CMPT 310 Project - Group 6

https://www.kaggle.com/datasets/spscientist/students-performance-in-exams/code

https://www.kaggle.com/datasets/datamunge/sign-language-mnist

http://neuralnetworksanddeeplearning.com/chap1.html

https://www.youtube.com/watch?v=f2TUxoaKIsA

—------ receipt databases
https://expressexpense.com/blog/free-receipt-images-ocr-machine-learning-dataset/

https://www.kaggle.com/datasets/dhiaznaidi/receiptdatasetssd300v2/code

MNIST dataset

https://github.com/andyrzwang/cmpt310-rcpt-scan

CMPT 310 Project - Group 6

ACTUAL WORK!!!!!

Todo list

1. Use AI to generate the overall structure and software design of the program
2. Have UML (DATA flow diagram)
3. Compare the structure - Saturday night
4.
5. Set up and have the same python environment
6. Python version 3.10.0??
    a. Install library - tensorflow, pytorch, openCV?....
    b. Make Github issues
7. Download and sort images - data
8. Pre-process the data, select meaningful data (not to bad looking)

Farzan

1. Use AI to generate the overall structure and software design of the program

    Software Design Overview

    Modules:

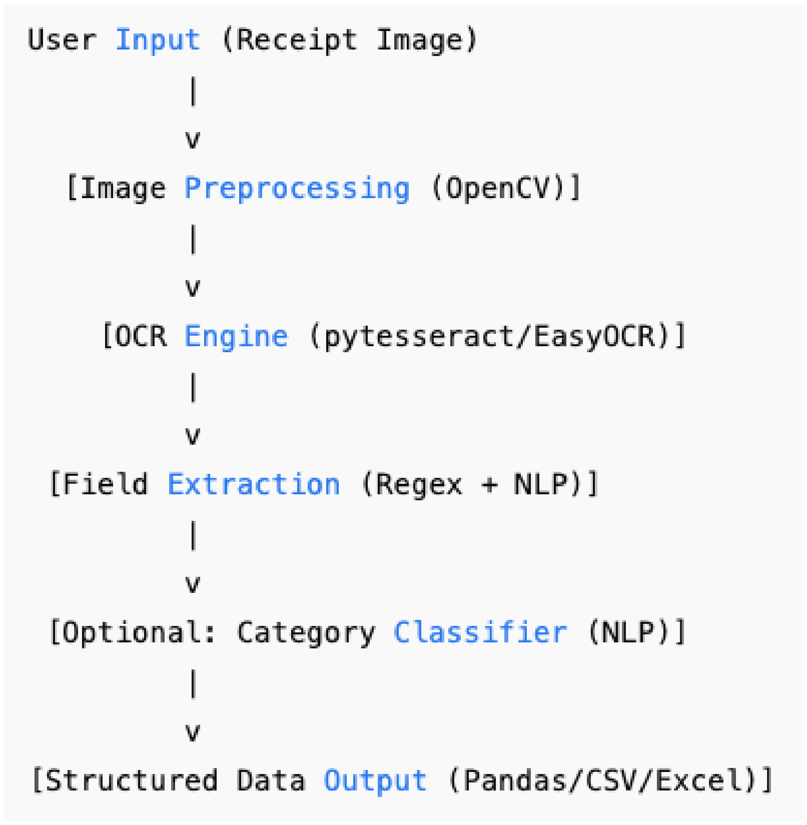
    a. image_input.py – Handles image upload or batch folder reading
    b. preprocessing.py – Uses OpenCV for denoising, thresholding, etc.
    c. ocr_extraction.py – Extracts text using pytesseract or EasyOCR
    d. field_parser.py – Uses regex to extract date, merchant, total, etc.
    e. categorizer.py – (Optional) Classifies receipt type using rule-based or ML
    f. exporter.py – Converts structured data to CSV or Excel
    g. main.py – Runs the entire pipeline (optionally with CLI)
    h. Architecture Style: Modular, CLI-based, with possibility to plug in UI later.

Modules Overview:

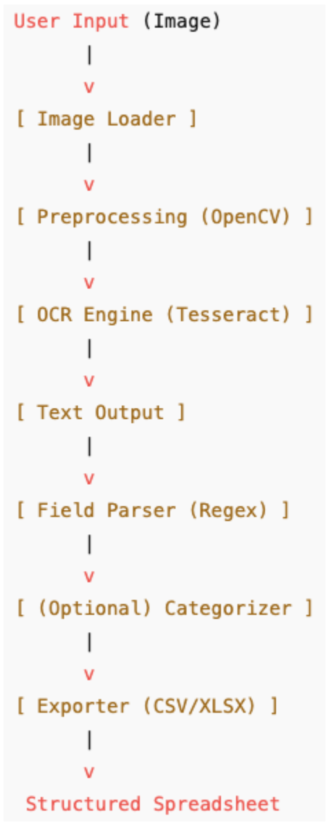
1. main.py
    a. Entry point of the program
    b. Calls each module in sequence (preprocessing → OCR → parsing → export)
2. Image_loader.py
    a. Loads image(s) from file or folder
    b. Handles both single and batch processing
3. preprocessing.py
    a. Uses OpenCV to clean the image (grayscale, thresholding, denoising)
    b. Optionally deskews or resizes image for better OCR
4. Ocr_engine.py
    a. Uses pytesseract (or EasyOCR) to convert image to raw text

5. Field_parser.py
    a. Uses regex and keyword filters to extract:
        i. Date
        ii. Merchant name
        iii. Total amount
        iv. Tax
        v. Payment method (if present)
6. categorizer.py (optional)
    a. Uses keyword mapping to assign a category (e.g., Food, Transport)
7. exporter.py
    a. Formats the output into a DataFrame
    b. Saves it to CSV or Excel using pandas or openpyxl

2. Have UML (DATA flow diagram)



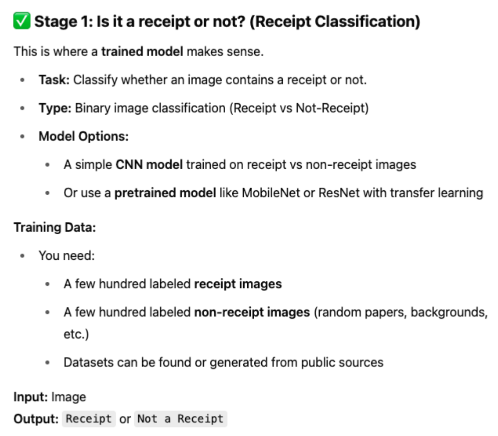
```
User Input (Receipt Image)
         |
         v
  [Image Preprocessing (OpenCV)]
         |
         v
    [OCR Engine (pytesseract/EasyOCR)]
         |
         v
 [Field Extraction (Regex + NLP)]
         |
         v
 [Optional: Category Classifier (NLP)]
         |
         v
[Structured Data Output (Pandas/CSV/Excel)]
```



```
User Input (Image)
      |
      v
[ Image Loader ]
      |
      v
[ Preprocessing (OpenCV) ]
      |
      v
[ OCR Engine (Tesseract) ]
      |
      v
[ Text Output ]
      |
      v
[ Field Parser (Regex) ]
      |
      v
[ (Optional) Categorizer ]
      |
      v
[ Exporter (CSV/XLSX) ]
      |
      v
 Structured Spreadsheet
```

Extra stuff that can be useful and/or important.

CMPT 310 Project - Group 6



✅ **Stage 1: Is it a receipt or not? (Receipt Classification)**

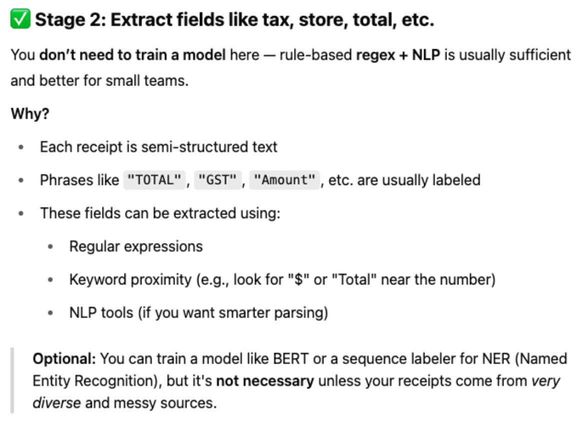
This is where a **trained model** makes sense.

- **Task:** Classify whether an image contains a receipt or not.
- **Type:** Binary image classification (Receipt vs Not-Receipt)
- **Model Options:**
  - A simple **CNN model** trained on receipt vs non-receipt images
  - Or use a **pretrained model** like MobileNet or ResNet with transfer learning

Training Data:

- You need:
  - A few hundred labeled **receipt images**
  - A few hundred labeled **non-receipt images** (random papers, backgrounds, etc.)
  - Datasets can be found or generated from public sources

Input: Image
Output: `Receipt` or `Not a Receipt`



✅ **Stage 2: Extract fields like tax, store, total, etc.**

You **don't need to train a model** here — rule-based **regex + NLP** is usually sufficient and better for small teams.

Why?

- Each receipt is semi-structured text
- Phrases like `"TOTAL"`, `"GST"`, `"Amount"`, etc. are usually labeled
- These fields can be extracted using:
  - Regular expressions
  - Keyword proximity (e.g., look for "$" or "Total" near the number)
  - NLP tools (if you want smarter parsing)

> **Optional:** You can train a model like BERT or a sequence labeler for NER (Named Entity Recognition), but it's **not necessary** unless your receipts come from *very diverse* and messy sources.

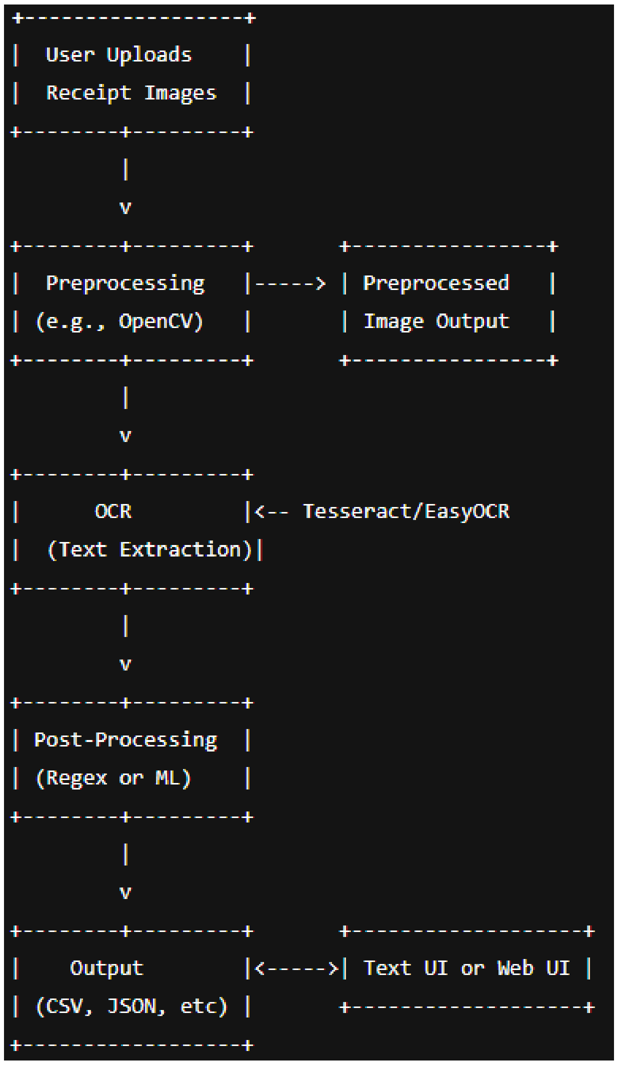
Focus your training effort on:

A binary classifier for detecting whether the image is a receipt.

Then use regex + rule-based methods to extract structured fields.

Daniel

UML



```
+------------------+
|  User Uploads    |
|  Receipt Images  |
+--------+---------+
         |
         v
+--------+---------+       +----------------+
|  Preprocessing   |-----> | Preprocessed   |
| (e.g., OpenCV)   |       | Image Output   |
+--------+---------+       +----------------+
         |
         v
+--------+---------+
|      OCR         |<-- Tesseract/EasyOCR
|  (Text Extraction)|
+--------+---------+
         |
         v
+--------+---------+
| Post-Processing  |
| (Regex or ML)    |
+--------+---------+
         |
         v
+--------+---------+       +------------------+
|    Output        |<----->| Text UI or Web UI |
| (CSV, JSON, etc) |       +------------------+
+------------------+
```

Category	Tools
Preprocessing	OpenCV
OCR	Python regex / spaCy / ML
UI	Textual (terminal UI), Tkinter, or Streamlit
Output	pandas (for CSV/DataFrame)
Dataset Source	Kaggle, Google Dataset Search

TODO LIST

Set up GitHub project and initialize Issues for each task

CMPT 310 Project - Group 6

Collect 2–3 public datasets of receipt images (e.g., from Kaggle)

Set up the project folder structure and development environment

Choose and install OCR tool (Tesseract or EasyOCR)

Research image preprocessing techniques (OpenCV: grayscale, thresholding, denoising, etc.)

Create a UML-style data flow diagram for the system

Build a prototype OCR pipeline: image → text

Design parsing logic to extract fields (date, total, items)

Choose UI approach (text-based terminal, Tkinter, or Streamlit)

Implement batch processing support

Export parsed data to CSV or display in DataFrame (using pandas)

Optionally improve UI and support editing/fixing parsed fields