Web Based Document Scanner for AQA Computer Science NEA

Kourosh Simpkins

# Analysis:

General Background Information:

The current problem is that document storage in school is not very efficient. Currently, documents are given to students as paper handouts. These paper handouts are then put into bags and forgotten about. Using a computer would make life much easier for students as it would allow them to document everything more effectively, as well as not having to carry around nearly as much documentation.

Recording documentation digitally would allow for more information to be carried around effectively, as well as ensure that no worksheets were lost, either by students or even teachers. Now some may argue that it would be easier to implement this by just allowing students to access the original versions of these documents, but in that case a student would need to be trusted print the documents and then hand them in. In most cases it would be easier to just give them a sheet of paper with the work that they are supposed to do on it.

Prospective Users:

The prospective users for my application are any students who wish to use it, as well as any teachers who wish to use it within the school. To connect to it any user would have to connect to a network which would allow them to access the program without needing to download any extra data. Therefore, the system should be designed as a web application, as not many students will have knowledge in command line interfaces and it will be easier for me to serve the information to every user, rather than building packages for each OS as well as apps for mobile devices, thus making the solution device agnostic.

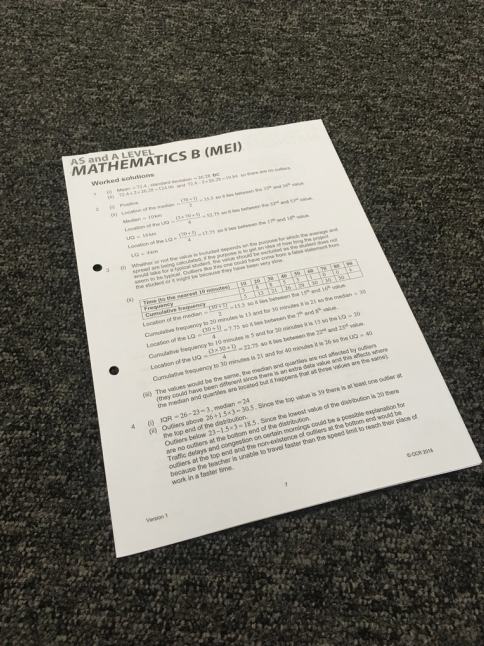
(initial) Proposed Solution:

The solution for this problem is quite simple. Build a document scanner! Something that is lightweight enough to be run from a server over a network, and robust enough to deal with stupid image inputs, as well as dealing with image inputs that wouldn’t necessarily work. The main purpose of the program will be to find a sheet of paper within the image, and perform a perspective transform on the sheet of paper so that it resembles more of a scanned image, in the sense that it takes up the whole image, rather than a portion of it. Certain aspects of the program may require input images to be taken with the object being scanned on a contrasting background (i.e. white paper on a black background), however this depends on the overall ability of the program.

Once the image has been found and returned as a ‘scanned’ document, the user would then be given the option to convert the text in the scanned image to a .txt file. The final files would then be shown to the user, and they should be given the option to download the files. When the user asks for a download, the files will be written to the same location on the disk and combined into a zip file which would then be sent to the user who requested the download. This would then allow the user to control what happens with their files, as they have full control over the documents and what they are called, where they are stored on the drive etc.

The majority of the image recognition techniques within this program will be performed with the help of OpenCV (Open Computer Vision), which has tools designed for specifying locations in an image. However, much of the program will also require a module to do the specific processing for the perspective transformation of the sheet of paper. I will also require the use of NumPy, to perform calculations for things that aren’t fully available in the standard Python Library.

In the case that the user decides that they wish to receive a .txt file, then the program should be able to determine if what it sees is actually what has been written on the page. Thus, I will require the use of a spell checker. In the case that the spellchecker catches an error with any of the written text, it should ask the user if the corrected response is correct. If the response is correct then the word is changed to the corrected word. In the case that the response is incorrect, the user can then ask to change it themselves or have the program leave the caught error as is (if the word written is not within the dictionary that is used to process the data)

A close up of text on a black background

Description automatically generatedThe processes performed by this program should turn the image on the left into the image on the right below: (fig. 1)

Acceptable Limitations:

Initial design requirements have led to the following limitations:

* The system will only be able to determine the contours of a single sheet of paper, which must be a quadrilateral, mostly because it is highly unlikely that someone will have an unusually shaped piece of paper and that they will try and scan any of the data that is on it.
* The system will only be able to process one sheet of paper at a time. If there is more than one document that a student wishes to scan, they should upload the images separately, as the program can only process one image at a time. To remedy this, I could add a function to the program that allows for multiple documents to be uploaded and scanned simultaneously.
* The function which converts text in the image to a .txt file will not be able to recognise complex mathematical formulae; however it may be possible to add this function to the program in the form of a checkbox, i.e. if there are mathematical formulae, the user ticks the box to ensure the correct part of the program is run.
* The system will only be accessible whilst a student is at school, as the network will be in school for students to access. It is possible to make it accessible from outside school, however, to do this would require implementation of cybersecurity, which isn’t covered by the scope of this project. Even so, it is completely possible for me to create this, but is a matter of time taken to create the program.

Objectives:

* Create a system that can recognise a piece of paper from its contours within an image.
* Create an internal server that can be accessed in school for users to send images that they want to be scanned.
* Create a front-end webpage that is easy for a user to navigate, where they can upload their files.
* Give the user an option to specify whether or not they want to receive a .txt file of the data from the image that they are using as their input.
* Ensure that any errors caught by the spell checker are shown to the user, and that they can tell the computer whether or not it is correct.

The system should be designed with accuracy in mind; however, speed is also of major importance, therefore there needs to be a fine balance between accuracy and solution speed. The speed of the system will also depend on the hardware that I decide to run my software on, as well as the load on the network, among other possible factors that come with implementing a system over a network and being remotely accessed.

Front End Proposed Solution:

Currently (as of 2/2/2020), the solution that is being followed is one that focuses on ease of access for each user, ensuring that they have the information that they require. To do this, I will utilise the Django python web framework to build a frontend website, which will allow students to log in and view previously uploaded files. This could also help teachers to cut back on the amount of paper they are printing in school, as they no longer have to give handouts to students, and can instead give them a link to the handout.

Due to the current nature of the design, it would therefore be beneficial to allow users to log in, so that they can see their previous uploads, and download them again if they need to. To do this, within the Django\_project there is an app called users, which allows a user to login and access the information that they need. To do this requires a database, in this case one user has one profile, but one user can have many uploads, but an upload can only belong to one user (I go into this further in Data sources and destinations). Going further with this we could allow users to have private uploads, but that currently adds unneeded complexity, particularly as this is a system designed for use in a school environment, so it is unlikely that anyone would attempt to upload anything compromising.

Due to the fact that the system should only be accessed in the school environment, it would be beneficial to ensure that anyone accessing the system is actually allowed to view the system. An easy way to do this is to check the email address of a registering user, to see whether or not the domain is a ‘@hfed.net’ email address. To further add security ensuring that no unauthorised users can access the network, it would be a requirement that a user validates their email address before they can access the application or document scanner, as well as hiding uploaded images from users who are not logged in.

The actual webpage will use bootstrap classes, as well as a few custom CSS classes. The overall UI does not need to be exceedingly complicated, as it needs to be intuitive for a user, even though all of the users should have experience with uploading images to a website. The server will also act to store data efficiently but could also be expanded upon to save a user’s image to their OneDrive account, as the school utilises OneDrive for business for all students and teachers, with each account having over one terabyte of storage. The downside of this is that the images could not be served to the user through the website once they have been exported, however it would allow the server to run faster, and would allow me to use a smaller storage media, as I wouldn’t need to focus on ensuring that all of the students data was safe.

## Data:

Data Sources and Destinations:

The website will utilise databases to control data about users who sign up but will use a file-based system for saving the images and serving them to the user (if they are not already stored within a user’s OneDrive. A file system will also be used to store user profile pictures, as is the convention with Django.

Website Database information:

|  |  |  |
| --- | --- | --- |
| What is it | Source | Destination |
| User information | New users signing up | Db.sqlite3 database |
| Admin information | Assigning admin privileges | Within auth\_user table |

Due to the fact that I will be using Django to serve a website to my users, I can use the built-in database functions of the Django module, where database ‘models’ are created, which Django then assigns to the root ‘db.sqlite3’ database. The information going into this database is then assigned to each table, wherever it is required. The two tables that will be used most will be the ‘auth\_user’ table and the ‘users\_profile’ table. Any table within the database which is prefixed by ‘django\_’ is a default table created by Django, and will be left untouched by me (these tables merely contain session information and logs from the Django admin page, they do not affect the way content is served or the site is run).

Data Volumes:

Enrolment at Harris City Academy Crystal Palace is 800[1], so I can assume that about 75% of students will sign up to use the system[2], therefore out of 800, 600 students (or there about) will enrol to use my document scanner to aid them in their studies. However, the system is designed to be accessible to anyone with a ‘@hfed.net’ email address to sign up with, therefore I can assume that about 75% of students and staff in the Harris Federation will sign up to use the system, which is about 30,300 users[3].

It is difficult to find a figure for the amount of processing that will need to be done, however, as this would be an arbitrary value. Even so it is probably safe to assume that the number of operations (i.e. image scans) required to be performed each day will not be prohibitively large, and due to the nature of OpenCV (being lightweight but powerful), one image operation takes very little time at all on simple images. A simple image being a white sheet of paper on a black background. Therefore, we can assume that image processing will need to be run in parallel (for the sake of redundancy), but that it will be nowhere near prohibitive limits for running on a raspberry pi 4 or two (my preferred hardware server). My assumption would be 1000 processes per day at the top end of image processing[4], but each image need only be processed once, so if a teacher has processed a document such as a test paper, a student need not perform the process again, instead following the link to the document.

The system may need to be able to handle several processes at a time, purely due to the fact that it is a web-based system. I currently see no way for me to handle this without buying time on a server (which I do not wish to do), but I may be able to build a cluster utilising a Python module ‘DispyNode’[5] which aids in running multiple processes across multiple hardware devices, specifically the Raspberry Pi.

Data Dictionary:

auth\_user table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
| Password [6] | Stores a User’s hashed password | Varchar (128) | Arbitrarily  long [6] | <algorithm>$  <iterations>$  <salt>$<hash> | Not Blank |
| last\_login | Stores the last time a user logged in | Date/Time | 20 digits  YY-MM-DD  HH:MM:SS.MS | 2020-01-02  17:07:30.000000 | Date/Time Format |
| is\_superuser | Designates that this user has all permissions without specifically assigning them | Boolean | 1 char | 1/0 | Not Blank  Boolean |
| username | Stores a username for a user | Varchar (150) | Up to 150 chars | kouroshsimpkins | Not Blank |
| email | Stores a user’s email address | Varchar (254) | Up to 254 chars | test@hfed.net | Must end with @hfed.net |
| first\_name | Stores a user’s first name | Varchar (30) | Up to 30 chars | Kourosh | Optional |
| last\_name | Stores a user’s last name | Varchar (150) | Up to 150 chars | Simpkins | Optional |
| is\_staff | Designates whether a user can access the admin site [7] | Boolean | 1 char | 1/0 | Not Blank  Boolean |
| is\_active | Designates whether or not a user is active on the site [8] | Boolean | 1 char | 1/0 | Not Blank  Boolean |
| date\_joined | Designates when the user account was created | Date/Time | 20 digits  YY-MM-DD  HH:MM:SS.MS | 2020-01-02  17:07:30.000000 | Date/Time Format |
| **id** | A user’s id in the database | Integer | Scales with database size | 12 | Not Blank (Primary Key) |

users\_profile table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
| **Id** | Stores the image id | Integer | Scales with size of database | 7 | Not Blank  (Primary Key) |
| image | Designates the location of the image on the disk [9] | Varchar (100) | Up to 100 chars | Default.jpg [10] | Not Blank or assigned Default.jpg |
| user\_id | One to one relationship with auth\_user table, taking its primary key | Integer | Scales with size of database | 12 (same as for id in previous table) | Not Blank (Should be the same as a key in previous table) |

image\_post table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Field Purpose | Field Type | Field Size | Example Data | Validation |
| **Id** | Stores the post id | Integer | Scales with size of database | 7 | Not Blank  (Primary Key) |
| title | Stores the title of the image given by the uploader | Varchar (100) | Up to 100 chars | Image 1 | Optional  (defaults to image name on disk) |
| date\_posted | Stores the date that the image was uploaded | Date/Time | 20 digits  YY-MM-DD  HH:MM:SS.MS | 2020-01-02  17:07:30.000000 | Date/Time  Format |
| author\_id | Foreign key from auth\_user table | Integer | Scales with size of database | 12 (same as for id in previous table) | Not Blank  (Should be the same as a key in the previous table) |

Get\_Perspective\_Transform.py variables (module):

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Field Type | Start Value | Description |
| rect | Array | ([0, 0, 0, 0]  [0, 0, 0, 0]) | Initialise a list of points, for taking coordinates from NumPy |
| s | Integer | pts.sum(axis=1) [11] | Finds the sum along the columns of rect |
| diff | Integer | np.diff(points, axis=1) [11] | Used to compute the difference between the points |

The variables in Get\_Perspective\_Transform.py are also studied later in the design section of the write up.

Data Flow Diagrams:

Data will flow into and out of the system via this diagram (I haven’t gotten around to it yet, will probably draw it by hand and the scan it in, as Catalina doesn’t support 32-bit applications and all Diagram software is 32-bit)

Objectives:

* The user must be able to register and login to an online system where they will be able to access the application. They may only access the application if they have been logged in.
* The user must be able to upload an image which will be operated on to attempt to make the image more legible using image processing system, that will make the document the main focus of the image (see fig. 1 above)
* The system should give the user the option to control what happens to the output file, so that they can decide whether to save it to the server, save it to OneDrive or save the processed image locally. [12] [13]
* The system must be able to accurately identify the edges of a piece of paper, or the document that a user wishes to scan into the image processor.
* There may be functionality to add a process that converts the image to a ‘.txt’ file, however this lies outside of the core functionality of the program and would have to be addressed as an extension to the project.

Potential Solutions:

There are several potential solutions to the problem at hand, several of which require knowledge outside of Python. For example, the application could be run as a local application, but it would have to be written specifically for each device, or at worst case ported over to each using emulation. Due to this fact, the simplest solution to implement is a web application embedded within a website, as doing this allows the solution to be multiplatform without having to rewrite code. However, there are downsides to running the solution over a network, such as the speed at which a response can be returned to a user. There are also limitations with running certain python libraries over a network, as they are optimised for CPU usage whilst running locally, rather than usage on a server.

Both local and network solutions use the same source code to perform image recognition, and as such could both be utilised to allow for a more rounded user experience overall.

Selected Solution:

The solution that I have selected to use is the network solution, as this will allow for a wider variety of users to access the system. It will also allow me to attempt to use more complex sections of computer science including databases and hardware selection. It will also allow me to use a web framework like Django, which should allow me to prototype the solution in a timely manner, as well as having the ability to use parts of Django such as the admin page, where I will be able to access any databases of users and allow the end-user / sysadmin to control the website without the necessary knowledge of Django and python.

# Design:

Overall System Design:

Due to the fact that I will be using a web based solution, the overall system design is not drastically complex, however it does utilise several solutions rooted within OOP, which is beneficial as it allows me to focus more on ensuring the user has a good experience when accessing the website to use the solution. The system design is outlined below:

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Processes | Storage | Outputs |
| User Details  Image to be Processed | Forgotten Password  Image Processing/Scanning  Adding New Users | Databases:  Users  Uploaded Images  Local Storage:  Images Users wish to share  OneDrive Storage: Images users don’t wish to share | Images shared by other users  User’s processed image  Link to processed image (if stored in local) |

Modular Design:

The overall system will need to be modular by design due to the fact that it is a website that will serve information to the end user. It will give a user the option to access different portions of the site without having to type a new URL into the address bar, and will follow the below hierarchy [14] :

* UserLand
  + Menu Bar (Signed Out)
    - Sign In
    - Sign Up
    - View Previously Uploaded Images
  + Menu Bar (Signed In)
    - Sign Out
    - View Previously Uploaded Images
    - Upload a New Image
* AdminLand
  + Admin page [15]
  + Menu Bar (Same as for UserLand when signed in, plus a few extra items)
    - Sign Out
    - View Previously Uploaded Images
    - Upload a New Image
    - Access the Admin Page [16]
  + Within a Post
    - Remove Post (This may only be a potential addition to the website, as there is potential for this power to be abused.)

The above is merely a list of information to help visualise the way that the Django website will work with the applications within it. It does not seem complex initially, however rest assured the code is rather more complex than the sum of its parts.

Code Based Modular Design:

A large chunk of my code will come from utilising best practices for Django, mostly due to the amount of Python code that is required to get Django to work effectively. The project will have to be split up into files for the Django project as well, but this all comes from Django best practices and documentation [17] .

The basic unmodified versions of all this code comes from running a terminal command, which creates a new Django application within the directory you have specified as the Django workspace, the majority of my folder structure is also dictated by how Django operates, such that it can work effectively. The fact that Django requires a large number of files also increases the size of the project, even if I do not need all of the documents. It is advantageous to use Django, however, as it allows me to focus on getting the main portion of my project correct, so that the document scanner works effectively. It also allows me to focus on programming portions of my project that will actually be marked, as HTML and CSS are not marked or taken into account, and it allows me to integrate with Bootstrap [18] styles with ease.

Other modules that will be of use will include the Python Image Library (PIL) or Pillow [19], which is used to allow python to perform operations on images more effectively. As well as NumPy, which is required for the use of multi-dimensional arrays (which are necessary for Machine Learning!), as well as Linear Algebra and some of the transforms that can be performed with PIL [20] . The final module that I will list here as it is of extreme importance to the backbone of my project is OpenCV (or Open source Computer Vision), which mainly focuses on real-time computer vision. This is required for actually finding the document that a user would want to scan into the website, as it is required to find the borders where a piece of paper ends and a table/the general background begins. The process where we go from finding an initial document and the transformation to become the reprocessed image utilises all of these modules, as well as a few others that are required for any kind of Computer Vision or returning an image to a user.

Any other design choices:

I have decided to implement a sort of Regex within Django, which ensures that a user is only a member of the Harris federation, by checking to see if their email address ends with ‘@hfed.net’, as below:

*from* django.core.exceptions *import* ValidationError

def ValidateSchoolEmail(value):

*if* "@hfed.net" not in value:

*raise* ValidationError("A valid Harris email address must be entered in")

*else*:

*return* value

This validation checker is then run through the forms module as specified by Django, to create a user sign up form that allows a user to sign up to the website. There is a distinct possibility that there are more email addresses that are used within the Harris Federation, however if they do exist I haven’t encountered them or been made aware of their existence by other users.

A screen shot of a smart phone

Description automatically generated

Any documents uploaded to the website will be stored in the media folder, so which will store user profile pictures (As well as the default image for a user profile) and (at a later date to this being written) also uploaded images a user wishes to save to the server rather than to their OneDrive account or to any local storage.