LetterBuddy's Detailed Design

Introduction:

A web application for handwriting exercises for children, with AI feedback based on an ensemble of OCR and VLM models. Adults guiding the children on this journey can view their progress.

System Overview:

The foundation of the system will consist of the following features:

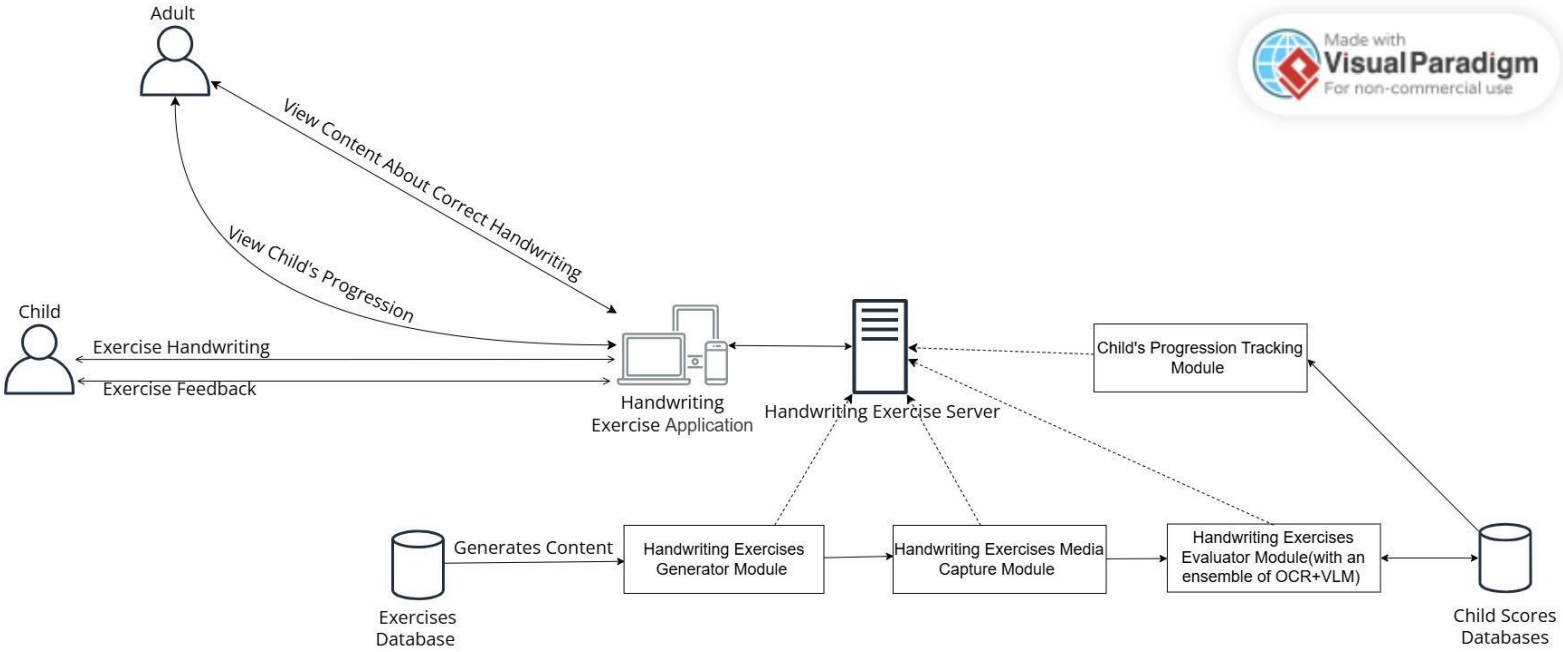
- Flexible AI-based feedback on the writings, partly based on an ensemble of OCR and VLM models.
- Tools to capture the child's writings - the device's camera or gallery.
- A database synchronized with changes according to the child's scores, helping to keep track of the child's current handwriting level.
- Exercises are provided based on the child's current level, thus providing varied and gradual experiences.
- Analytics provided to the adult are displayed by interactive graphs that show each child's progression and performance over time, allowing for more focused face-to-face exercising sessions between the adults and children if needed.

Design considerations:

Due to the project time and resource limitations, we will make the following considerations:

- The OCR's model accuracy: Due to computing and financial limitations, we will use an open-source OCR model called paddleOCR, instead of the more accurate analysis offered by paid API calls to popular models. In addition, our ability to train the model is limited by the free GPU offerings in platforms like Kaggle and Google Colab and the lack of free qualitative handwritten text datasets.
- VLM's API limited requests: due to using the free plan for the VLM, it may be slower and with limited requests as we would have liked.
- Language supportability: Due to limited time and resources, we won't be able to train PaddleOCR's model on handwritten text in every language it supports. This may result in less accurate recognition of languages other than English.
- Performance: Due to limited resources, we won't be able to deploy such an app that can maintain many users simultaneously.

System Architecture:



Component design:

1. User management component
    - Adults' accounts login/registration and authentication
    - Linking children to an adult account.

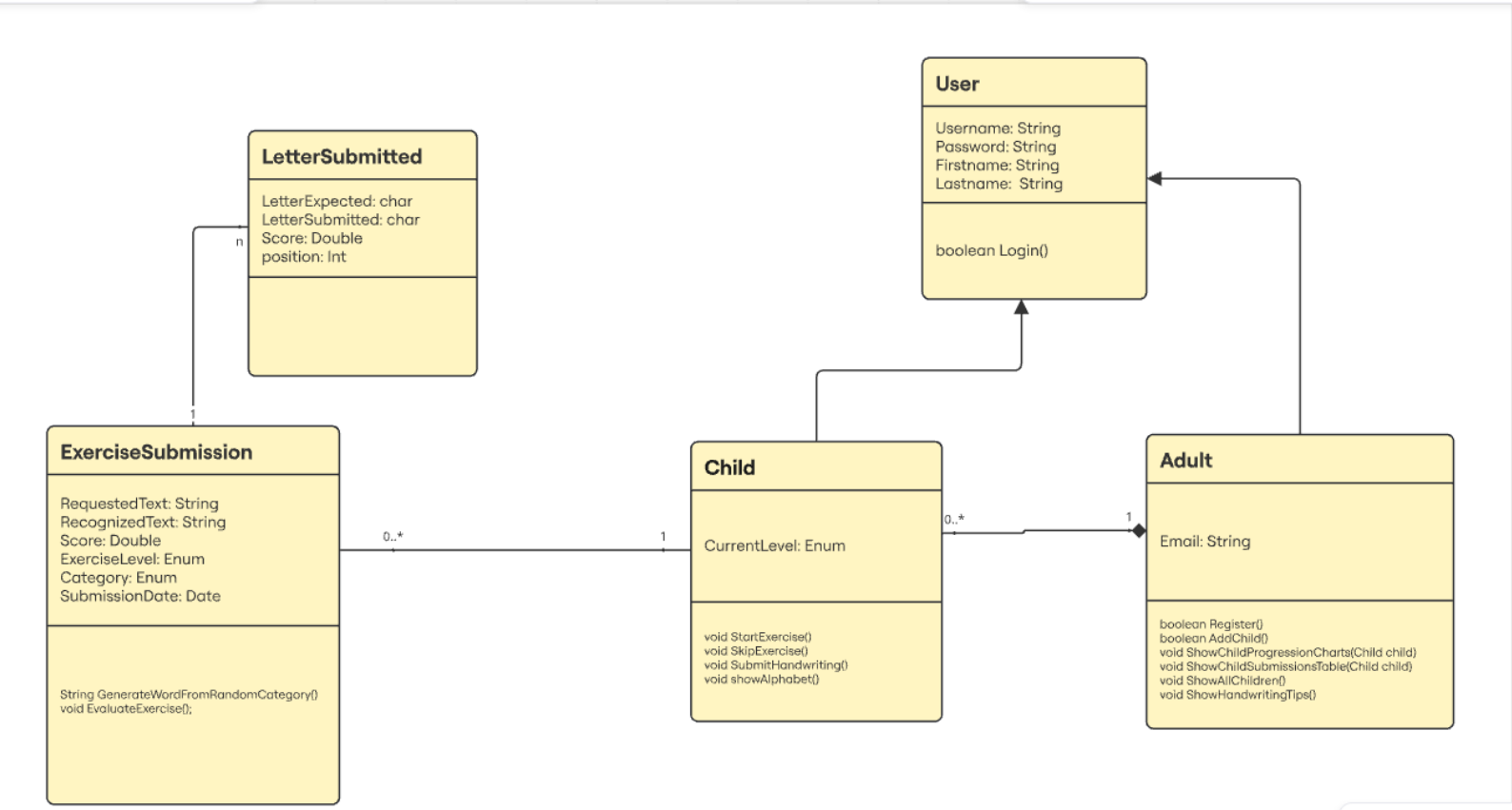
2. Handwriting exercise component
    - Generates handwriting exercises based on the child’s progression.
    - For the level words, it retrieves words by category from our Exercises database.
3. Handwriting media handling component
    - Handles upload and temporary storage of the handwriting media.
    - Integrates the device’s camera for capturing the child’s handwriting.

4. OCR & VLM evaluation component
    - Process handwriting images by an ensemble of OCR & VLM models to evaluate accuracy and provide feedback.
    - Sends the results to the child progression tracker and the database components.
5. Child progression tracker component
    - Calculate child progression scores.
    - Visualizes scores data.

6. Educational content component
    - Store tips, tutorials, and educational content on children’s handwriting for adult accounts.

7. Database component
    - Stores all the system's data, including user info, child scores, tips, and education content.

Data design:


User
Username: String
Password: String
Firstname: String
Lastname: String
boolean Login()
LetterSubmitted
LetterExpected: char
LetterSubmitted: char
Score: Double
position: Int
n
1
ExerciseSubmission
RequestedText: String
RecognizedText: String
Score: Double
ExerciseLevel: Enum
Category: Enum
SubmissionDate: Date
String GenerateWordFromRandomCategory()
void EvaluateExercise();
0..*
1
Child
CurrentLevel: Enum
void StartExercise()
void SkipExercise()
void SubmitHandwriting()
void showAlphabet()
0..*
1
Adult
Email: String
boolean Register()
boolean AddChild()
void ShowChildProgressionCharts(Child child)
void ShowChildSubmissionsTable(Child child)
void ShowAllChildren()
void ShowHandwritingTips()

Algorithms Description:

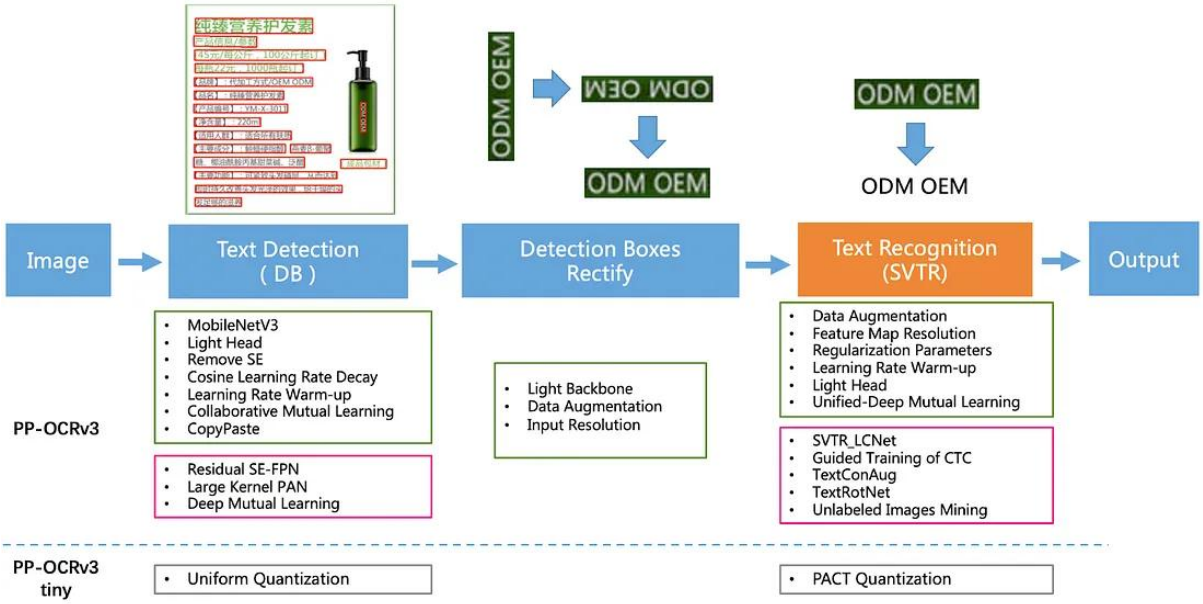
1. **PaddleOCR's detection and recognition:**

   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.



OCR Algorithm Time Complexity:

1. **Text detection:**

   DBNet algorithm (Differentiable Binarization), split into 3 parts:

   1.1 Backbone network for extraction of image features- ResNet50

   O(W * H * C * K^2 * 50) where:

   W, H: width and height of the input image

   C: Number of input channels (3 for RGB)

   K^2: kernel size (for example, 3x3)

   50: number of layers in ResNet50

   1.2 FPN (Feature Pyramid Network) for feature enhancement

   O(W * H * D) where:

   W, H: width and height of the feature map at each pyramid level.

   D: Number of feature channels

   1.3 Head network for calculation of the probability map of the text region

   O(W' * H' * D) where:

   W', H': width and height of the downsampled feature map.

   D: Number of feature channels

2. **Text recognition:**

   SVTR_LCnet algorithm, split into 3 parts:

1. **2.1** Lightweight CNN Backbone (**LCNet**): used to extract features from the input text images

   O(W * H * C * K^2 * L) where:

   W, H: width and height of the input image

   C: Number of input channels (3 for RGB)

   K^2: kernel size

   L: number of layers of convolutional layers

2. **2.2** Scalable Vision Transformer (**SVTR**): A transformer-based architecture that processes the sequential features produced by the LCNet.

   2.2.1 Self-attention model

   O(T^2 * D) where:

   T: sequence length

   D: hidden dimension

   2.2.2 Feed-Forward Network (FFN)

   O(T * D^2)

   T, D - (same as 2.1)

   Overall: O((T^2 * D) + (T * D^2))

3. **CTC** Decoder: Decodes the output of the SVTR into character predictions.

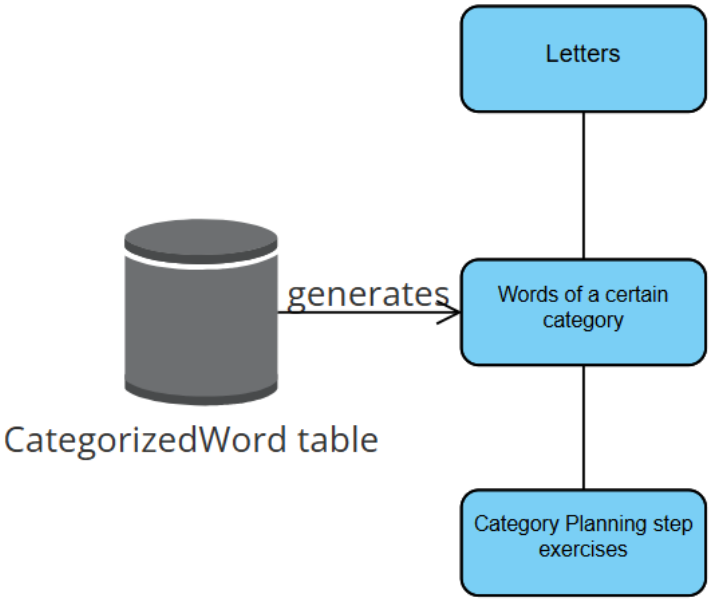
   O(T * C) where:

   T: sequence length

   C: number of possible output characters

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.



The algorithm's time complexity:

According to these steps, each time we want to generate a new exercise for the child according to his level, we need to:

1. If his current level is letters, generating one of them is O(1)
2. If his current level is words, generating one of them requires:
    - Generating a random category from a limited set: O(C)
    - Choosing a random word of that category from the CategorizedWord table - O(W) (W amount of words in category C)
    - Total: O(C + W)
3. If his current level is categories, generating one will only require generating one of the chosen categories - O(C)

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.

The algorithm's time complexity:

1. The VLM and OCR runtime - O(OCR + VLM)
2. SequenceMatcher - O(n * m), where n length of the expected input, and m length of each of the models.
3. Scoring each expected letter - O(n)
4. Calculating Levenshtein Distance - O(m^2)

Overall: O(n*m + m^2)

UI design:

1. Landing page:
   1.1 App logo and name.
   1.2 Getting started/Already have an account buttons.
2. Register Page for adults(parents/teachers, etc): username, password, first name, last name, and email fields.
3. Login Page for both children and adults: username and password fields.

Pages designed only for the children:

4. Handwriting practice page
   4.1 Exercise panel: displaying the exercise for the child to write.
   4.2 Submission form: Accessible button to open the device camera/gallery.
   4.3 Feedback panel: Positive and simple feedback based on the model's evaluation.
5. GIFs page - guiding the child with proper letter writing for each letter.

Pages designed only for the guiding adults:

6. Children's account manager page
   A grid layout of all the children that are connected to that adult, in each row:
   6.1 The child's username
   6.2 A button to access that child's submissions table page.
   6.3 A button to access that child's charts page.
   6.4 An option to add a new child with a pop-up that will be connected to this adult account, by entering their username, password, and name.

7. Child's charts page
   7.1 Child's performance chart in each section - Letters, words, category.
   7.2 Success rates per letter in the language, displayed as bar charts.
   7.3 Line chart showing success rate for each of the child's submissions by day, indicating his improvement over time.
   7.4 Often confused letters by the child(pairs)

8. Child's submissions table page
   8.1. A table (with pagination) containing all of the child's submissions, including their level, submission date, the exercise itself, the score, and an option to view it.
   8.2 An option to sort the table by the submission date, the exercise itself, or the score.
   8.3 An option to search for an exercise by its name.
   8.4 An option to view a specific submission submitted image, VLM's feedback, and score for each requested letter.
9. Content about correct handwriting - a list of articles with references