[Abstract 2](#_Toc164088292)

[Introduction 2](#_Toc164088293)

[Problem statement 2](#_Toc164088294)

[Background 2](#_Toc164088295)

[Aims 3](#_Toc164088296)

[Objectives 3](#_Toc164088297)

[Literature review 4](#_Toc164088298)

[Receipt Extractor 4](#_Toc164088299)

[Paper 1 - Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms. 4](#_Toc164088300)

[Paper 2 - Information Extraction from Scanned Invoices using Machine Learning, OCR, and Spatial Feature Mapping Techniques 7](#_Toc164088301)

[Paper 3 - Computer Vision for Document Image Analysis and Text Extraction 9](#_Toc164088302)

[Mobile Application 11](#_Toc164088303)

[Paper 4 - React Native vs Flutter, cross-platform mobile application frameworks. 11](#_Toc164088304)

[Conclusion 13](#_Toc164088305)

[Requirements specification 13](#_Toc164088306)

[Receipt Section Detection 14](#_Toc164088307)

[Text Extraction 14](#_Toc164088308)

[Format and Classify text extraction. 15](#_Toc164088309)

[User Interface 15](#_Toc164088310)

[Design 15](#_Toc164088311)

[Server 16](#_Toc164088312)

[Receipt extraction feature. 16](#_Toc164088313)

[User Correction Feature. 18](#_Toc164088314)

[Server Communication 18](#_Toc164088315)

[Database 18](#_Toc164088316)

[Application 19](#_Toc164088317)

[Receipt upload for receipt extraction. 19](#_Toc164088318)

[List Receipt Historical 19](#_Toc164088319)

[Analytics 20](#_Toc164088320)

[Implementation 20](#_Toc164088321)

[Server 20](#_Toc164088322)

[Docker Usage 20](#_Toc164088323)

[Makefile 21](#_Toc164088324)

[Receipt Section Detection Model (YOLO) 23](#_Toc164088325)

[Dataset 23](#_Toc164088326)

[Training Process 24](#_Toc164088327)

[Classification/Formatting 24](#_Toc164088328)

[Mobile Application 27](#_Toc164088329)

[Classes 27](#_Toc164088330)

[Widgets 28](#_Toc164088331)

[Pages 30](#_Toc164088332)

[Testing 32](#_Toc164088333)

[Result & Analyzing 32](#_Toc164088334)

[Conclusion 33](#_Toc164088335)

[Reference 33](#_Toc164088336)

[Appendix 33](#_Toc164088337)

# Abstract

# Introduction

## Problem statement

This project of ‘smart receipt management and Extraction’ aims to enhance the efficiency of company accounting processing, which “plays a significant role in the effective management process” (Alabdullah, 2019). A big part of this accounting process consists of recording varied expenses and, for numerous reasons, has always been a manual process. This repetitive process requires a significant amount of time and energy. Therefore, this project aims to develop a solution to this problem: How to facilitate the receipt logging process?

## Background

As previously mentioned above in the problem statement, the purpose of this project is to facilitate the recoding process of receipts. The problems that arise from this manual receipt logging task, as said before, requires individuals to log the different receipts one by one manually, which significantly uses a lot of time and energy. But, by automating this process, enabling this task of receipt logging to be digitalized, it could considerably assist the users in this task, not just by reducing the amount of energy used to complete the process, but also valuable time. Concurrently, there are solutions already in existence across the market, such has “Recipator AI” (AI, no date) or “Veryfi” (*Transform Documents into Actionable Data in Seconds using Veryfi OCR API*, no date) but these applications are limited in functionalities and are not adapted to the Mauritian market, lacking an easy-to-use system for individuals. Moreover, the digitalization of this process of receipt handling would improve the ecological issue of receipt storage and production, thereby reducing the ecologic and environmental impact of this task, as it currently produces a high amount of paper waste and enlarges business’ carbon footprint. In summary, this project aims to tackle the specific task of handling and managing receipts through a user-friendly system using the latest state-of-the-art technologies.

## Aims

The project “Smart Receipt Management and Extraction” system is designed to streamline and simplify the repetitive and time-consuming task of handling, organizing and informatize receipts, whether physical or digital. Therefore, the primary purpose of this project is to serve a comprehensive receipt extraction and management system, offering the user a set of features allowing for a user-friendly user-interface experience.

## Objectives

Literature Review

Gather and analyze similar projects and research literature to examine their technologies used, compare their advantages and disadvantages, and to appropriately apply the state-of-the-art technologies from these papers to the different elements of the current system.

Receipt extraction model

Develop a receipt extraction model which will be used to efficiently extract the data from multiple variations of receipts, offering the user the wanted receipt extraction automation to facilitate the receipt logging process.

Receipt dataset

Gather a large quantity of receipt for the dataset to have a precise, accurate and efficient model for extraction of the user’s receipts as effectively possible, and for this dataset to be used for the training and testing purposes of this receipt prediction model.

Mobile Application

Develop a Mobile application of front-end of the project, to be used by the users as a gateway for the system, which will be partially in a server.

Local Server

Develop a local server which will be used to host the extraction model and the database. It will contain all the “heavy” process need by the system.

System Test

Create tests for the system and its different components, such as the mobile application, the server, the communication between them and the different features which will be provided to the user.

# Literature review

## Receipt Extractor

### Paper 1 - Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms.

This paper from Joel Odd and Emil Theologou (2018) is a study that “investigated if it was feasible to use machine learning tools on OCR [Optical Character Recognition] extracted text data to classify receipts and extract specific data points”. This process firstly extracts the receipt data through different Optical Character Recognition (OCR), then classifies through a model into different categories. They have tested different technologies for the OCR extraction and the model prediction, all listed below in *Table 1*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Technologies** | **Pros** | **Cons** |
| **Optical Character Recognition (OCR)** | Azure Computer Vision API | * Good accuracy in text recognition. | * The security is uncontrollable due to third party dependance. * May require significant data preprocessing. |
| Google Cloud Vision API | * Good performance and time response. | * The security is uncontrollable due to third party dependance. * Managing variations in receipt formats can be complex |
| Tesseract OCR | * Deployable locally * Not dependent of third party. * Strong community | * Varying receipt formats may affect accuracy. * Accuracy depends on image quality. * Slow process for large amount of data. |
| **Machine Learning Models** | Linear Support Vector Classification (LinearSVC) | * 94% accuracy achieved. * Efficient with large text dataset | * Not has efficient for non-linear data relationship. * Require optimal parameters tuning, otherwise reduction in performance. |
| Multilayer Perceptron Classifier (MLPClassifier) | * Can capture complex relationship (non-linear). * Flexibility with multiple parameters. | * May be computationally intensive as the model become complex. * Risk of overfitting if the parameters are not well set. |
| Naive Bayes Classifier | * Easy implementation to handle large dataset. * May be use has baseline for classification problem. | * Can make naïve assumption which will lead to a reduction in accuracy for complex data. * Training dataset balance strongly influence the accuracy. |

Table 1 - Advantage and Limitation of technology used in "Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms" written by Odd and Theologou

From the OCR, three principal technologies are tested “Azure Computer Vision API” provided by Microsoft, “Google Drive REST API” provide by Google, and “Tesseract OCR”. The “Azure Computer Vision API” such has “Google Drive REST API” are efficient and powerful OCR tools but are third party dependent and all their processing power are deported in their own servers. This deported strategy allows powerful OCR but creates delay using API to upload and download the data and create a dependence to their services and could be costly. On the other hand, “Tesseract OCR” is an open-source OCR which can be deployed locally and, therefore, have a quicker response time compared to “API” OCR. In the context of our project, the user will have the ability to correct any error from the extraction, therefore a quick response time is essential for a streamline user-experience and is preferable over a slice reduction of extraction accuracy.

After extracting the text from the receipt, the output is categorized through models. Different models were used, such as the Linear Support Vector Classification, the Multi-layer Perceptron classifier, and the Naive Bayes Classifier, which all have their individual advantages and disadvantages, which are shown in *Table 1*. Their model is based on a text extract before using the model strategy, and therefore, is not used for image recognition.

### Paper 2 - Information Extraction from Scanned Invoices using Machine Learning, OCR, and Spatial Feature Mapping Techniques

The second research paper “Information Extraction from Scanned Invoices using Machine Learning, OCR and Spatial Feature Mapping Techniques” is a project focusing one extracting information from scanned invoice using different technologies for different step (Darsha, 2023). In the context of our receipt extractor project, we will focus on the receipt detection/classification and the text extraction step only.

For the Optical Character Recognition, like in the first paper in the section *Paper 1 - Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms.*, similar technologies were used, such as Tesseract OCR and Google Cloud Vision API. Since we have already discussed the advantages and disadvantages of these technologies, there is no need to re-analyze them. There it shows that these technologies are the state of the art in their domain and prove their efficiency.

As for the text detection and classification, it uses the YOLO (You Only Look Once) model, which is a state-of-the-art object detection application provided by Ultralitics in python, that is shown in *Table 2*. The YOLO model is well known for its rapid speed in prediction, while having a low background mistake, which is an important characteristic for a receipt extraction application wherein the picture would be taken quickly from a mobile phone. However, it also has its own limitations such as the trade-off between speed and accuracy, which could cause problems in case of too low accuracy. Also, the processing could be resource-intensive for training and prediction.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Technologies** | **Pros** | **Cons** |
| **Object detection model** | YOLO version 5 | * Good accuracy in detecting and classifying multiple objects. * Fast inference time and global speed. * Reduction of background error by processing all picture | * Require extensive computation power for many classes. * Potential trade-off between speed and accuracy. * Less densely pack data may reduce accuracy. |
| **Optical Character Recognition (OCR)** | Tesseract OCR | See *Table 1* | |
| Google Cloud Vision API | See *Table 1* | |
| Convolutional Neural Networks (CNNs) | * Good at extracting hierarchical feature from images. * Versatile for a wide range of image recognition. | * High computational resources for training. * Risk of overfitting training data. * Need a lot of tuning and optimization for optimal performance. |

Table 2 - Advantage and Limitation of technologies used in "Information Extraction from Scanned Invoices using Machine Learning, OCR and Spatial Feature Mapping Techniques" written by Darsha.

### Paper 3 - Computer Vision for Document Image Analysis and Text Extraction

The third paper “Computer Vision for Document Image Analysis and Text Extraction” is a research article aiming to improve Optical Character Recognition (OCR) systems, particularly for image processing (Benchekroun, 2022). It explores multiple technologies, such has Convolutional Neural Network (CNN) + Long Short-Term Memory (LSTM) Network for feature extraction from images, Deep CNN + Transformer/seq2seq Network to handle sequential data, Generative Adversarial Networks (GANs) to generate synthetic training data, and Morphological Operations for preprocessing training data with different operation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Technologies** | **Pros** | **Cons** |
| **Optical Character Recognition (OCR)** | Convolutional Neural Networks (CNN) + Long Short-Term Memory (LSTM) | * Effective for non-handwritten text. * Shows significant accuracy improvement with synthetic data. | * Are computationally intensive and may require large dataset for high accuracy. * Struggle with complex patterns in data |
| Deep CNN + Transformer/seq2seq Network | * Capable of parallel processing, offering faster execution. * Can extract complex features from images. | * Require significant computation resources. * Difficulty to optimize due to his complexity. |
| **Synthetic Data Generation** | Generative Adversarial Networks (GANs) | * Increase training data. * Help model learn feature that may not be present in real-world scenario. | * Generated data might not always represent real-world scenarios. * Can reduce accuracy in case of generated data not accurate to the scenario. |

Table 3- Advantage and Limitation of technologies used in "Computer Vision for Document Image Analysis and Text Extraction" written by Benchekrou.

The advantages and disadvantages of these technologies can be found in *Table 3*. Furthermore, the usage of GANs to create synthetic data is an interesting feature to improve the training dataset and, therefore, increases the accuracy if an OCR algorithm is developed from scratch, such as the different ones used in this third research paper. Developing an OCR from scratch without using an already made system, such has Tesseract or Google Cloud Vision API, could allow for a more precise extraction especially for Mauritian receipts, thereby it will increase the development time, but it will be limited by its training data, which in out context, is limited and, therefore, would not be recommended compared to the powerful pretrained system.

## Mobile Application

### Paper 4 - React Native vs Flutter, cross-platform mobile application frameworks.

The last research paper “React Native vs Flutter, cross-platform mobile application frameworks” focuses on comparing two frameworks to develop cross-platform mobile application: React Native and Flutter (Wu, 2018). These two frameworks are prominent factors in mobile development, and their advantages and disadvantages are available in *Table 4*. React Native, which was developed by Facebook, is using JavaScript and React, which are famous programing languages. However, for complex development, it will require a specific programming language per platform. On the other hand, Flutter is developed by Google using Dart, which is not as widely adopted as JavaScript, but uses a single code base for both iOS and Android. However, this single code base creates a larger app size compared to React Native app.

In the context of the receipt extractor application, the emphasis will be on the execution speed and stability on the mobile app. Therefore, Flutter, as shown in *Table 4*, is fast with different features for development, such has the hot reload with the portability from Android to iOS without code modification.

|  |  |  |
| --- | --- | --- |
| **Programming Language** | **pros** | **Cons** |
| **React Native** | * Strong Community Support with big tech contribution. * Bring modern web techniques to mobile support. * Use JavaScript syntax extension for designing UI. * Access native hardware feature like camera and storage. * Encourage modularity and reusable of component. | * Have performance limitation compared to native app in complexes scenario using JavaScript thread or memory optimization. * Highly dependable on third-party libraries for navigation and file system operations which may affect consistency and the reliability. |
| **Flutter** | * Hight-Performance using his own rendering engine for view component offering a close performance to native application. * Using Dart Programming language which is efficient for memory management and garbage collection offering fast performance. * Providing customizable widget for development. * Hot-Reload feature for development. * Assuring a consistency across the different platform. | * Produce larger application size due to the different widget renderer that may affect the app size. * Relatively new community compared to react native which may affect the resources available. |

Table 4 - Advantage and limitation of Flutter and React Native from "React Native vs Flutter, cross-platform mobile application frameworks" written by Wu.

## Conclusion

# Requirements specification

[ tell the feature, why these features related with literature review].

* Detect Receipt Part
  + Inspired by paper 2.
  + Help for classification.
* Extract receipt data.
  + Key feature in all the papers
* Classify Receipt
  + Important for analytics
  + Use of an LLM
  + Refer to paper 1 and 2.
* User interface
  + Give access system to users.
  + No viable solution out of the paper.
* Database management
  + No referring in current paper.
  + Storing of historic for user -> may provide analytics.

## Receipt Section Detection

As referenced while analyzing *Paper 2 - Information Extraction from Scanned Invoices using Machine Learning, OCR, and Spatial Feature Mapping Techniques ,* it presented technologies of detection of the different sections of a receipt before applying the text extraction (Darsha, 2023). These strategies will be kept by detecting the key element of the receipt to increase the formatting and classification of the receipt data by developing an image recognition model to identify the precise areas representing some elements of the receipt, such has the total, items list, time, and shop information.

## Text Extraction

A key feature of the receipt extraction will be the character recognition which will allow the digitalization of the data from the paper to the receipt management system. As per the inside from the work from Odd and Theologou (Odd and Theologou, 2018)(*Paper 1*), Benchekrou (Benchekroun, 2022)(*Paper 3*) and Darsha (Darsha, 2023)(*Paper 2*) showing Optical Character Recognition (OCR) technologies, such has Tesseract OCR and Google Cloud Vision API, along with advanced machine learning models. Presented in all three research papers, it shows the importance of the text extraction as a vital feature as it impacts the precision of the global system.

## Format and Classify text extraction.

After extracting the raw data from the receipt, a formatting and classification showed in *Paper 1 - Utilize OCR text to extract receipt data and classify receipts with common Machine Learning algorithms* and *Paper 2 - Information Extraction from Scanned Invoices using Machine Learning, OCR, and Spatial Feature Mapping Techniques* underscore the effectiveness of the machine learning algorithm for text classification using models such has LinearSVC and MLPClassifier. This approach facilitates the classification of the receipt and the formatting of the output to enable the uniformities of the data structure for a possible database storage and analytics.

## User Interface

Even though the papers criticize the receipt extraction, it showcases multiple technologies for the processing, but none of them provide user-interface for common users to employ these technologies, which without this will restrict the usage of the software to specialized set of users. Therefore, a User Interface (UI) is required to increase the spectrum of possible users. As showed in *Paper 4 - React Native vs Flutter, cross-platform mobile application frameworks ,* comparing Flutter and React Native mobile developing framework (Wu, 2018). These frameworks demonstrated the portability of the UI development through multiple devise common use of the system.

# Design

[ design of these feature (globally) with the technologies I think I’ll use (note at this point I haven’t start working in the artefact, so purely theorical]

After discussing the existing work in *Literature review* and discussing the different requirements in *Requirements specification*, we can now discuss the design of the system application. At that stage of development, it can be separated into two distinct parts, which will be working together simultaneously. The server side, which will manage all the computationally exhausting processing, such as the “receipt extraction feature” which will reduce the execution time and provide a better user experience while communicating with the database. Also, everything will be built in a docker container to offer an easy deployment on any device. And on the other hand, the mobile application will be used as a gateway for the user to operate the system and communicate with the server. In the next following sections, a list of the features will be integrated.

## Server

* + Receipt section detection model.
  + Receipt extraction feature (section detection + OCR + Classification + user).
  + Prediction user correction feature.
  + Server Communication.
  + Database.
  + Database management.

### Receipt extraction feature.

As previously mentioned, the receipt extraction is the key feature of the system which has global purpose to extract from a receipt image into a specific format while predicting the type of receipt. The fields which will be extracted are the shop information, the list of items, the total, and the date of the transaction, which are to be formatted into a json format to be stored in a database. The structure of this feature is composed of three major parts; the receipt section image recognition model, followed by the text extraction of this section, and finally, the formatting and classification of the receipt extracted data.

The receipt section detection model is the first step of the receipt extraction feature. The model will compose of a YOLO v8 model to predict the different receipt sections, which are explained in the preceding paragraph trained by Mauritian receipt to better predict the section of the receipt. To describe in more detail, the prediction will take input as a picture of 640 per 640 pixel and return for each class (chop information, item list, …) their coordinate on the picture.

The second step of this feature is the extraction of the text display of these class extracts through OCR, for each detected class, a sub-image composed of coordinates of the class prediction, and the extracted text. As for the choice of the OCR, based on the different OCR technologies showed in the *Literature review*, the OCR used would be Tesseract OCR due to its high efficiency and capacity to be deployed locally without using a third-party service.

Finally, the classification and formatting of the of the output of the previous function output. This task will be performed by a Custom Multi-Layer Perceptron (MLP) to, firstly classify the receipt into categories (groceries, restaurant, cosmetics, electronics, etc.), then format all the information into a specific format (see *Figure 1 - Receipt Extraction Feature output format*) in json for an easy manipulation afterward.



Figure 1 - Receipt Extraction Feature output format.

### User Correction Feature.

While the data will be automatically extracted, the user will have the opportunity to “review” the extracted data, through the flutter application interface, to ensure the data is correct, which will be stored in the database. The corrected data will be set as “reviewed” and will be accessible by the analytics feature to ensure the accuracy of the data. The structure of the feature can be seen in the sequence diagram.

[ sequence diagram to be input]

### Server Communication

As for the communication between the server and the application, the HTTP (Hypertext Transfer Protocol) will be integrally handled by a FastAPI application host into the server. The primary function of the FastAPI app is to execute the requested server feature and relay back the result to the mobile app. Each endpoint will be related to a feature of the server and therefore, forming the server’s operation core.

### Database

The server architecture will include a database system which will maintain user and receipt’s related data. Using Docker technology, the server will use a PostgreSQL database initializing alongside the server’s image. During the server’s image building, the PostgreSQL image will be initialized, and if in absence of pre-existing data, the database schema will be initialized through the execution of an SQL scripts. The usage of docker will ensure a seamless integration of the system.

The database schema is composed of three primary tables, outline as follows:

* Users table: A table for user related information.
* Raw Receipt Table: A table to store the unprocessed data before any data manipulation.
* Receipts Tables: A table which will contain all data related to the receipt extraction.

Each table use a specific purpose and all tables’ keys can be found in the database schema showed in [insert reference to database schema]

## Application

* + Upload receipt picture for receipt extraction.
  + List receipt historical.
  + Analytics generation
  + Application communication with server.

### Receipt upload for receipt extraction.

On the user side, the key feature of the application is the upload of the receipt of the extraction of its data to be stored in the database. The first step is the loading of the receipt, either through the uploading from the phone gallery, or a live capture through the camera. It is then followed by the upload to the server through http post request, where the receipt extraction will be done. The result will be stored as “pending” status in the database before being sent back to the mobile application. With the result of the extraction, the user will have the option to review and modify the result to correct potential error before being sent back to the server to update the receipt data stored in the database and set is status as “reviewed”.

### List Receipt Historical

The application will allow the user to review their historical entries in a form of a list where key information about the receipt will be display. It will query the “reviewed” status receipt from the user before formatting the data for display.

### Analytics

The application will offer the user the possibilities to show analytics through their data stored in the database. When this page has loaded, the current user’s data will be requested to the server, which will query it from the database back to the mobile application. The data will be processed directly from the application during the building of the widget to create different analytics for the user. The first graph provided will be a line graph showing the sum of purchase/s per day within a range of one month, with the capacity to display for older month/s. It will allow the user to keep track of the money spent during the month. A second graph provided will be generated as a pie graph that shows the sum of money spent per receipt categories (groceries, restaurant, etc.) per month, also with the ability to see data of older month/s. It will allow the user to view the amount for each category and allow it to manage a potential budget.

# Implementation

[ implementation that I have done, how I have done, reason if change from design section (e.g.: using chatgpt instead of homemade llm)]

## Server

During the development of the server, multiple technologies where used, alongside multiple development tools and technique. This section is going to focus on the key component of the server and the modification needed to be done compared to the original design. In a first time, the integration of docker in the server will be discuss, follow-up by the usage of make file. And finally get in details about the Receipt Section Detection Model and the modification needed to be done has for the classification/formatting section of the server.

### Docker Usage

Since the server is the backbone the whole system, the capacity to be easily deploys into server with public IP to allow a connection through internet. For developing purpose, the server was host locally using the hosting machine private IP to make the http request connect from the mobile application to the server with the objective in the future to be deploy into server with a public IP. The solution for an easy migration in the future is the usage containers technologies, and in this case Docker. Two images are built from the docker-compose.yml file, firstly the app image building alongside with a Dockerfile file which set the python environment of the app to Python3.9 and install the different dependency and python package required for the different process from the server to run. In addition, the whole server has been package and install by the Dockerfile and finally at the end of the initialization of the app image, start the FastAPI application. The second image build is a Postgres image to build a PostgreSQL database which the server will use to store data, mapped to a ‘/data’ folder, if this folder doesn’t exist the initialization will build the database, its schema and mock data by running the SQL file placed in ‘/postgres\_init’ folder. All these actions allow the server to be quickly and easy deploy into any kind of computer or server powerful enough as long that the support have docker.

### Makefile

To facilitate the usage of the server and reduce the complexity of command to use the server. A Makefile is integrate with a set of rules useful for the user and use the server and some specific case of the mobile application. There are 6 rules that can be use at the root of the project:

**‘make server-build’**: This rule is a simple set of command which will go to the server code root and build the images of the server without starting it. It is mostly useful for developing purpose.

**‘make server-start’**: This set of command does a bit more than just starting the server’s and database’s images. Due to the local usage of the server in within local network, it firstly run bash script which get the current host machine IP and update a .env file which is use by the mobile application when building the executable (apk) to construct the host for the http requests. After updating the IP in the .env file, the rule gets into the server’s code root to start the server containers and FastAPI application.

**‘make server-stop’**: This rule, similarly to ‘make server-start’ has for purpose to change the status of the containers and stop then without the need to alter the IP.

**‘make rebuild-db’**: This set of command is to be use in cautious because it completely rebuilds the database by removing the current database completely. Firstly, it stops the containers if running, then remove the mounted database base folder ‘/data/’, and finally restart the container which will recreate a clean database base on the schema in the ‘/postgres\_init/’ folder.

**‘make server-test’**: This rule objective is to execute the server’s set of tests, but the complication is that for effectively test the server’s features as a local deployment, need to be run inside the docker directly. The Makefile rule then use the ‘exec’ command to connect to the container through his name and run the pytest set of tests to ensure the well execution of its features.

**‘make application-test’**: This set of command, such has ‘make server-test’ as for purpose to test the different features and widget of the mobile application, but this time without the need to connect to the container. It just goes to the application code base and run the flutter inbuild test command to run all the test file in the ‘tests’ folder.

**‘make test’**: This rule with it less specific name, englobe the execution of both testing: server and mobile application. It executes in order the server’s set of tests, then the application set of tests by using the two previous testing rules. More details about the testing will be in **[insert cross reference to test part]**.

For more details about the Makefile and the command execute, it is available in the appendix section has **[insert cross reference to Makefile part]**.

### Receipt Section Detection Model (YOLO)

As seen in the *Literature review*, there is multiple way to predict different element of an image, but the state of the art is Yolo provided by Ultralitics. To be more precise we are using YOLOv8 which is “the latest version of the acclaimed real-time object detection and image segmentation model”(Ultralytics, no date). Moreover, Ultralytics provide multiple size of the Yolo model (nano. Small, medium, large and extra-large) which we will use the biggest version yolov8x.pt for better performance.

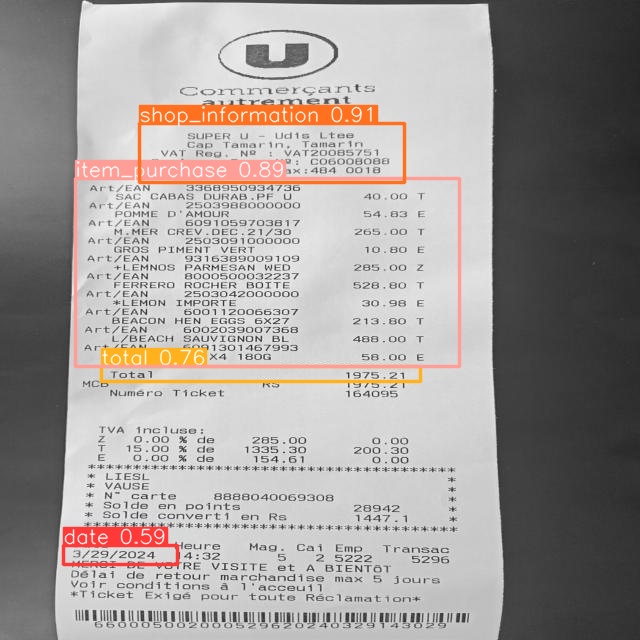


Figure 2 - prediction model area with prediction percentage.

The model will take has input a reformat image of 640 per 640 pixels before applying the model to predict the coordinate of the different label. The set of labels represent the key section of the receipt containing the wanted information to be extracted afterward, the labels are the date, the shop information, total and the items purchase as showed in *Figure 2*. The usage of this text prediction serves as a cleaning of unhelpful in the context of the system such has the vat information or cashier information. This step reduces the amount of data which will be send to the classifier and formatter which will reduce potential error due to the high amount of parasite information.

#### Dataset

As for the dataset require for the training and testing, it was manually gathered from day to day living to fit the reality of receipt used to increase is efficiency in the most regular receipt provided. It is composed has for most of the receipt from Super U, London Supermarket, SmartBox, Intermart, Artisan Coffee, Central Electricity Board and Mauritius Telecom. From this database, the annotation was made using in a first time Roboflow web-application (*Roboflow*, no date) providing some good annotation and dataset increasing feature but mostly limited using free version, then were use Label-Studio which is an open source locally deployable labelling platform (*Open Source Data Labeling*, no date) which contrary to the free access of Roboflow doesn’t have an dataset export limit. The data is then exported into YOLO format making two folders, one for image and one for labeling sharing the same name. As for the dispatching of the dataset, Yolo work with a separation in 3, the training dataset which is used to train the model, the validate dataset used to evaluate the dataset during the training and use to avoid overfitting by tunning the model parameter, and finally the test dataset which is used to evaluate the final performance of the model after being trained.

#### Training Process

The training of the model, despite how Ultralitic simplify it, it is still require for the larger model like the one we use, a lot of computational power, more than my personal laptop (Apple MacBook Air 15 equipped with a M2 chip) can process within a reasonable amount of time. During my test, the training took around 3 minutes per epoch using the laptop. A solution for this computational power is given by Google which provide temporally virtual machine with a powerful GPU. To be more in details it provides an environment to execute python code using a Tesla T4 GPU with 15 GB of ram in addition of 12.7 GB of ram. Using this service, has allow a reduction of the training time to 3-4 seconds per epochs. The parameters from the yolo model get automatically tune by the training process using the validation dataset, the only parameters manually set is the epoch parameter which in the context of our system where set to 200 after multiple try and retry through multiple versions of the training dataset.

### Classification/Formatting

As said before, in the receipt extraction feature, the extracted data of the receipt prediction from Tesseract OCR needs to be classify and formatted. Therefore, due to the quality of the extraction, which is messier than plan, which make the usage of a custom Multi-Layer Perceptron unrealistic for the complexity of task with the current setup of the system, amplified by the computational limitation of my personal laptop. This feature of classification and formatting being a key element in the receipt extraction feature, other solution needed to be tried to find a viable solution.

After multiple research for solution, language model (LM) sort out to be pretty efficient for classification purpose (Lenzmann, 2024). Therefore, in the optical of offering a cost-effective system to the user, a local deployment of a LM on the hosting machine could solve the current problem. A small officiant LM “Starling-LM-7B-alplha”(Jiao, 2023) using 7 billion parameters were tried to be deployed. But still due to the limitation of the host computation power, the response time during the test was way longer than it can be for a realistic usage (around 5 to 7 minutes per query) and therefore not be viable solution.

Since the problem being the computational power to host a LM, the solution is to search for a service which give access to LM while handling the heavy computation power to host it. The obvious and efficient solution is to use OpenAI API to access and use they Large Language Model (LLM) to act has classifier and formatter. The system therefore use “gpt-3.5-turbo” with the prompt show in *Figure 3* to restrict the usage of the LLM into the classifying of the receipt data and to output only a json format has showed in *Figure 1 - Receipt Extraction Feature output format*. However, this solution comes at a cost price for the project, as OpenAI API is not available for free.



Figure 3 - Chatgpt classifying and formatting prompt.

## Mobile Application

For the development of the user side, the mobile application is develop using Flutter framework from Google which is program in dart programing language offering a multiplatform which can be open on computers, Android and Apple IOS phone. The development was done on a Xiaomi 11T using Android 14 with Xiaomi Hyper OS overlay with some test on an emulation of an Apple iPhone 15 under IOS 17.4. The application code can be decomposed in 3 sections: classes, widgets and pages, which all together form the application.

### Classes

The application class section is composed of two major classes. In the context of flutter, we call “class” a set of useful function which isn’t related to any visible element from the app. There are two feature which fit this description, the data services and receipt class.

The **data services** class has is name relate to, is managing the communication between the application and the server. All its functions are related to each endpoint of the server which are used by the application while formatting the data for the endpoint to accept it and send it through http protocol. The first function ‘fetchReceipts’ query from the server the users “reviewed” status data and format it to return a list of Receipt object (which will be discuss later). The second function is ‘sendPicture’ which format the image selected by the user to send it to the server for extraction, the result from the http is then return for further manipulation. The last function follows the logic of the receipt extraction, it’s the upsert of the reviewed and corrected back to the server to be upload into the database using the ‘sendReviewedData’ function from this class.

As for the **Receipt** class, is a representation of the receipt data to be easily manipulated in the application. The main function of the class is ‘fromJson’ which parse the response format from the http ‘getRecipt’ endpoint from dictionary of data to a list of Receipt object while parsing them into the right datatype. In case of error in the parsing log an error and fill the receipt with a default error value showing an error in the different filed. Other feature from the receipt object is to remap the data into a dictionary similar and the capacity to convert it into string with ‘toJson’ and ‘toString’ functions. These two features are mostly use for the edited of old data to fit in the editing feature and for the formatting for upload to the server.

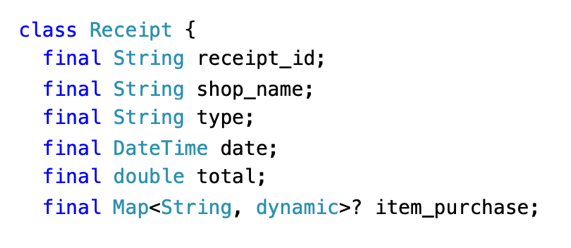


Figure 4 - Receipt class parameters.

### Widgets

Widget is a key feature in the Flutter framework, it gives reusability to created object leading to a global reduction in the code complexity while making it more readable by encapsulating them callable anywhere. In the context of the project, it was used to encapsulate complexes element which would have made the different pages codes too long and complexes and represent key element of the mobile app.

The **Navigation Bar** widget is the main app menu, located at the bottom of every page of the application. It is used to navigate from a page to another. Technically speaking, each time a page is selected it overwrite the current page with the build function of the page selected and change the color of the button to display the current page display.

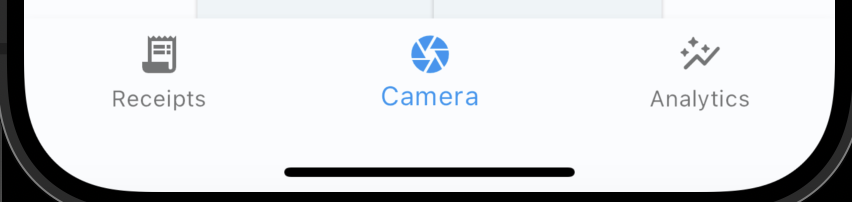
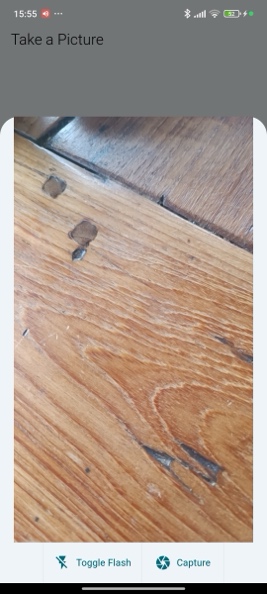
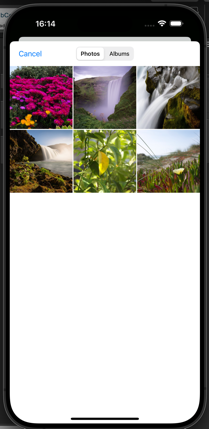


Figure 5 - Navigation bar.



The **camera** Widget is used by the application “camera” page and is one of the two option to select a picture for the receipt extraction. When used for the first time, it ask the user to authorize the access to the camera to fit the European legislation for “General data protection regulation” (*General data protection regulation (GDPR) | EUR-Lex*, no date). When the authorization is provided, the user will be able to capture a picture by pressing the ‘Capture” button with the option to activate the phone flashlight if needed. When a picture is capture, the popup will close while passing the image to the “camera” page.

The **camera roll** Widget, similarly, to the camera widget is the other way to select a picture to be send to the send to the database for extraction. To respect the same legislation, on the first use of the feature, an authorization prompt will pop up to give the application access to the phone gallery. The user will then be able to access his picture and albums to select the wanted picture which will be send back to the “camera’ page.

The **popup-receipt** is a key widget reuse multiple time to display receipt data and edit it. It is used in multiple way and is a layer of security after extracting the receipt data, is used to edit the different part of receipt, newly extracted or add new receipt manually. The different fields have securities which display an error message in case of submission with an empty field. If needed the user can add or remove item through the trash icon button which remove the item or add an empty row for editing by pressing “add item” button while managing the different controllers for the text field. There is also a closing button to remove the popup and the submit button which use the data services class to upload the edited data to the server to be store into the database.

The different graphs display on the “graph” page are also encapsulate as widget as **line chart** and **pie chart** widgets. These widgets take before building the data previously fetch from the database and format them to format the corresponding chart to be display. Per example for the line chart, it filters to the corresponding month while summing the total value per day. From formatted data, the widget draws the chart for the corresponding month. The user also has the option to change the displayed month through a selection field allowing to draw older month.

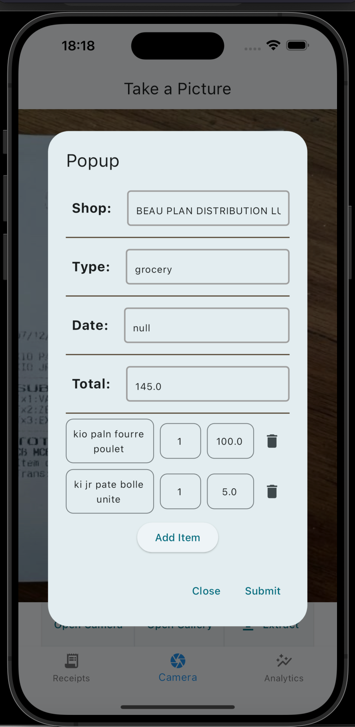
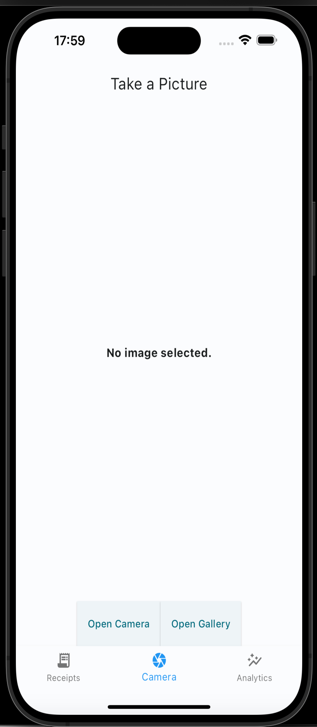
### Pages

Compared to classes and widgets which are tools to be used for the development of the application. On the other hand, pages use these tools to give a flowless and user-friendly experience to the application users. The application currently contains three pages all connected by the Navigation Bar widget.

The first page display by the application when launched, is the **camera page**. This page manages is use for the key feature of the system, the receipt extraction. As showed in [insert reference], it opens a blank page with two button each link to a widget. The first button is ‘Open Camera’ which when press trigger the camera widget to select take a picture. Next to it is the ‘Open Gallery’ which trigger the camera roll widget and allow the user to select an old picture. When a picture is selected from either the camera or the camera roll, it will be display at the center of the screen to ensure that it is the picture selected, and only when an image is display that the “extract” button appear. When press, it’ll use the data services class to upload the picture to the server for receipt extraction, during the extraction process in the server, a rolling animation will be display on the button as showed in [insert reference]. When the server finishes the extraction, the response is then parse and display in the popup widget for user review of the extracted data to ensure his accuracy. After being edited, the user will either be able to cancel process by closing the popup or confirm the edited or not data to be send back to the database and be store has reviewed.

The second page accessible by the use is the **List Page**, this page has for purpose to display in form of a list all the user receipt from the more recent to the oldest. When the page load it will get the user’s “reviewed” data using the data services class while displaying a loading animation. If any error appends during the loading process due to connection error or any other potential error, an error message will be display. A specific message is also display if no data exist for the user. If the response contains receipt, it will be parse into a list of receipt using the Receipt class, from this list of receipt is generated the list which is display while showing the key information to the user. The list display show for each receipt the name of the shop, the total and the date of the receipt. In addition, a small edit button at the end of each row is displayed to give the use the opportunity to see in detail the receipt and the items that it contains or edit it before updating the data store in the database using the popup widget. As well as the list of receipt, there is two buttons at the top, one to reload the data from the database and redisplay it, and an add button to manually create an entry to be send to the database still using on the popup widget. The data manually entered by the user is automatously set as “reviewed” and will be used for analytics.

The last page of the application is the **Graph Page**. This page gathers the different chart and analytics to be display on this page. During the loading of the page, the data services class gather the users “reviewed” data while a loading animation is display and appropriate message is display in case of loading error. At the top of the page is a tabBar menu which load the analytics widget by giving them access to the gathered data. The details about the different analytics are in *Widgets* section.



# Testing

[ In waterfall model: explain the different test case I have done, then testing of model (what test was done, what used to calculate accuracy of the model)]

# Result & Analyzing

[Result of the accuracy , then explaining the reason of these result]

# Conclusion

# Reference

# Appendix