficult*. https://www.theverge.com/2019/2/1/18205610/google-captcha-ai-robot-human-difficult-artificial-intelligence. 2019.

[6] MMPose Contributors. *OpenMMLab Pose Estimation Toolbox and Benchmark*. Aug. 2020. URL: https://github.com/open-mmlab/mmpose.

[7] Grégoire Passault. *Captcha*. https://github.com/Gregwar/Captcha. 2022.

[8] Prasun Roy et al. "Effects of Degradations on Deep Neural Network Architectures". In: *CoRR* abs/1807.10108 (2018). arXiv: 1807.10108. URL: http://arxiv.org/abs/1807.10108.

[9] Vinay Shet. *Are you a robot? Introducing "No CAPTCHA reCAPTCHA"*. https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html. 2014.

[10] Vinay Shet. *Street View and reCAPTCHA technology just got smarter*. https://security.googleblog.com/2014/04/street-view-and-recaptcha-technology.html. 2014.

[11] Vox. *Why captchas are getting harder*. 2021. URL: https://www.youtube.com/watch?v=lUTvB1O8eEg&t=376s (visited on 05/14/2021).

[12] Rodrigo Wilhelmy and Horacio Rosas. *captcha dataset*. July 2013.