LetterBuddy

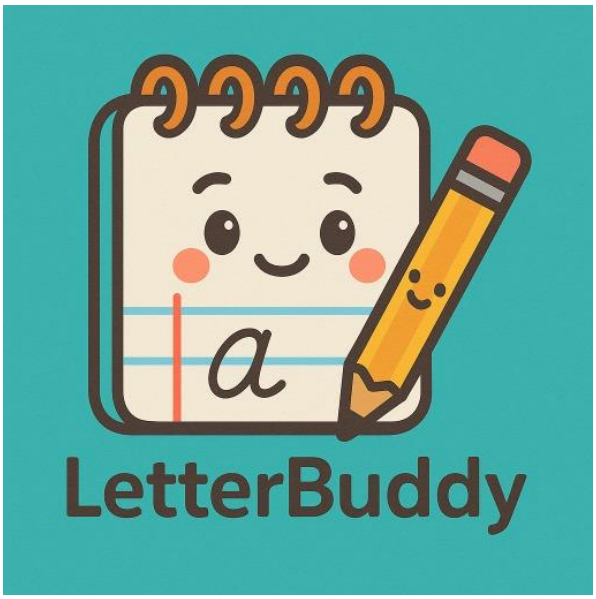
A personal AI assistant
to help kids improve their handwriting

Final Report

Submitted by: Ofir Beck, Erez Lavi

Afeka College Of Engineering

Instructed by: Dr. Sharon Yalov-Handzel



Abstract

The handwriting learning process has always been one of the first most important steps we had to conquer in school as kids. This simple yet critical path proved to be challenging for many kids to get right, as it required a mixture of fine motor, focus, and memory skills. Therefore, it requires frequent practice and constant feedback, which can be challenging to provide in schools today, due to a growing number of students in each class, making it harder for teachers to provide the necessary attention for each student.

Our project aims to develop an effective handwriting practice platform that provides AI-based feedback to the children, guiding them in their handwriting and making this learning path interactive, independent, effective, and easier than ever before.
The adults who are guiding the child, such as teachers and parents, can follow the progress of each of their children and view more precise feedback, allowing them to easily identify strong and weak points in their handwriting.

Table of Contents:

Introduction:

LetterBuddy is a web application suitable for use via both desktop and mobile for children's handwriting exercises.

The exercises it provides are separated into three levels of increasing difficulty:
letters in which the child is requested to write a specific letter several times
words in which he is required to write a specific word
category in which he has to write a word that he thinks of in a specific category.

The child receives minimal feedback, indicating strengths and weaknesses in his writing. This feedback is based on an ensemble of models - both OCR and VLM - that, combined with an algorithm, grade each letter submitted in its exercise based on how confident they are about each letter recognized. This is then compared to the expected letters to grade the exercise as a whole. Letters that are confused with each other are presented to the child, along with letters that were recognized with a high confidence score.
Movement between the different exercise levels is versatile and based on the recent scores of the child in his current level.

An adult guiding the child in their writing learning process (such as a parent or teacher) can review their progress, including each of their submissions, their scores, and a detailed textual VLM analysis of the handwriting's quality. In addition, the adult can view graphs based on the child's performance, including: average scores per level, per day, per letter, and often-confused letters, providing a wide range of statistics to better understand the child's weak sides.

Our goal is making the handwriting learning process easier than ever before, allowing the child to exercise with themselves, mimicking the exercising process in class with direct and harmless feedback based on AI, while allowing the adults guiding them to focus on the real matters - identifying weak points in their writings through statistics and thus allowing to give each one of them to-the-point personal feedback and help.

Background and related work:

Competitors Review:

We reviewed three applications that are currently available, each designed to simplify the handwriting exercise process in its unique way.

The first app we reviewed is “**Dynamilis Handwriting Practice,”** which offers fun and personalized activities for kids to improve their handwriting skills, using AI to create a profile for the child, showing their weaknesses and strengths.

The next app we have researched is “**Writing Wizard - Learn Letters,”** which takes on a much simpler approach, guiding the child to write the various characters by tracing with guided animations, with feedback provided based on correct or incorrect tracing.

Finally, we reviewed “**Kaligo”,** an app primarily designed for classroom use. Similarly to the last app, the child is provided with word tracing for him to fill in, and a more open option to write the words on his own, receiving feedback for each character as it. Giving teachers the option to build lessons and track each student's progression.

Our app **fills the gaps** present in those kinds of platforms - they do not reflect the actual environment the children will be expected to write in. None of them teach the child to write on top of mobile platforms like mobile phones and tablets. Thus, they are not allowing the child to fully develop the much-needed skill of handwriting on paper, which may differ from writing on top of a screen. In addition, there is yet such an app to make full use of the power of OCR abilities in order to provide both the child and the adult with flexible and accurate analysis of the writing by the child, to mimic the exercise process in class.

App / Criteria	Dynamilis Handwriting Practice	Writing Wizard - Learn Letters	Kaligo	Our application
Variety	High - various types of games provided.	Low	Medium - teachers can upload lessons that include words to practice	Medium - offer letters, words, or category exercises.
Learning path customization	Personalized activities based on AI assessments.	Adjustable difficulty with tracing, guides, and custom letter or shape practice.	Adults can adjust the student's learning path.	Personalized exercises based on past AI assessments.
Progression tracking	Detailed - including graphs of the child's progress on each writing aspect, by week.	Non	Highly detailed, including replayable videos of the student writing and the scores given	Provide the adults with personalised progression tracking of each child.
Direct feedback (to the child)	Provides a color-coded feedback system - green/yellow/red according to writing quality.	Provides immediate feedback for correct or incorrect tracing.	Correct or incorrect tracing, in free writing - AI feedback per character.	Feedback based on the model's analysis, pointing out similar letters that got misrecognized(like i and l)
Requirements	iPad + stylus.	Any mobile platforms	iPad + stylus.	Mobile platforms or a computer with a camera.
Popularity	approx. 10,000 users	5M+ downloads	100,000+ users according to their site	Currently, there are no users
Cost	Monthly: parents:10$ therapists:2$ schools: 150$ (for classroom)	A free app with in-app purchases, and a school version for $10.	Quotation basis (depending on the user)	free to use.

Literature Review:

Handwriting skills are a critical part of a child's learning path and are often taught between ages 6 and 8, while they are finally automated in ages 10-14. [1] According to one of the most accepted cognitive models, the writing process has 3 separate steps, which often occur simultaneously:

- Planning - goal setting, idea generation
- Translating – production of handwritten text for the generated ideas
- Reviewing – the writer evaluates and revisits their written text.[4]

To enhance and improve this process, studies have recognized key principles:

- Practice – handwriting should be practiced at least 2 times a week for 10 weeks, and be repetitive (like writing the same letter a lot of times)
- Handwriting instruction should be explicit.
- Intrinsic and extrinsic feedback of sufficient quantity and quality should be available. Feedback is a crucial parameter in the learning process.
- Variability should be introduced during handwriting learning (Varying the letter font and size)
- Learning environments should be motivating, supportive, and include self-regulation of performance.

The time dedicated to practicing handwriting skills has decreased over the last 30 years (varies from country and teacher), as well as growing challenges such as a large portion of teachers who don't consider themselves to be properly trained to teach handwriting, and the growing amount of students in each class that makes it difficult to implement individualized instruction, to provide physical guidance for each pupil.

These challenges may be the cause of the growing number of students struggling with writing, which ranges from 6% to 30%. These students face challenges such as poor legibility, slow writing speeds, spelling errors, and excessive pen pressure[1]. These challenges may be caused by struggles with the planning phase of the writing process, which may be due to immature working memory[4].

The main options to evaluate handwriting today are tests that often assess aspects like letter formation (difficult to quantify), writing speed, and ergonomic features. However, these evaluations face limitations, including subjectivity and difficulty identifying the underlying causes of handwriting issues[1].

Students who are identified with poor handwriting can attend writer's workshops to improve their skills. In such workshops, they are given important feedback on their handwriting, revisiting their written text, and showing it in front of their classmates who provide feedback themselves[4]. Another option to improve those skills is rehabilitation, which often involves occupational therapy and is personalized based on the child's challenges and environment[1].

Feedback on handwriting plays a crucial role in the student learning process, providing guidance, academic interaction, and accepting it could lead to increased writing success. Unfortunately, due to the difficult process of accepting feedback, not all students consider it helpful, and that can lead to a variety of emotions, both positive and negative; the latter could invoke negative responses and decrease motivation. Knowing the student's openness beforehand could improve the feedback given, and it is connected to the student's confidence in their writing skills[4].

Article [4] describes a study: 867 students in grades 3-5 across four elementary schools took an online survey in which they were asked: a closed question (yes or no) “Do you like to receive feedback about your writing?”, and an open question in which they explained their first answer. To the closed question, 765 students answered "Yes"(88%) and 102 "No"(12%).

The reasons that were most provided for liking feedback were separated into these groups:

- Mastery – to become a better writer.
- Positive aspects – when the feedback highlights positive aspects of their work.
- Helps with recognizing areas of mistakes in their writing and what to avoid.
- Others could offer good ideas for their future writings and appreciate others' opinions.
- Invokes positive effects, emotions (such as pride), and experiences.
- Gives motivation to keep improving their writing skills.

The reasons students have given for disliking writing feedback:

- Avoidance – students don't care about the feedback, it doesn't matter.
- Negative feedback – expectation for negative feedback, without providing what is good, only good feedback was appreciated by them, which might be turned into positive experiences that may open their minds to accept negative feedback in the future, and improve their beliefs about their handwriting.
- Embracing (the teachers may be mean or disappointed)
- Negative affect may cause negative emotions or experiences.

One way to encourage the feedback process is to have conversations about it with the students, about its use and purpose[4].

AI could help build tools that will take these findings into account and automate such feedback for evaluating and improving handwriting, and offer personalized practice. Such AI could be implemented as a part of a digital learning technology, which is used by people to overcome the social and infrastructural barriers of learning, and is often used as tutors, teaching aids. There are already some tools designed for handwriting exercises that aim to support the acquisition of handwriting for all children, including those with special needs.

However, implementing these AI solutions would require overcoming challenges like ensuring accessibility, maintaining data privacy, and integrating cultural and linguistic

variations in handwriting styles, as well as overcoming the lack of acceptance of new technologies by educational professionals that may prevent their mass use[1].

The accessibility challenge may prevent children from feeling comfortable with using such tools. This requires a clean, intuitive, and easily understandable user interface.

Article [3] dives into possible solutions to this issue. It outlines best practices for creating AI interfaces that prioritize safety, engagement, and trust. Key recommendations include:

- Simplified design- the interface should be easy to navigate with clear buttons, simple visuals, and age-appropriate language.
- Interactive features- animations, games, and interactive elements to engage children and enhance motivation should be included.
- Customization- personalization options like avatars and themes.
- Safety measures- implement content filtering, privacy policies, and parental controls to ensure safe interactions.
- Fairness and diversity- all children should be treated equally, avoiding content biases.

In addition, the AI itself should have these attributes:

- Accuracy - provide reliable information to prevent confusion.
- Consistency - ensure stable, predictable AI behavior.
- Robustness - adapts to various inputs and scenarios.[3]

With these attributes in mind, an OCR model can be designed to detect, analyze, and evaluate handwritten text.

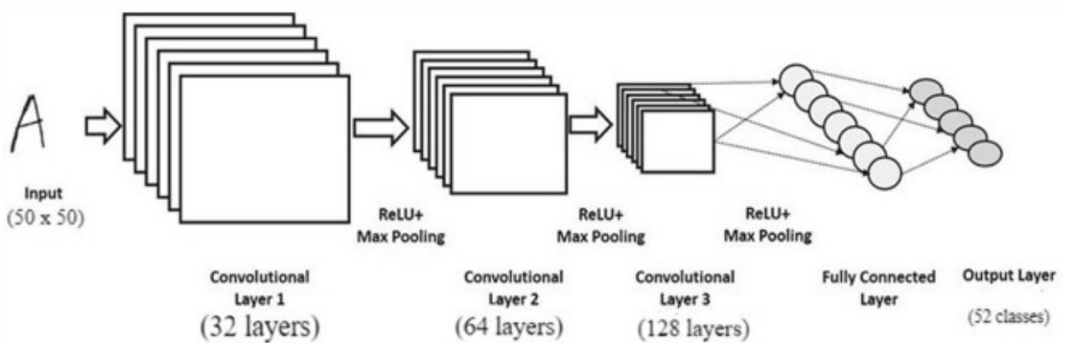
OCR stands for optical character recognition, and there are two approaches for such models:

- The “online” approach involves real-time user input (rather than scanning or using pre-captured images).
- The “offline” approach refers to the process of recognizing text from an image or scanned document that has already been captured and stored on a device.[2]

Articles [2] and [5] each describe possible models to evaluate the handwritten text of these different approaches.

One of the proposed models, using the “online” OCR approach, is a CNN(convolutional neural network) based model. This model architecture includes three CNN layers, one flattened layer, and two dense layers, each followed by a ReLU activation function and max pooling operation. The model demonstrates strong performance, classifying all 52 English character classes with 96% accuracy on both test and validation sets. It was trained using Adam optimizer, which is known for its efficiency in optimizing deep

learning models. The training process consisted of 80 epochs. The handwritten input has been captured in real time through an HTML canvas.[2]



A recent study, conducted in 2024, presents a comparative analysis and integration of 4 'offline' OCR engines - Tesseract OCR, Keras OCR, Paddle OCR, and Microsoft Azure Computer Vision.

The study evaluated the performance of these engines across different image preprocessing techniques, such as Edge Detection, Image Thresholding, and diverse datasets like the IAM and the FUNSD dataset. It aimed to investigate the strengths and weaknesses of each engine in various scenarios, including recognizing different handwriting styles and extracting key information from complex layout forms, even under challenging conditions like poor image resolution.

The evaluation metrics used in the study were accuracy, precision, recall, F1 score, Character Error Rate (CER), Word Error Rate (WER), and running time per image(RT). These metrics offer insights into inclusivity, detail sensitivity, and the trade-offs between speed and accuracy in practical scenarios.

The study proposed an integrated OCR scoring system (which is similar to the ensemble method in ML). The idea is to process a text image simultaneously by the four OCR engines, and then each one of them will produce outputs of detected words and their according confidence scores. If an OCR engine consistently performs well on certain types of texts, more weight is given to its output in the scoring process. The final output is a string of words that have been selected based on the highest score from the aggregated results, aiming to form the most accurate rendition of the text from the image.

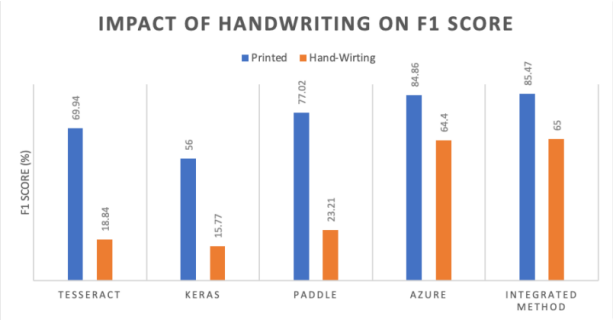
The main findings of that study are:

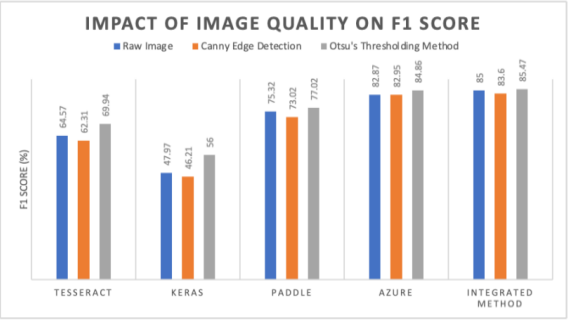
- All OCR engines tend to perform significantly less effectively on handwritten text compared to printed text, as they have a lower F1 score and longer running time per image. This disparity is primarily because OCR works by pattern recognition, which is easier for printed text recognition. However, human handwriting typically lacks consistent patterns. Moreover, handwriting is usually more

diverse and often features variations in character shapes, sizes, and spacings, which challenge the pattern recognition capabilities of general OCR technologies.

- Azure OCR stands out in handwriting recognition compared to other OCR models. It might be due to advanced machine-learning algorithms trained on a wide range of both handwriting and printed samples, and the use of Intelligent Character Recognition (ICR) techniques.
- Otsu's thresholding image processing method positively impacts performance metrics across all OCR engines.

However, the result of comparative analysis may not completely reveal or indicate the superiority of one OCR engine to another. Due to the limitation of the project scope, no fine-tuning of weights or any customizations were done to the OCR engines selected. It has been proven that certain OCRs, especially open-source engines, benefit a lot from task-specific learning and fine-tuning by adopting a pre-trained weight while training and evaluating on a prepared dataset[5].





System Design:

Functional Requirements:

1. User Registration:
    - Adults can register by providing their name, email, username, and password.
    - Adults can register their children by providing their name, username, and password. Once they register them, they are assigned to the adult who added them, and they are able to view their progress.

2. User Authentication:
    - Both children and adults can log in to their accounts using their username and password.

3. User Authorization(roles and permissions):
    - Children can access handwriting exercises and GIFs of how to write letters correctly. They are forbidden from viewing their progression tracking, the content about correct handwriting, or managing other accounts.
    - Although the adults don't have access to handwriting exercises, they are able to access detailed progress tracking, including graphs and each of the children's submissions, content about correct handwriting, and management of their children's accounts.

4. Handwriting Guidance:
    - During exercises, animated GIFs are available for each letter in the child's language to guide proper handwriting.
    - Provides content about correct handwriting for adults to help them guide the children.

5. Handwriting Exercises:
    - The system generates exercises based on the child's past performance and their current level, as determined by the OCR and VLM evaluations.
    - The children are prompted with 3 types of exercises based on their current level: letters, words, and category-based (e.g., "Write a word from the category animal") to write.

6. Handwriting recognition:
    - Children can upload or capture their handwriting (on a blank page) using the device's camera or gallery.
    - The system uses a combination of an OCR and VLM model to detect and evaluate handwriting directly from those images.

7. AI-based submission evaluation:
    - The OCR model gives a confidence score for each character detected in the child's handwriting, and the VLM provides its reading of the image. Combined, an evaluation of the submitted image is created, scoring each of the exercise's letters and the exercise as a whole.

    - Provide the child with simple and mostly positive feedback on their handwriting based on the evaluation. Focusing on what the child got right rather than what they got wrong, to encourage them to continue exercising.

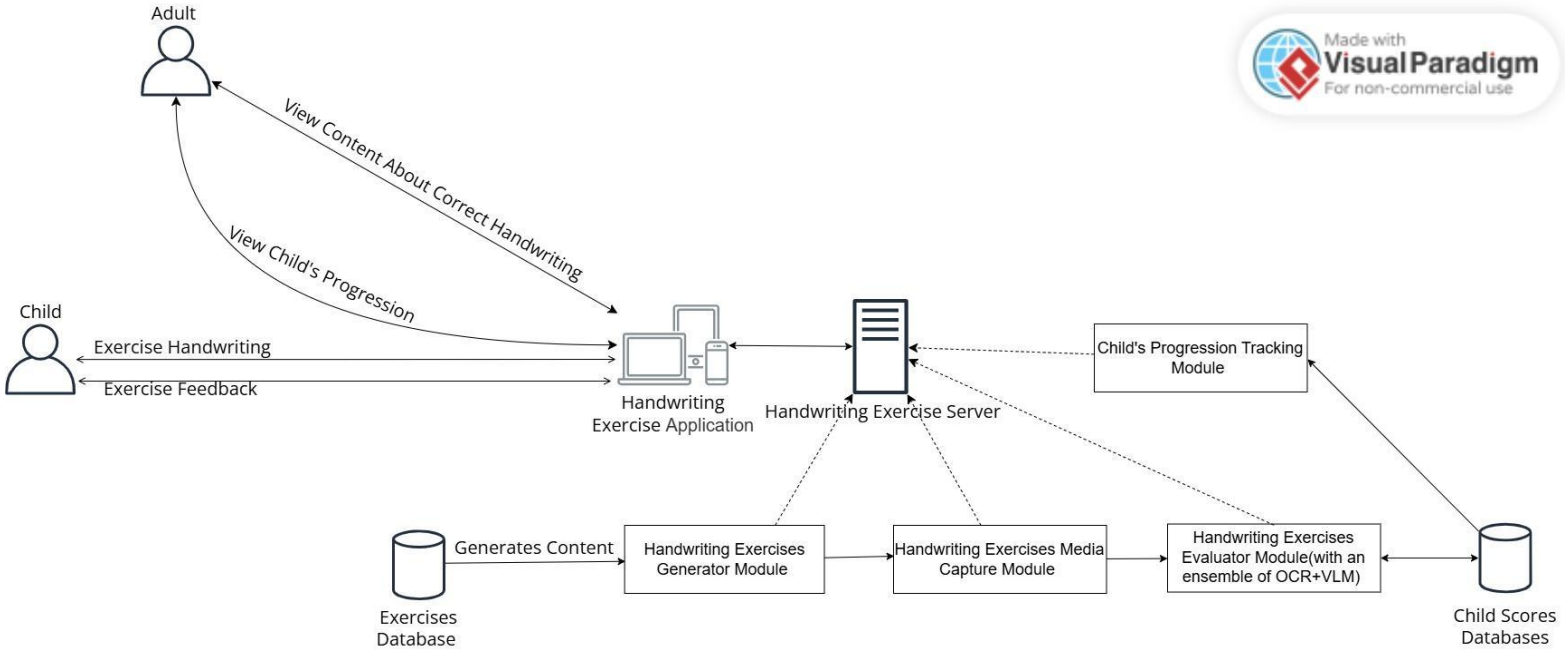
8. Data stored in the database:
    - The users of the system are saved in it, each with their own role. For the child, we are also saving the ID of the parent who is guiding him.
    - After a handwriting exercise has been practiced, we save in the database: the practiced word, the expected one(if there is one), its category, and its score. In addition, we save each letter submitted in the exercise, along with the expected letter and its confidence score. To be able to detect easily confused letters and provide statistics for each letter.

9. Progression tracking:
    - Adults guiding each child can view their progression, displaying the information saved in the database in several ways:
        - Each child's submissions table - to view each one of his submissions, with an option to view the score given for the exercise, the score for each letter, the submitted image, and a textual analysis of the handwriting quality by the VLM - in a similar form that teachers and professionals often provide.
        - Visual representation of the child's progression in the form of graphs, including: average score for each letter in their alphabet, which allows spotting difficulties the child encounters with specific letters, average score in each level, daily average score showing the progression of the child over time, and letters that the child often confuses between so the adult could work on them with him.

Non-Functional Requirements:

1. Security:
    - The user's authentication process is done under a secure protocol.
    - The user's information is saved in a secure database.
2. Performance:
    - The content the child is requested to write will be generated in 3 seconds or less, at least 95% of the time.
    - The handwritten capture submitted by the child will be analyzed in 10 seconds or less, at least 90% of the time.
3. Portability:
    - Support popular browsers such as Chrome, Safari, and Edge
    - Responsive design for Android, iOS, and PC resolutions.
4. Accessibility:
    - Clear, understandable UI with large buttons and minimal text, thus easy to learn and use by children ages 5-12.
    - Ensure adults can easily access and interpret progress tracking data.
5. Safety:
    - The word generator won't generate any harmful or inappropriate words that the children are requested to write.

System Architecture:



Technology stack:

Frontend Technologies:

- **React.js** - a very popular front-end development framework.
- React libraries such as **react-router**, **zustand**(for state management), **axios**(for api requests), and recharts for interactive charts.
- **TypeScript**, which is a programming language very similar to JavaScript (the more traditional language for web), but the main difference is its static typing feature. We chose this language because it can catch more errors at compile time, which helps create more stable apps.
- **Vitest** for automated unit-testing and the **Cypress** library for automated E2E tests.

Backend Technologies:

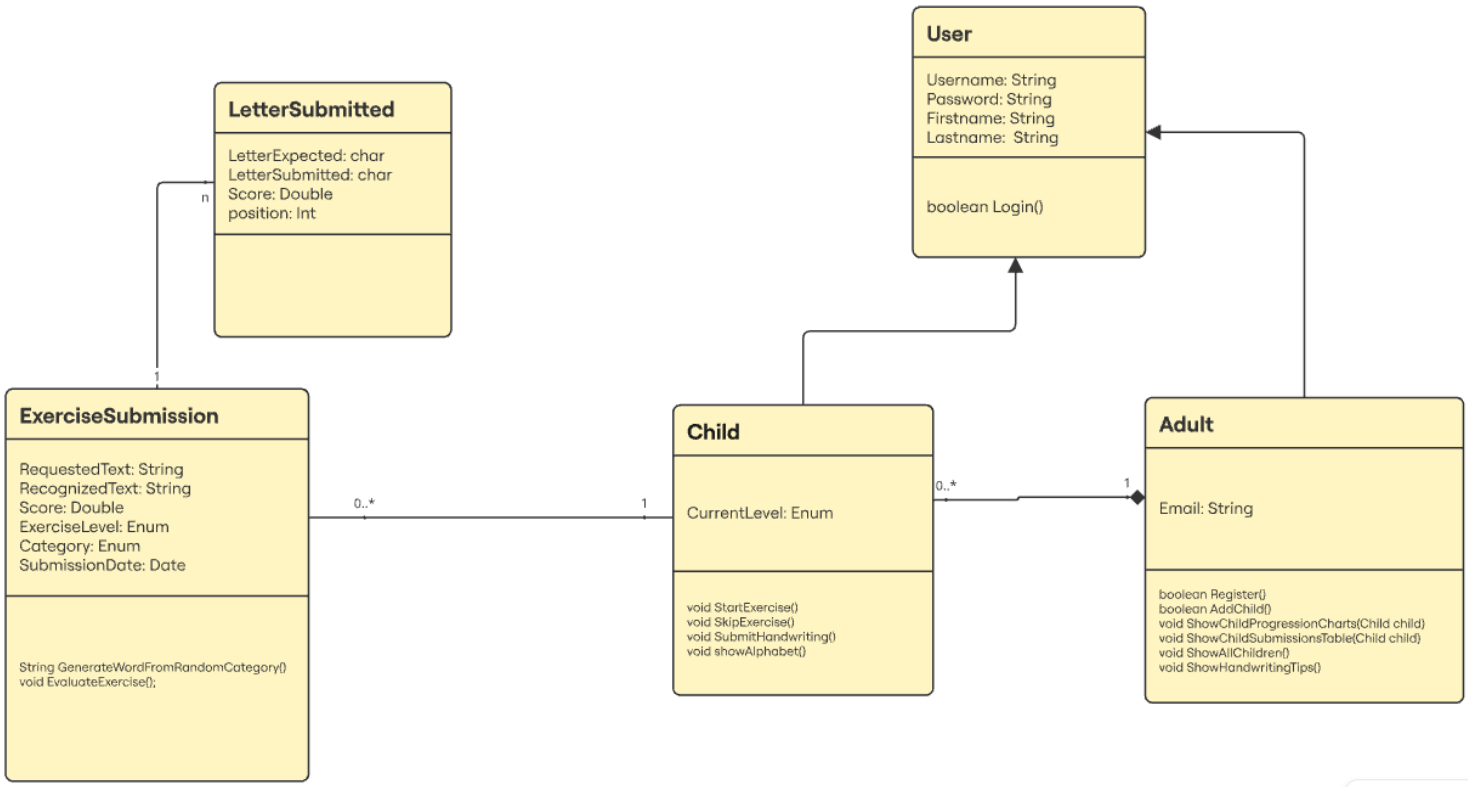
- Using **Python Version 3.9.10**
- **Django Rest Framework(DRF)** as our backend framework, which includes an ORM that maps SQL's entities into Python objects and handles the SQL queries itself.
- **Django's simplejwt app** to easily handle users' authentication using JWT access and refresh tokens.
- **Django's permissions** handle correct authorization to each of the backend's endpoints.
- **Django's APITestCase** for automated unit testing for the backend's endpoints.

External API used:

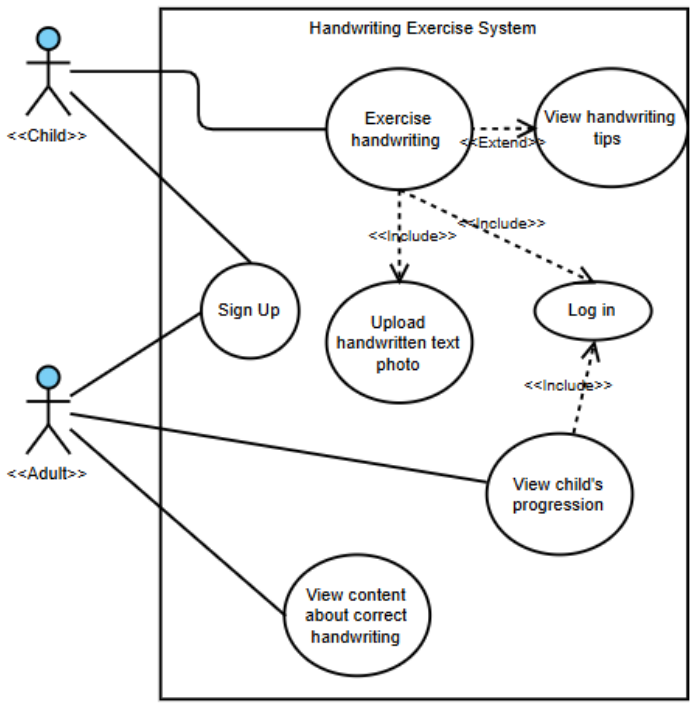
- Open AI **GPT 4.1 mini** is a VLM (vision language model) used via the **GitHub Models API**.
- **Cloudinary**: an end-to-end image and video management solution for websites and mobile apps. We used it to store the child's submission images, so when the adult users want to view their child's submission image, they would retrieve it from there.
- **GroqCloud**: A service that provides fast and free AI model inference API. We used it to run the **llama 4** VLM model, which we found slightly less powerful for our needs. Therefore, it serves as a fallback for the GPT 4.1 mini model (in cases we have reached our GitHub models API limit or other possible problems).

UML Diagrams:

Data Design:



Use-case diagram:



Methodology:

Design choices:

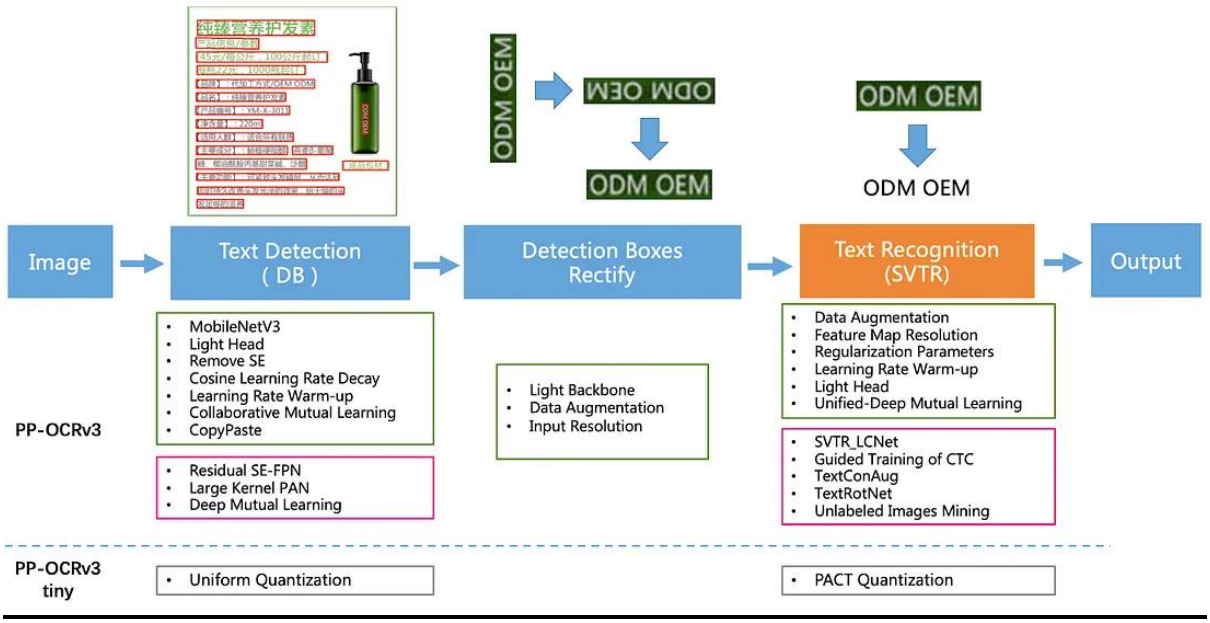
- Frontend framework - We used the React.js framework and CSS modules in order to create a responsive UI for both desktop and mobile devices, while keeping a fast and maintainable system with clear separation of concerns.
- Backend framework - DRF (Django Rest Framework) for the Backend to be able to create a scalable and flexible API, with minimal DB interactions as possible, thanks to its ORM.
- PostgreSQL DB for relational entity relationships, easily integrated with DRF's ORM. For the exercise submissions, we stored the image URL, while the images are stored in Cloudinary.
- An ensemble of AI models - PaddleOCR, together with a VLM (GPT 4.1 mini and llama 4) for creating an accurate submission evaluation.
- Custom words DB - To have complete control of the words generated for the exercises, we stored a handful of child-friendly beginner vocabulary words in our DB.

Algorithms and models used:

1. **PaddleOCR's detection and recognition:**

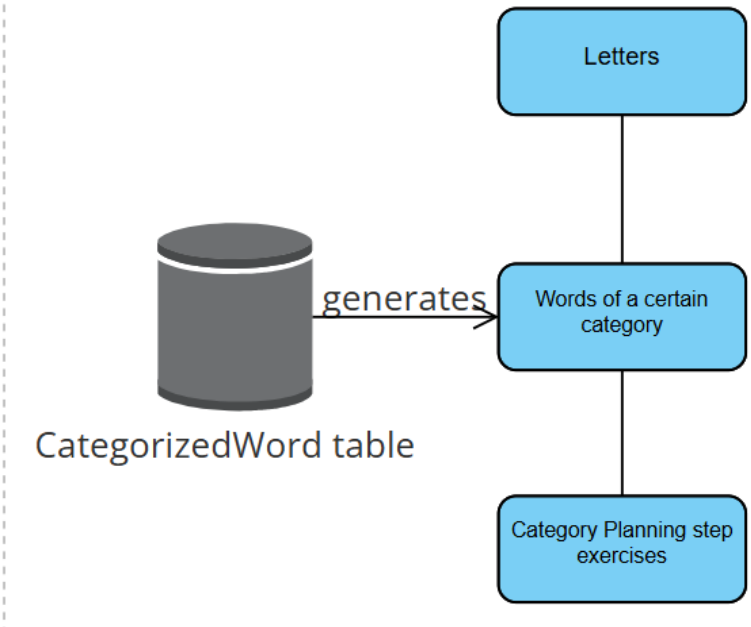
   PaddleOCR receives the handwritten image and processes it through 3 separate steps, where the goal is to recognize the text from the photo.

   A. The text detection step involves preprocessing the image to fit the model requirements and segmenting it into separate "boxes" of text.
   B. The rectification step for the detection boxes involves rotating each box to align with the correct reading direction in the selected language.
   C. The text recognition step: The trained recognition model interprets the written content of each detected box, makes predictions, and determines the confidence score of each box prediction. We have adjusted the open-source library to get the model's confidence scores per character, allowing us to give more detailed feedback to both the child and the adult on the handwriting.



2. Adaptive handwriting exercise generation

When tracking the learner's data, we can offer exercises that fit their current level(letters, words, or categories). While allowing dynamic movement between the different levels, based on their recent performance in their current level.
The exercises in each level are generated in the following manner:
Letters: The child is requested to write a random upper or lower case letter a couple of times (for instance - 'please write the letter 'a'7 times').
Words: We have created a table in our database to store hundreds of child-friendly words separated by their categories. Once the child requests such an exercise, we randomly fetch a word from that table for them to exercise.
Category: The child is provided with a child-friendly category(for example, "Name an animal"). According to our literature review, these exercises should help the child with the planning step of the handwriting process, which is the hardest step for those who struggle with handwriting.



3. Exercise scoring based on a VLM and an OCR ensemble:

Since PaddleOCR's results on handwritten text may vary and have to be clear for it to understand all the written better, we have decided to score the submitted by the child by an ensemble of models - both the OCR and the VLM - which often grants a more reliable read, with the downside of it not being able to provide the exact confidence score of his read.

In both the letter and word exercises, we expect the child to write exactly what we have required them to write. We request both models to read what is written, and then using Python's SequenceMatcher with the requested text we receive for each requested letter the letter that was submitted in its place, using SequenceMatcher we can handle situations like "vhello", "vello", that can be submitted instead of "hello", and still be able to identify expected and missing parts successfully as long as they come in the order of the original word(unlike going over the expected and submitted letter by letter - where a slight hist was amounted to a score of 0).

After SequenceMatcher provides us with the submitted letter for each of the expected - we can go over each expected letter and grade it according to each model's score(assuming the VLM is always 100% sure in his read, since he doesn't provide confidence scores), thus we grade each expected letter and save it in the database together with the submitted one, while giving the VLM 70% of the score weight and 30% to the PaddleOCR.

The overall score of the exercise is granted based on both the average of each expected letter confidence score(while mismatched letters get a confidence score of 0, and letters that are often confused with each other get half the credit) and the maximum Levenshtein distance between the two models' guess - punishing redundant and replaced letters and their different order then expected.

For exercises of the level category in which the child is requested to write a word from a specific category, we first let the VLM decide if the written word could be close to a word from that category(same or misspelled). Then we use the same algorithm to score the exercise as we did for the previous levels, when what we expect to find in the written is the closest word from the category that the VLM mentioned.

Implementation Steps:

During the whole implementation process, we kept an “agile” approach, both in our actions and our mindset. We have implemented features in an agile way of working, meaning we have created the features in sprints within a reasonable time limit, while continuously communicating with each other. In addition, we reacted quickly to technological changes, which allowed us to flexibly change features according to those changes, such as the addition of a VLM model in the submission evaluation, a model that wasn’t accessible at the beginning of our project to freely use.
We believe this approach helped us to create a much better final product.

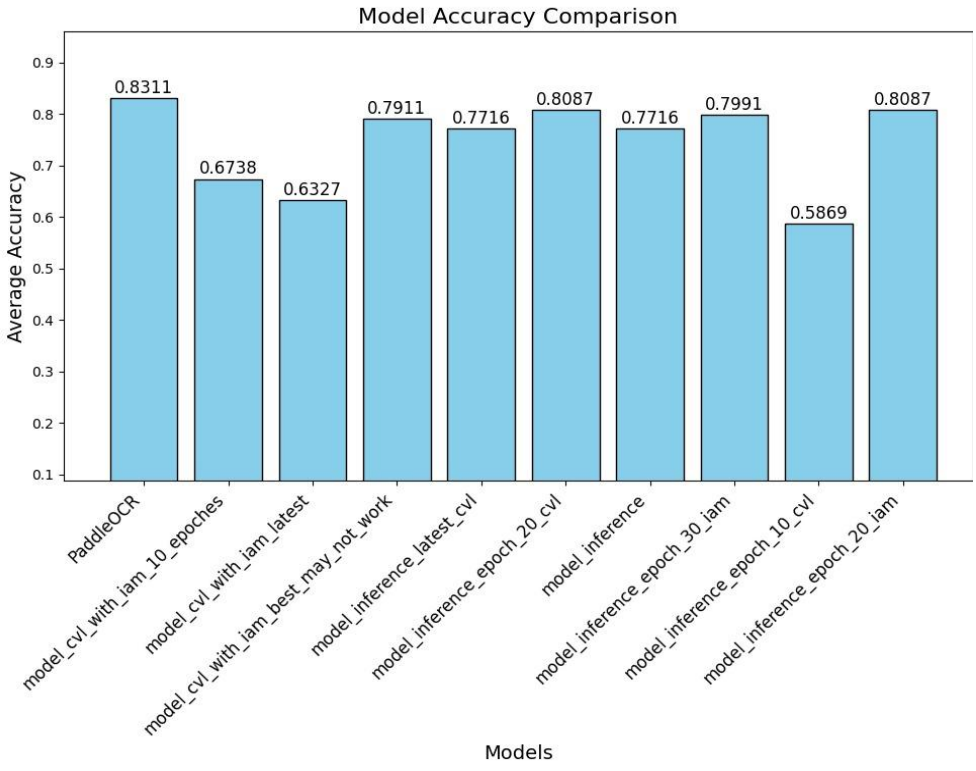
Methodology challenges and solutions:

Submission Evaluation Method:

During our project, we experimented with different ways to evaluate the exercises submitted by the child user.

Initially, we attempted to evaluate various OCR and HTR models. Based on both our research and the findings from the article reviewed in the literature, we found that PaddleOCR achieved the best results on handwritten text. Though after analyzing its results on expected input, they still weren’t to our satisfaction.

After that, we have tried fine-tuning PaddleOCR on different datasets of handwritten images, in the hope that it will achieve our desired results. We evaluated each of these models against each other while using the Levenshtein distance from the expected output. Unfortunately, as the comparison below shows, we have found that the original fine-tuned model(PaddleOCR) still had the best results that weren’t satisfying our needs, and thus, we have decided to search for a different way to improve our algorithm.



After acquiring access to a couple of VLMs, we were impressed with their results, but didn’t want to completely redundant the use of PaddleOCR, which gives us a confidence score on each letter that we consider important for this project.

Thus, we decided to make an ensemble of both of those models, a VLM together with PaddleOCR. To make the most out of this combination, we crafted an algorithm to combine both evaluations.

PaddleOCR's inability to give us a confidence score per letter

To give informative progression tracking to the adult about their child’s handwriting, we wanted to score not only the exercises of the child, but also each letter they wrote, and thus be able to point out weaknesses and letters they often confuse. After reviewing all the main OCR models available on the market, we found that they don't give feedback per written letter but only per sentence or word. In order to solve this challenge, we had to dive into the PaddleOCR open-source code and make some small logic changes in it in order to extract this info.

Non-deterministic VLM handwriting feedback

Although we were generally satisfied with the VLM model's ability to give feedback on handwriting features such as inconsistent size and spacing, we still had challenges with making the optimal prompt that would fit our use case the best.
To solve that, we made a custom prompt for each level, limited the number of tokens per output feedback, and did some testing with trial and error for validation. We did this until the model returned a satisfying feedback for all levels.

Implementation:

After working on the system design and our UI prototype, our next important step was implementing our system. ‘LetterBuddy’ was built as a web application with a clear separation of concerns between frontend and backend.
During our implementation, we prioritized modularity between the different components with emphasis on reusability and testability.
In addition, we used the TDD methodology by writing automated tests during development and using them as regression tests after implementing new features or solving bugs in the system.

Software Components and Key Modules:

Frontend

Good prototype planning has let us write our frontend in a more reusable way, which means creating our reusable custom components. In addition, we used CSS modules for the design, which means separating design and logic into different files - that is considered a good practice for high maintainability and scalability.
Our frontend system architecture is built in a best-practice structure and clean separation of concerns, split into a few main modules:

- **Components**
  Auth: Includes functionality and design for both child and adult login and signup authentication.
  Adult: Holds all the adult’s related features- child’s list, adding new child, submissions table, feedback view, progress data graphs, and useful articles.
  Child: Holds all the child’s exercise components functionality and design- getting new exercises, submitting handwriting images, getting basic feedback, and some guidance with letters GIFs.
  UI: Includes all the reusable custom components - buttons, labels, input forms, modal, and camera.

- **Pages**: Represents the app's separation into different pages that the user can navigate between. Each page holds all the components related to this page means it is higher on the component tree hierarchy.

- **State Management store**: Used to manage all the app’s internal states. Almost any change happening on the app, including data flows from the backend or user interaction, is managed from here.

- **Tests**: Includes both unit and E2E automated tests of the app - we tested the crucial app's flows: login/signup, child exercise, and adult view child submissions.

Backend

Our DRF's backend is separated into 2 different apps, accounts and exercises - each of them handles a different aspect. Both of these apps have a similar structure, consisting of:

- **Views** - in which the main logic for our API's endpoints is present, including a function for each one of them.

- **Models** - that contain the app's models - an object representation that is mapped to an SQL database, using this ORM, we are able to interact with those tables without SQL queries, within the Python code.

- **Serializers** - that "convert" the JSON data input to the API to a Django model(or the other way around) for the views using them.

- **URLs** - map the views created to a specific URL of the API.

- **Apps** - contain configs - that are steps taken before each app is fully loaded, allowing for one-time initializing activities.

- **Tests** - contain API tests using Django's BaseTestCase, which allow building test cases that assert the API's responses based on specific requests. Those tests often use a temporary database that is set up before them, so order not to interfere with the original database.

The accounts app mainly handles the users' models - both the children and the adults - their registration, log-in, log-out, and the adult's children list view.

The exercises app mainly handles everything surrounding them - generating them for a child, submitting them, and viewing the child's submissions, and stats based on those exercises.

Implementation Challenges and Solutions:

Conflicts during development

Regarding our team collaboration, we worked on the project both together (pair programming on Microsoft Teams) and by developing separately using Git as our version control. We used the 'GitHub flow' branching strategy as our Git workflow, which means each new feature/ bug fix/ refactoring is implemented on a new branch. After finishing the work, opening a pull request into the main branch so each one reviews the other's code, and then deciding to merge it into our main branch. That workflow enabled us to work effectively, avoiding code conflicts and problems during the implementation.

Making the exercise camera accessible from both mobile and desktop

One of the key elements of our system that separates us from our competitors is its accessibility - no need for expensive iPads and pens - just internet access, a computer or mobile phone, a pen, and paper. That's why it was important for us to let our users capture their handwriting both by mobile and computer camera.
To achieve that, we used a library called react-camera to create our own custom and reliable component, letting users exercise from any device with a camera.

Generating a reliable and relevant child's exercise content

We tried different public datasets and corpora, including the 'WordNet' corpus, to generate simple and child-friendly categorised words to exercise, but have found the words there too complex for young writers. Because of that, we decided that the best solution would be to create our own local categorised words DB, which is based on a vocabulary that fits exactly our child users' age range. That gave us the most reliable and accurate result for our needs.

Experiments and Results:

During the project, our main goal was to create the most accessible and reliable AI-based handwriting exercise application. Besides making a well-designed and accessible interface that is suitable for both children and adults, there are some more elements that make our system more reliable for our specific use case of giving handwriting feedback, which differs from using other, more generic tools.

Ensemble of models

We decided to make an ensemble of models by adding a VLM evaluation together with PaddleOCR, resulting in a more balanced feedback system. Each of the two models has its unique strength in different cases. Sometimes one of the models can identify better than the second, which makes our score more balanced and reliable overall. Without the VLM, making sure a word submitted at the category level indeed matched the requested one would prove much difficult.

Our SequenceMatcher algorithm:

We decided to integrate SequenceMatcher to better fit situations like undetected letters or ones that arrange differently than expected, to prevent falling the child on an exercise, for example: a submission of “vhello” instead of the expected “hello” in our original algorithm would result with a 0 score, while still fitting almost perfectly to the expected, while in our new one will grant a rather higher score as it should. Levenshtein distance also contributes to the exercise’s final score to punish redundant and misordered letters. We have also created a map of often confused letters - when one is confused from there, it contributes half its score(“s” vs “S” for example).

Sample Outputs:

We have conducted a few sample outputs to convey the need for both the ensemble and our scoring algorithm.

B	C	D	E	F	G	H
exercise	vlm_read	paddle_ocr_read	combined_text_after_ensemble	combined_score	levenshtein	final_score
ballon	bnloh	bloh	bnlloh	0.655	0.6	0.627
dog	dob	dob	dob	0.638	0.667	0.652
hat	hat	nat	hat	0.894	1	0.947
hello	vhelob	Yheo	heloo	0.715	0.727	0.721
red	red	ed	red	0.859	1	0.858
rocket	kOcket		kOcket	0.491	0.667	0.579
steamroller	Steamroller	steomler	steamroller	0.775	0.909	0.842

We can see the exercise the child was requested to write, and what each model read - in all of the cases here, the **VLM read exactly what we wrote in each image**, while PaddleOCR found trouble with some words - with the image that has kOcket in it, he could not recognize anything, thus wasn’t suitable only on itself for this task.

When running SequenceMatcher on each of the model’s guesses, we acquire for each requested letter the letter that came in its place, in the order expected. Then,

going over these matches, we decide about the final guess for each letter(when either one of them succeeds in their guess of the expected it is taken, otherwise the majority). From this process, we acquire the combined text, which is the ensemble guess for the word, and the avg_score, which is built from the average of the confidence scores of the correctly read letters.

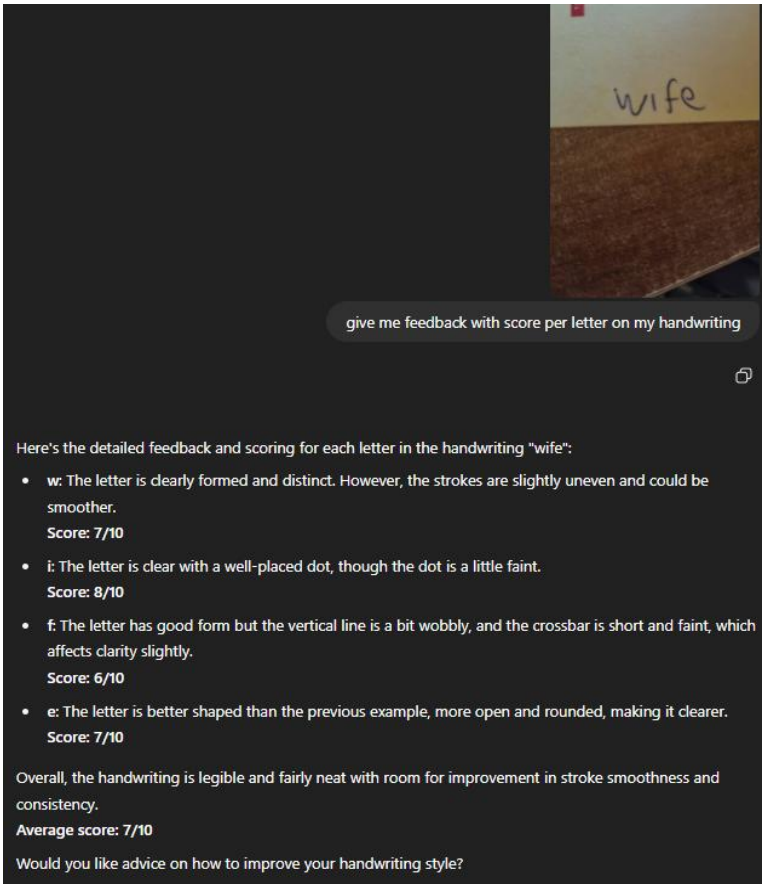
Without SequenceMatcher, trying to score exercises where the order mismatched the requested(like the “hello” exercise above, in which the child wrote “vhelob” instead) would amount to a score close to 0(since trying to going over the requested and the expected letter by letter won’t work since the sligh hist at the start of the word. This method allowed us to avoid common situations of extra letters that were read by the models by mistake(or were written by the child), or a picture that included letters or words from previous exercises that may cause the same scenario.

Different custom prompts

We have also made a unique prompt for each exercise’s level, that prompt is based on articles and papers on handwriting evaluation and meant to guide the model to focus on very specific handwriting elements, such as letter formation, spacing, size, and line quality. As a result, our feedback is more relevant and aligned with professional metrics.

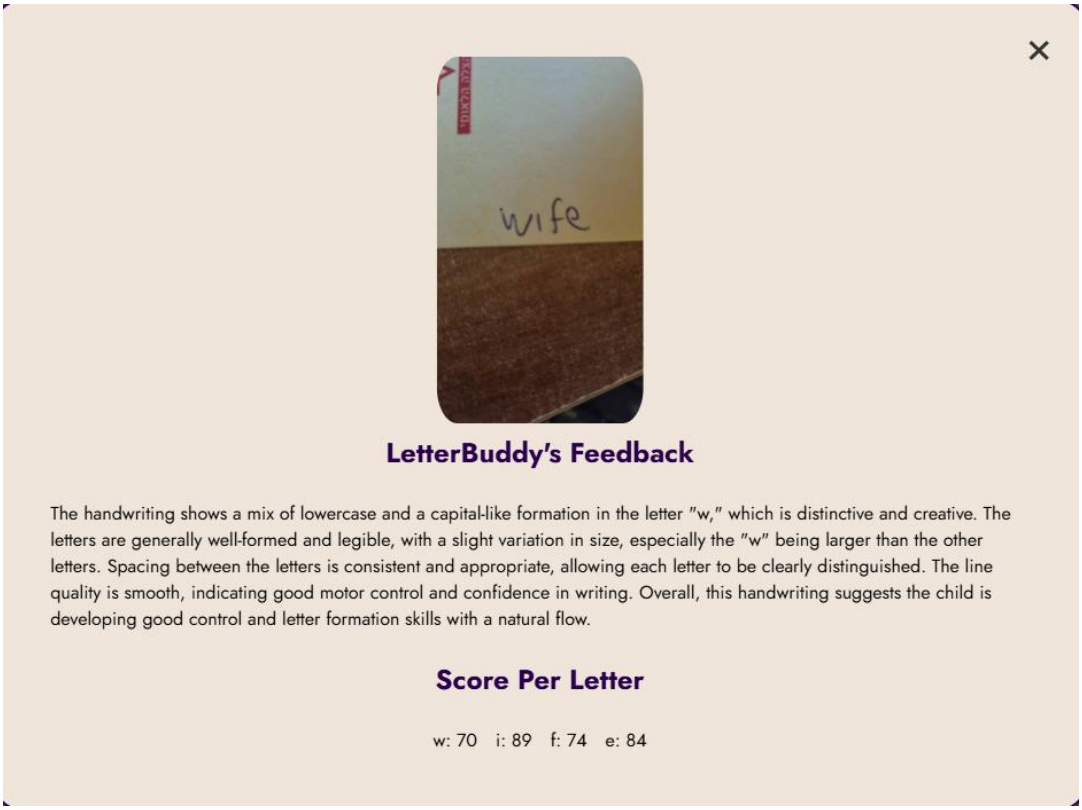
As we can see in the images below, our ensemble, together with our scoring algorithm, provided a balanced score between the 2 models, and a more specific and relevant feedback compared to the general feedback we got from using the ChatGPT model directly.

Direct handwriting analysis of GPT 4.1 mini:



PaddleOCR identification:

Handwriting feedback from our platform:



Discussion:

From the experiments and results we have done through the project and the sample outputs provided, we inferred the following insights and limitations of our final approach:

Key Takeaways:

- PaddleOCR alone was insufficient for most children's handwriting, and the VLM alone may hallucinate and not provide the desired confidence score per letter. The ensemble balanced both models' strengths and weaknesses.
- Our use of SequenceMatcher and letter confusion mapping led to fairer grading.
- Prompt engineering for each exercise level improved the VLM feedback, aligning it more with expert metrics.

Limitations:

- Requiring the use of a VLM API made the algorithm significantly slower due to the free plan time limitations of the VLM's use.
- The VLM sometimes hallucinates in ambiguous images.
- The VLM is limited in its ability to guide the child for better handwriting practices - guidance in cases of bad pencil holding habits and other possible fine motor skills-related problems that happen during the writing process itself may still require human insight, since it isn't easy to identify based only on the final written image.

Conclusion and future work:

In this project, we have demonstrated how the integration of AI could enhance modern handwriting learning journeys, without feeling forced upon the user. The fast-growing unavailability of teachers in the child's handwriting practice phase and thus the lack of direct, important feedback about their skills could be partially provided by advanced VLM analysis, while letting the adults who guide them understand quickly and easily the child's current progression - strengths and weaknesses, through the statistics.

To further expand upon our current application, such steps could be taken:

- Further expand the content and diversity provided to the child by adding more advanced levels of exercises, such as sentences and essays.

- Gamification - gamify the exercising experience by adding scores, achievements, and rewards to motivate the child to keep practicing.

- Provide the adults with more control over their child's learning - allow them to dynamically change the practice levels of each child after their considerations. They could also provide handcrafted exercises by themselves to better personalize the content for their children.

- Allow tracking of the same child by more than one adult - maybe both the teacher and both parents would want to review their progression.

References:

[1] Bonneton-Botté, Nathalie, et al. "Teaching and rehabilitation of handwriting for children in the digital age: Issues and challenges." *Children* 10.7 (2023): 1096.

[2] Kumar, Ashutosh, J. B. Simha, and Shinu Abhi. "A Deep Learning Approach for Evaluating Children's Handwriting." International Conference on Smart Computing and Communication. Singapore: Springer Nature Singapore, 2024.

[3] Ragone, Grazia, Paolo Buono, and Rosa Lanzilotti. "Designing Safe and Engaging AI Experiences for Children: Towards the Definition of Best Practices in UI/UX Design." arXiv preprint arXiv:2404.14218 (2024).

[4] Marrs, Sarah, et al. "Exploring Elementary Student Perceptions of Writing Feedback." Journal on Educational Psychology 10.1 (2016): 16-28.

[5] Li, Yuchen. Synergizing Optical Character Recognition: A Comparative Analysis and Integration of Tesseract, Keras, Paddle, and Azure OCR. Diss. University of Sydney, 2024.