PsychExtract: Implementation Write-up

CM3070: Final Project
BSc Computer Science
University of London
1941 words

CONTENTS

PROJECT OVERVIEW AND PROTOTYPE CONTEXT

This report follows Template 4.1: Orchestrating AI Models to Achieve a Goal, which frames AI systems as coordinated pipelines rather than isolated techniques. PsychExtract adopts this approach by treating optical character recognition (OCR), natural language processing (NLP), and insight generation as interdependent components within a staged prototype.

Mental health practitioners frequently rely on handwritten clinical notes and patient journals during therapy sessions. These materials contain rich emotional and contextual information but remain difficult to integrate into digital workflows. Manual transcription is time-consuming and error-prone, while most OCR systems are optimised for printed or structured text and perform poorly on unconstrained handwriting. This creates a gap between the expressive value of handwritten material and its usability within digital mental health tools.

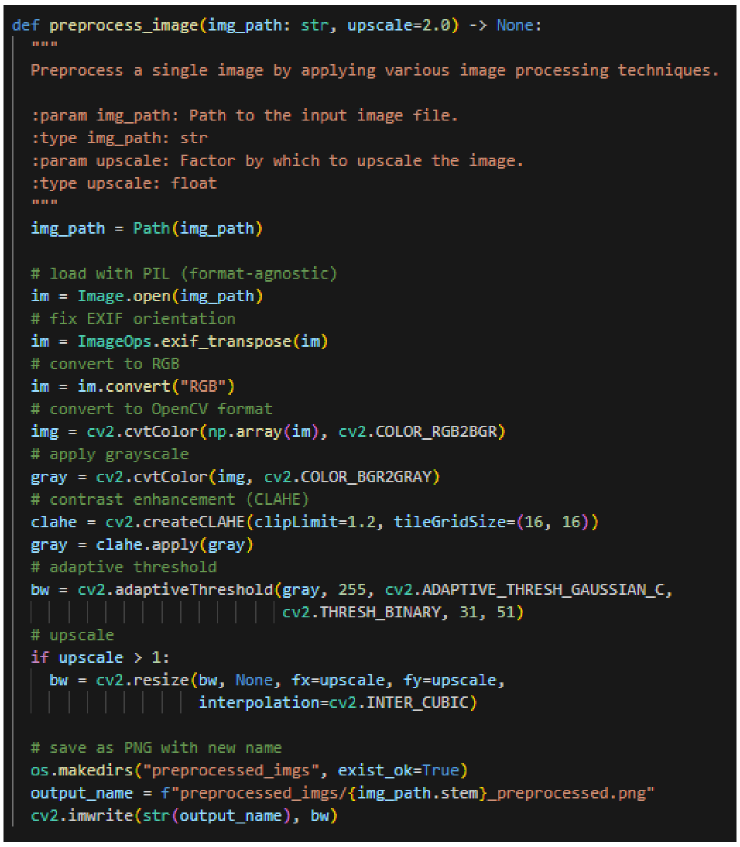
PsychExtract investigates whether AI-based pipelines can help bridge this gap. This stage focuses specifically on OCR feasibility as the system's entry point, evaluating whether modern OCR approaches (particularly transformer-based vision–language models) can reliably transcribe free-form handwritten text into machine-readable form. Accurate transcription is a prerequisite for later stages, including emotion analysis, linguistic insight extraction, and accessibility features such as text-to-speech.

This work represents an early technical feasibility study, not a complete end-user solution. The OCR prototype establishes baseline transcription performance, identifies common failure modes, and compares pretrained OCR models under controlled conditions. By combining quantitative accuracy metrics with qualitative error analysis, this stage determines whether OCR can realistically support downstream NLP modules, providing a necessary foundation for later PsychExtract iterations.

PROTOTYPE FEATURES

The PsychExtract OCR prototype implements a minimal but complete pipeline for converting handwritten notes into digital text. The system accepts images of handwritten journal entries or clinical notes as input, using both real handwritten samples and synthetically generated text images to support controlled evaluation and reproducibility.

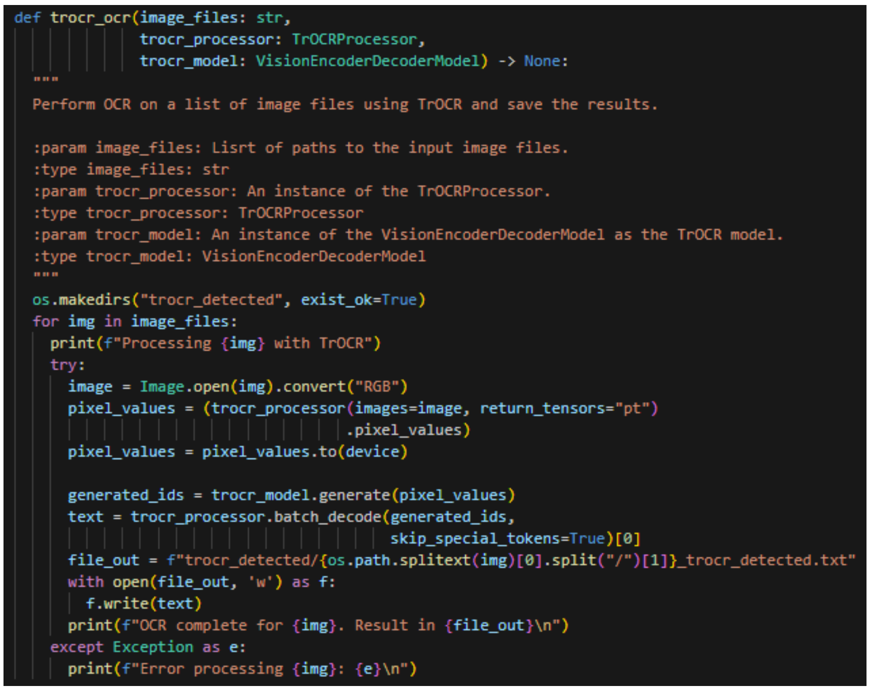
Before OCR inference, basic preprocessing is applied where required by individual models. As illustrated in Figure 1, this includes image resizing, colour space conversion, and normalisation to match model input specifications. No handwriting-specific enhancements (such as stroke thinning or segmentation) are applied, as the prototype aims to assess model robustness under realistic conditions.



```python
def preprocess_image(img_path: str, upscale=2.0) -> None:
    """
    Preprocess a single image by applying various image processing techniques.

    :param img_path: Path to the input image file.
    :type img_path: str
    :param upscale: Factor by which to upscale the image.
    :type upscale: float
    """
    img_path = Path(img_path)

    # load with PIL (format-agnostic)
    im = Image.open(img_path)
    # fix EXIF orientation
    im = ImageOps.exif_transpose(im)
    # convert to RGB
    im = im.convert("RGB")
    # convert to OpenCV format
    img = cv2.cvtColor(np.array(im), cv2.COLOR_RGB2BGR)
    # apply grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    # contrast enhancement (CLAHE)
    clahe = cv2.createCLAHE(clipLimit=1.2, tileGridSize=(16, 16))
    gray = clahe.apply(gray)
    # adaptive threshold
    bw = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                               cv2.THRESH_BINARY, 31, 51)
    # upscale
    if upscale > 1:
        bw = cv2.resize(bw, None, fx=upscale, fy=upscale,
                        interpolation=cv2.INTER_CUBIC)

    # save as PNG with new name
    os.makedirs("preprocessed_imgs", exist_ok=True)
    output_name = f"preprocessed_imgs/{img_path.stem}_preprocessed.png"
    cv2.imwrite(str(output_name), bw)
```

Figure 1. Image preprocessing before OCR inference

The core functionality of the prototype is OCR inference using multiple pretrained models. Each model processes the input image and generates a textual transcription. The system is designed to support comparative evaluation, allowing different OCR approaches (such as traditional engines and transformer-based vision–language models) to be applied to the same input data under identical conditions. The TrOCR inference function, shown in Figure 2, illustrates this standardised inference process.



```python
def trocr_ocr(image_files: str,
              trocr_processor: TrOCRProcessor,
              trocr_model: VisionEncoderDecoderModel) -> None:
  """
  Perform OCR on a list of image files using TrOCR and save the results.

  :param image_files: Lisrt of paths to the input image files.
  :type image_files: str
  :param trocr_processor: An instance of the TrOCRProcessor.
  :type trocr_processor: TrOCRProcessor
  :param trocr_model: An instance of the VisionEncoderDecoderModel as the TrOCR model.
  :type trocr_model: VisionEncoderDecoderModel
  """
  os.makedirs("trocr_detected", exist_ok=True)
  for img in image_files:
    print(f"Processing {img} with TrOCR")
    try:
      image = Image.open(img).convert("RGB")
      pixel_values = (trocr_processor(images=image, return_tensors="pt")
                      .pixel_values)
      pixel_values = pixel_values.to(device)

      generated_ids = trocr_model.generate(pixel_values)
      text = trocr_processor.batch_decode(generated_ids,
                                          skip_special_tokens=True)[0]
      file_out = f"trocr_detected/{os.path.splitext(img)[0].split("/")[1]}_trocr_detected.txt"
      with open(file_out, 'w') as f:
        f.write(text)
      print(f"OCR complete for {img}. Result in {file_out}\n")
    except Exception as e:
      print(f"Error processing {img}: {e}\n")
```

Figure 2. TrOCR inference function

The system outputs plain-text transcriptions, which are saved for both quantitative evaluation and qualitative inspection. Character-level accuracy and word error rate (WER) are computed against ground-truth transcriptions, while qualitative analysis focuses on substitution errors, omissions, insertions, and hallucinated text. Together, these features provide an interpretable assessment of OCR performance within the PsychExtract context.

ALGORITHMS, TECHNIQUES, AND METHODS

OPTICAL CHARACTER RECOGNITION APPROACH

OCR in this project is framed as a multimodal learning problem, mapping visual information from handwritten images to textual sequences. Traditional OCR systems rely on handcrafted features and character segmentation, which perform poorly on unconstrained handwriting. In contrast, modern approaches use deep neural networks to learn a direct mapping from image pixels to text.

Transformer-based OCR models are particularly suited to this task. Vision encoders extract high-level features from the input image, while attention mechanisms allow the decoder to focus on relevant spatial regions when predicting each output token. Formally, given an image tensor $I \in \mathbb{R}^{H \times W \times C}$, the encoder produces a sequence of embeddings $E = (e_1, \dots, e_n)$. A decoder then models the conditional probability of a text sequence $Y = (y_1, \dots, y_T)$ as:

$$P(Y \mid I) = \prod_{t=1}^{T} P(y_t \mid y_{<t}, E)$$

This formulation enables end-to-end transcription without explicit character segmentation, making it well suited to variable handwriting styles.

VISION–LANGUAGE MODELS FOR OCR

Several pretrained OCR-capable models are evaluated, with emphasis on transformer-based vision–language models such as TrOCR and Qwen-VL 2.5. Initial experiments showed that traditional OCR engines (Tesseract, EasyOCR, PaddleOCR) struggle with handwritten captures, motivating a shift toward models that jointly model visual and linguistic structure.

TrOCR is purpose-built for OCR, combining a vision encoder with a transformer decoder trained on large-scale text image datasets. Qwen, by contrast, is a general-purpose multimodal vision–language model that integrates a vision encoder with a large language model (LLM). While not OCR-specific, this allows Qwen to leverage linguistic context during transcription.

Both models convert images into visual embeddings attended to during autoregressive decoding. Self-attention enables alignment between handwriting strokes, character shapes, and language patterns, improving robustness to noise and handwriting variability. Evaluating both models supports a balanced comparison between OCR-specialised and general multimodal approaches.

SYSTEM PIPELINE

The OCR pipeline follows a modular sequence, shown in Figure 3. Handwritten images are collected or generated, preprocessed for model compatibility, and passed independently to each OCR model. Raw predictions then undergo lightweight text normalisation (applied consistently to predictions and ground truth) before quantitative and qualitative evaluation.

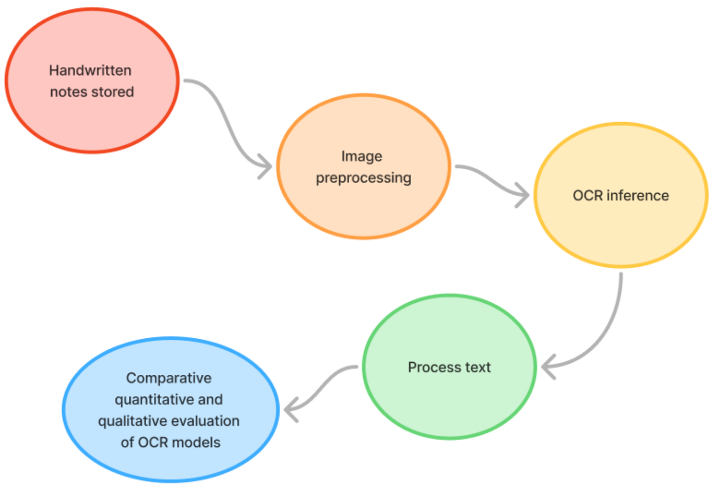


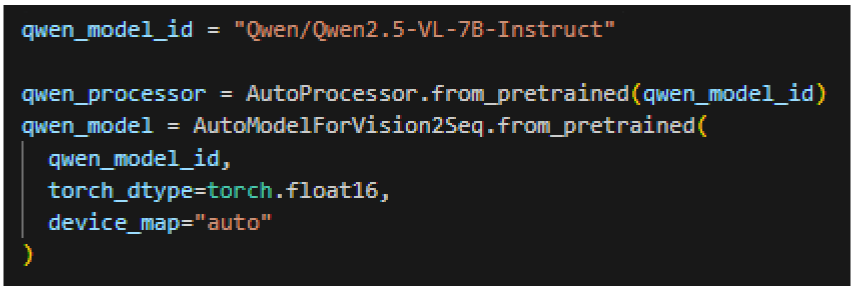
Figure 3. OCR system pipeline

This modular structure supports reproducibility and systematic comparison while aligning with the orchestrated system design principles of Template 4.1.

Code Implementation

Model Loading and Configuration

All models were executed in a Google Colab environment, where memory constraints strongly influenced implementation choices. Large multimodal models, such as Qwen, required explicit device placement and careful configuration. Figure 4 shows the Qwen loading strategy, including pretrained weights and GPU allocation.



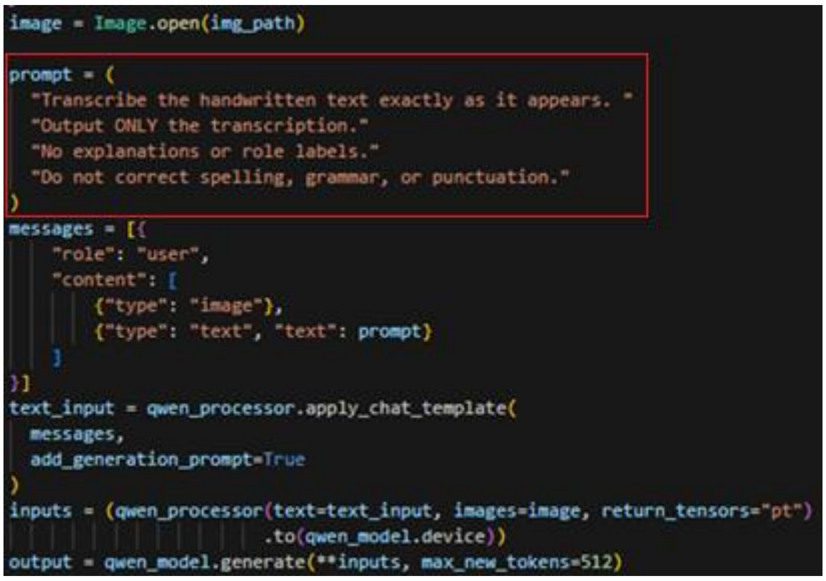
```python
qwen_model_id = "Qwen/Qwen2.5-VL-7B-Instruct"

qwen_processor = AutoProcessor.from_pretrained(qwen_model_id)
qwen_model = AutoModelForVision2Seq.from_pretrained(
  qwen_model_id,
  torch_dtype=torch.float16,
  device_map="auto"
)
```

Figure 4: Loading a pretrained Qwen multimodal model with GPU support

The `device_map="auto"` parameter enabled automatic layer placement across available hardware. Half-precision (float16) inference reduced GPU memory usage while preserving output quality. Due to model size, inference was performed with batch size one, and Colab Pro was required for stable execution. Smaller models such as TrOCR and EasyOCR did not require this configuration, highlighting the computational demands of large vision–language models.

Inference and Prompting Strategy

Instruction-based models such as Qwen require careful prompt design. A strict transcription-only prompt, shown in Figure 5, was used to discourage hallucination and stylistic rewriting.



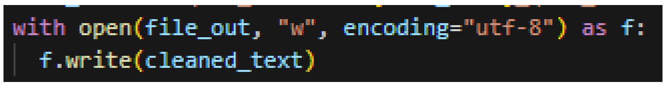
```python
image = Image.open(img_path)

prompt = (
  "Transcribe the handwritten text exactly as it appears. "
  "Output ONLY the transcription."
  "No explanations or role labels."
  "Do not correct spelling, grammar, or punctuation."
)
messages = [{
    "role": "user",
    "content": [
        {"type": "image"},
        {"type": "text", "text": prompt}
    ]
}]
text_input = qwen_processor.apply_chat_template(
  messages,
  add_generation_prompt=True
)
inputs = (qwen_processor(text=text_input, images=image, return_tensors="pt")
                      .to(qwen_model.device))
output = qwen_model.generate(**inputs, max_new_tokens=512)
```

Figure 5: Prompt used for instruction-based OCR

Without this constraint, Qwen frequently generated additional contextual sentences not present in the handwritten input. Traditional OCR engines do not support prompt control and rely entirely on visual pattern recognition, illustrating a key architectural difference.

Output Handling

OCR predictions were saved as plain-text files to support evaluation. Figure 6 illustrates how outputs were written to disk with model-specific identifiers.



```
with open(file_out, "w", encoding="utf-8") as f:
    f.write(cleaned_text)
```

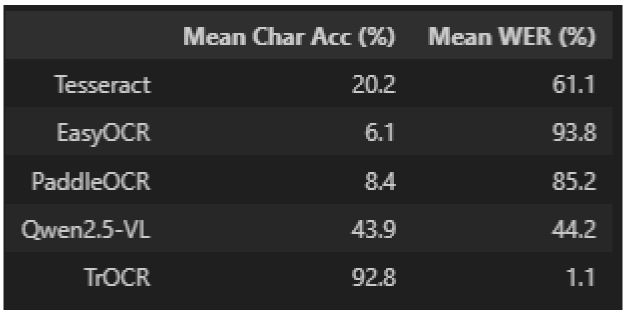
Figure 6: Saving OCR predictions for later review

Filenames encoded the model name and sample ID, enabling direct cross-model comparison while avoiding unnecessary post-processing.

Results and Visual Output

Five OCR models were evaluated: Tesseract, EasyOCR, PaddleOCR, TrOCR, and Qwen. Table 1 summarises mean character accuracy and WER across all samples. Traditional OCR systems perform poorly, with character accuracy between 6–20% and WER above 60%. TrOCR shows moderate improvement but retains substantial errors. Qwen achieves the strongest results, with 92.8% character accuracy and 1.1% WER, substantially outperforming all other models.

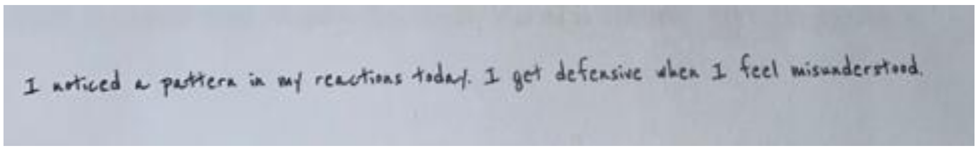
Taken together, the quantitative metrics and qualitative examples reveal a clear performance hierarchy among the evaluated models and demonstrate the effectiveness of transformer-based vision–language approaches for handwritten OCR tasks.



	Mean Char Acc (%)	Mean WER (%)
Tesseract	20.2	61.1
EasyOCR	6.1	93.8
PaddleOCR	8.4	85.2
Qwen2.5-VL	43.9	44.2
TrOCR	92.8	1.1

Table 1: Mean Character Accuracy and WER Across Models

Qualitative outputs reinforce these findings. Figures 7 and 8 show representative handwritten samples alongside model outputs. Traditional OCR results are largely illegible. TrOCR produces readable fragments for clearer handwriting but degrades with increased complexity. Qwen consistently generates coherent, structured transcriptions that preserve sentence flow and word order.


I noticed a pattern in my reactions today. I get defensive when I feel misunderstood.

Tesseract output: L neticed a pattern @ my reactions ada. Le get Arkensive then 1 Feel wisendee steed,

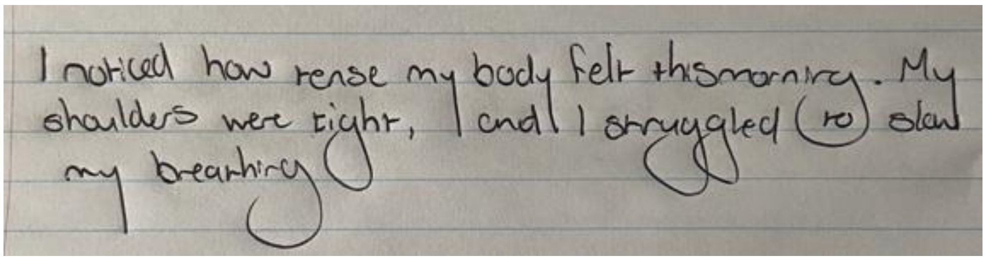
EasyOCR output: 1 Ared putecr "f 'ato} Xasc4: 1 p6r dzfensin #le^ 1 fccl miceadzerfud

PaddleOCR output: 1 natied per i f eat tdf. I gt defenie n I fee isdeesed.

TrOCR output: I arrived a pattern in my reactions today . In yet defensive when I feel misunderstood

Qwen Output: I noticed a pattern in my reactions today. I get defensive when I feel misunderstood.

Figure 7: Text 7a with OCR outputs across models


I noticed how tense my body felt this morning. My shoulders were tight, and I struggled to slow my breathing.

Tesseract output: Vargruh how rease my body Falk
thsmeraica . M
elves woe tune, | bo } ov vole (ro) clad
4 enien Ch

EasyOCR output: ~oked hxw rease ~4 bocly fel+ +hssrorar M shaldus nec richt 1 cnd" 1 8-x4Gled + sky bearwcCA

PaddleOCR output: I noticd how rerse my body fel thsoniy. My shauldes wee cihr, I cndlI or uygled(ro)oln branhirc

TrOCR output: page in the

Qwen Output: I noticed how tense my body felt this morning. My shoulders were tight, and I struggled to slow my breathing.

Figure 8: Text 1b with OCR outputs across models

Evaluation of Prototype

Evaluation focuses on whether transcription quality is sufficient to support downstream psychological analysis rather than clinical automation.

Qualitative OCR Accuracy

Traditional OCR outputs are unsuitable for NLP or human-assisted review. TrOCR offers partial improvement but remains sensitive to handwriting variability. Qwen's outputs are consistently readable and editable, aligning with PsychExtract's goal of assistive insight extraction rather than automation.

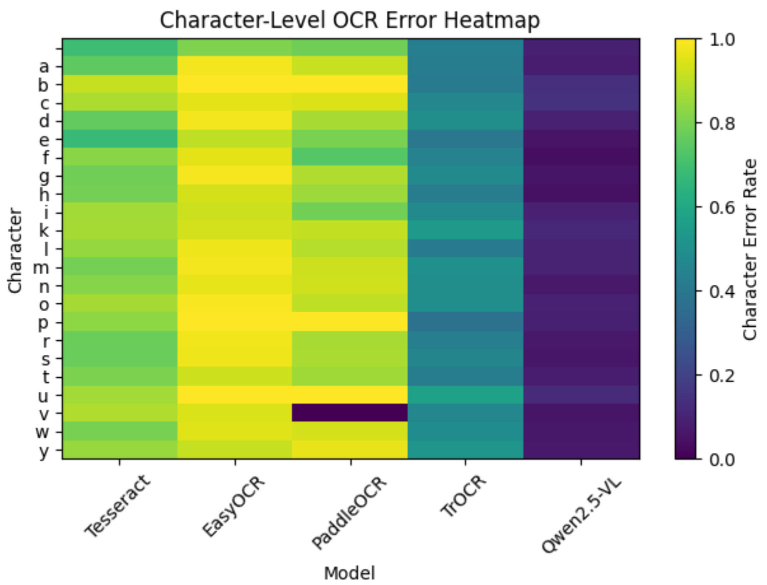


Figure 9: Character-Level OCR Error Heatmap

Clinical Meaning Preservation

Meaning preservation is critical for reflective journal entries. Traditional OCR fails due to fragmentation, while TrOCR frequently omits modifiers and connectors. Qwen best preserves contextual and emotional meaning, producing outputs suitable for human review and downstream NLP, though human oversight remains essential. The system deliberately avoids automated interpretation or diagnosis, reinforcing ethical boundaries between assistive transcription and clinical decision-making

Technical Limitations

Limitations include high computational demands, occasional hallucinations, and sensitivity to image quality. These reinforce the prototype's role as an assistive transcription layer rather than a standalone clinical system.

PLANNED IMPROVEMENTS AND FUTURE WORK

Future work for PsychExtract extends beyond OCR refinement and follows the staged development plan outlined in the project Gantt chart. Once OCR feasibility is established, the next phase focuses on evaluating baseline NLP models for emotion detection and linguistic feature extraction from transcribed journal text. These models will be assessed for both technical performance and the interpretability of extracted insights.

Subsequent stages involve structured user testing with students to assess the usefulness and clarity of generated insights. Feedback from these evaluations will inform the design of a dedicated linguistic insights module, including the refinement of feature sets, rule-based patterns, and category definitions. Peer review and ranking exercises will be used to reduce noise and prioritise clinically meaningful outputs.

Later phases focus on system integration, connecting OCR, NLP, insight extraction, and text-to-speech (TTS) into a single end-to-end pipeline. This integrated prototype will allow testing of information flow, interpretability, and accessibility without requiring a full user interface. Low-fidelity UI wireframes will then be designed and evaluated before final implementation.

The final stages of the project involve full system evaluation, user satisfaction testing, and iterative refinement. Together, these steps position OCR not as the final outcome, but as an enabling component within a broader AI-assisted mental health journaling system.

CONCLUSION

This report documented the design and evaluation of the OCR component within PsychExtract, an orchestrated AI system aimed at transforming handwritten mental health journals into actionable digital insights. Although OCR has been widely studied, applying general-purpose vision–language models to unconstrained mental health journaling introduces a novel trade-off between linguistic fluency and transcription fidelity. By evaluating transcription feasibility and identifying key limitations, this work establishes a reliable entry point for subsequent NLP, insight extraction, and accessibility modules.

OCR alone does not constitute the full system; rather, its evaluation provides critical evidence to inform integration decisions and guide system refinement. This staged, modular assessment reflects the orchestration principles of Template 4.1, validating OCR as a viable upstream component prior to full system integration. Overall, the findings support the feasibility of PsychExtract's incremental development strategy and reinforce the project's direction toward an end-to-end, user-informed mental health support pipeline.