University of Waterloo
Faculty of Engineering
Department of Electrical and Computer Engineering

ReceiptIt: A Receipt Image Recognition Application

Detailed Design and Project Timeline

Group 2020.05

Prepared by

Name: Ziyan Liu	Student Number: 20608393
Name: Boyang Cheng	Student Number: 20619518
Name: Tianpeng Hong	Student Number: 20567046
Name: Jinming Zhang	Student Number: 20613667
Name: Zhidong Zhang	Student Number: 20619543

Consultant: Wojciech Golab

9 February 2020

Abstract

Nowadays, Canadians are feeling more in debt than ever with 90% claiming to have more debt today than they did five years ago. However, only 47% of Canadians plan their spending based on budgets due to how tedious bookkeeping could be. The objective of this project is to implement a mobile application that optimizes the process of expense management by automating tedious aspects of bookkeeping. Major functionalities include receipt scanning, receipt recognition and generating expense reports. ReceiptIt provides a personalized experience via integrating Computer Vision and Machine Learning into the design. The application utilizes a standard client-server architecture with the input to the system being receipt images captured by the client and the output being receipt details extracted by the server. Statistical analysis is conducted on recognized texts to provide dedicated bookkeeping experience. The main advantage of this system is that it provides a single source of user purchase history and robust yet scalable data security.

Acknowledgement

We are sincerely grateful for the help of Professor Wojciech Golab from the University of Waterloo, our consultant, who provided helpful advice in guiding the design of our project. We would also like to thank Professor Sagar Naik for arranging the logistics of the Capstone Symposium.

We hereby confirm that no further assistance was received excluding the previously mentioned acknowledgements.

Table of Contents

List of Figures

List of Tables

1. High-Level Description of Project

1.1 Motivation

Nowadays, Canadians are feeling more in debt than ever, with 90% of them saying that they have more debt today than they did five years ago. However, only 47% percent of Canadians use a budget to plan their spending. [1] One of the significant reasons why Canadians believe they are in debt is that they rarely do bookkeeping to keep track of the things they had purchased, as most of the people consider bookkeeping as a tedious process. Nevertheless, accounting is an essential method for personal financial management, which motivates the group to create an application. It helps people with bookkeeping by recording the purchases on receipts and summarizes them into a detailed report to enhance the personal experience of financial management.

1.2 Project Objectives

The objective of this project is to implement a mobile application that focuses on computerizing receipt information, generating reports for the purchasing history of users and recommending products that are potentially needed by users. To achieve those functionalities, product names, product prices and total prices of email receipts or photocopies of paper receipts are recognized by applying computer vision techniques. The total prices are summed up to generate purchasing reports for a particular period to help the financial management of users.

1.3 Block Diagram

The high-level design of the ReceiptIt project is shown in Figure 1 below. The project is divided into six subsystems which will be discussed in detail in the following subsections.

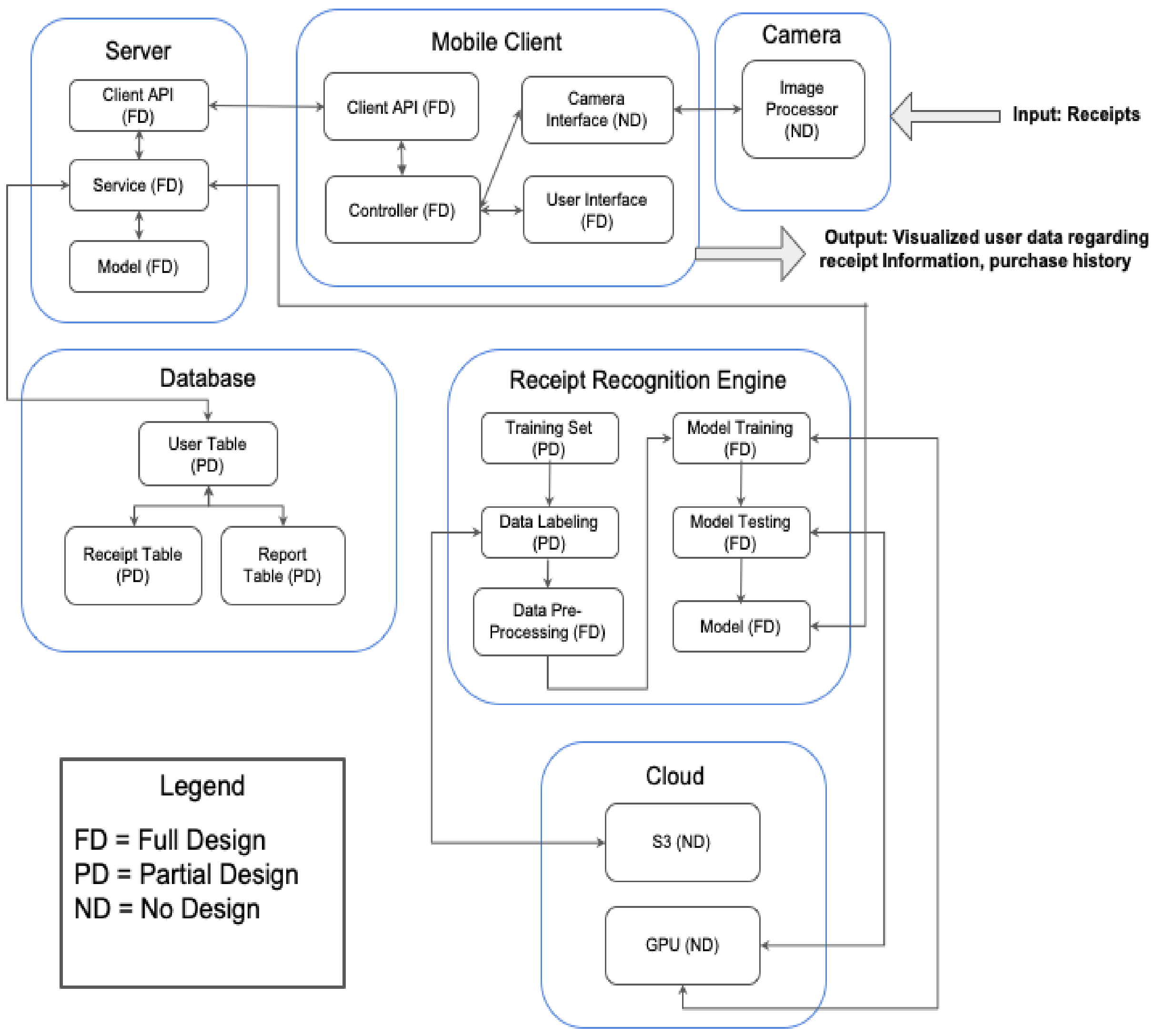


Figure 1: Block Diagram

1.3.1 Camera

The camera system is responsible for receipt images capturing, and images processing happens in the mobile client. It auto-detects and draws the boundary of receipt objects to guide the user when and where to obtain the receipt. Images stitching is provided as a solution for capturing a long receipt. Users take partial receipt images and stitch them as one. The camera system pre-processes the captured image for grey scaling, image brightening, sharpening, and contrast ratio increasing.

1.3.2 Mobile Client

The client presents processed data from the server in a mobile application. It communicates with the server for verifying user credentials when login, retrieving recommended products, receiving expense reports, and accessing purchased history through a client interface. It also plays a role as middleware between the camera system and server for receipt image recognition. The controller module inside the client manages the behavior of the client, camera, and user interface when data is transferring between different sub-systems.

1.3.3 Server

The backend server serves as the backbone of the application connecting the following four subsystems, namely the mobile client, the database and the receipt recognition engine. The server writes and reads from the database where critical information is stored in tabulated format. Meanwhile, the server makes use of the receipt recognition engine to further process extracted texts from images and the recommendation engines to make informed decisions on what products the users will most likely need in the near future. The user interface resides in the mobile client and calls APIs from the server to serve information in a human-friendly format. The APIs are interfaces defining communication protocols between servers and other components. They accept requests from the outside and returns JSON as a response. APIs make use of internal services, which are containers of business logic and computation rules about how data is expected to be manipulated. Further down the chain, there are models, which are representations of database tables with various attributes to imitate objects in the real world.

1.3.4 Database

The relational database, MySQL, is the primary data storage solution used in this project. The backend server will operate directly on top of it. The user database will contain credentials and basic information of each user, such as names and emails. The receipt database will store receipt information identified by the recognition engine. Attributes will include but not limited to receipt ID, product names and prices. The report database will have expense reports requested by the user in the past, this is meant to accelerate requests in the case where the user asks for the same report more than once.

1.3.5 Receipt Recognition Engine

The receipt recognition engine is aimed at recognizing information from receipts. The dataset for model development contains images of scanned Roman alphabets and digits, which is separated to training and testing datasets that contain 80% and 20% samples from the original dataset respectively. Labels of samples are the corresponding letters or digits. Pre-processing operations including image brightening, binarizing and segmentation are applied to training and testing samples to filter out useless features from those images. An artificial neural network is used for training a model, which is then tested by the testing set. The model is finally encapsulated as the recognition engine if the testing result reaches the expectation, which is utilized by the service provider in the backend server.

1.3.6 Cloud

The cloud block contains the storage and computing services that are needed to be obtained through cloud sources. In this project, Amazon S3 storage will be used for storing training and testing datasets for the model development process of the receipt recognition engine. Amazon EC2 GPU instances are to be used for the model training process to significantly increase the training efficiency.

2. Project Specifications

The project specifications are categorized as functional and nonfunctional specifications. Functional specifications describe the expected functionalities of the project, while the non-functional specifications provide constraints of the project design. Each type of specifications is comprised of attributes: specification, description, and necessity. Specification attribute gives brief information about the specification. Description attribute defines the requirement of the specification in detail with numerical values. Necessity attribute classified each specification as essential or non-essential. The essential specification must be met for the success of the project, while non-essential ones are not critical but would improve the overall performance of the project.

2.1 Functional Specifications

Table 1 below lists functional specifications that must be met for the success of the project.

Table 1: Functional Specifications

Specification	**Description**	**Necessity**
Recognition Accuracy	The recognition engine should recognize product name, product unit price and total price from paper or email receipts. The recognition accuracy is expected to be over 80%, while the precision and recall are expected to be 80%.	Essential
Image Processing	The camera module should provide functionalities to crop an image and stitch partial receipt images into one.	Essential
Receipt Boundary Detection	The camera should auto-detect and draw the boundary of a receipt for guiding users to capture the receipt.	Non-Essential
Expense Report Generation	The application will provide the functionality to generate expense reports of an arbitrary period containing prices of different products.	Essential
Expense Comparison	The application will allow users to compare expenditures over different periods of time and visualize them via charts.	Essential
Receipt Information Editability	Users should be able to edit and save names, unit prices and total prices of recognized products.	Essential
Handwriting Content Recognition	The recognition engine is expected to recognize the handwriting contents on the receipts (eg. written tips amount).	Non-Essential
GPU Memory Requirement	The memory requirement for model training processes on the GPU is expected to be smaller than 64 GB.	Essential

Character Segmentation Accuracy	The segmentation method used in the recognition engine is expected to separate at least 80% characters and digits on the receipts to regions for the recognition process.	Essential
Training Data Properties	The model is expected to reach the required accuracy with less than 200,000 training images with a size that is not larger than 200 * 200 pixels.	Essential

2.2 Non-Functional Specifications

Table 2 below lists non-functional specifications that serve as constraints of the project.

Table 2: Non-Functional Specifications

Specification	**Description**	**Necessity**
Data Security	User credentials and sensitive information need to be tokenized and transited in a secured network channel such as TLS or SSL. A permission system should be built to ensure the user does not have access to other people's data.	Essential
Application Screen Adaption	The user interface of the application should be adaptive to different screen resolutions ranges from 240 x 320 to 1440 x 2560 px with screen density from 120 to 640 dpi. [x]	Essential
Server Resilience and Stability	The application should be resilient to malicious behavior and tolerant to transient failures to achieve 99% uptime.	Essential
Application System Compatibility	The application should support at least Android API level 24 (Android OS 7.0 Nougat).	Essential
Recognition Efficiency	The recognition process for interpreting one receipt is expected to be shorter than 10 seconds.	Essential
Android User Interface Design	The design of the user interface should follow Android Material Design guidelines.	Non-Essential
Application Accessibility	The application should be compatible with Android screen reader tools such as TalkBack to provide accessibility to visually impaired users.	Non-Essential

3. Detailed Design

3.1 Mobile Client Design

The section discusses software architectures implemented for the network and UI modules in the mobile client. Due to the border function of the Android activity files in the UI module, the UI module selects an alternative architecture than that in the network module.

3.1.1 Network Module

The network module communicates with the back-end server to request or retrieve JavaScript Object Notation (JSON) format data via Representational State Transfer (REST) Application Protocol Interface (API). The module was implemented on Model-View-Controller (MVC) software architecture for achieving decoupling and maintenance. The mobile client subsystem uses Retrofit, a third-party library designed to build a type-safe HTTP client for Android.

Figure 2 below shows the MVC architecture that was implemented for the network module.

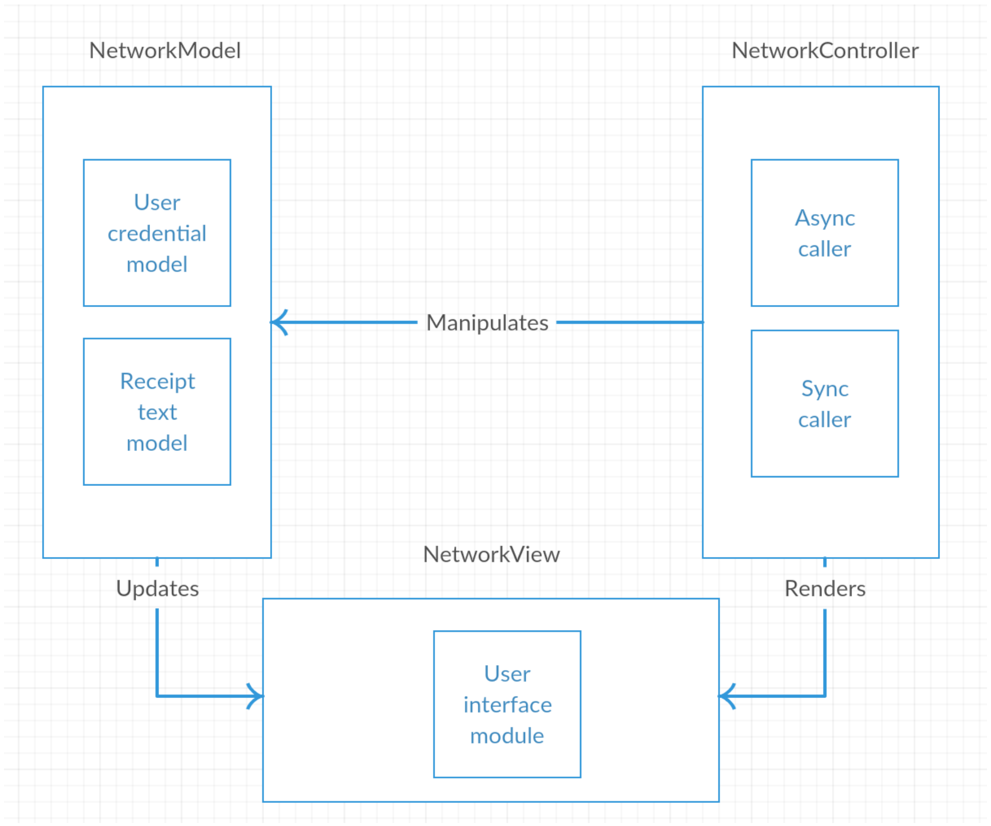


Figure 2: MCV architecture of network module

From Figure 2 above, the NetworkModel defines the behavior of the user credential and receipt text models. The user credential model, which stores the credential data, is used for granting the users for accessing the application by identification and authentication. The receipt text model, which is corresponding to receipt text data, is used for receipt image recognition, purchase history presentation, expense report generation.

The NetworkModel creates Plain Old Java Object (POJO) as containers for user credentials and receipt related models. With Retrofit supporting the conversion between POJO and JSON format, the

NetworkController does not need to worry about formatting issues between the server and the client during a conversation. When sending or receiving a request, the NetworkController manipulates the NetworkModel by constructing or parsing POJOs into or from the network request callers, respectively.

By accessing the POJO from the NetworkModel, the data presented in NetworkView could be updated after each network request.

The NetworkController provides asynchronous and synchronous callers for types of network requests. The passing and returning arguments for the callers are POJOs defined from the NetworkModel during manipulation. As mentioned above, since Retrofit plays as a middleware between the client and the server, the caller does not concern about formatting conversion.

The NetworkView is comprised of User Interface (UI) and Camera modules. Its primary duty is organizing and displaying data retrieving from the NetworkModel. The NetworkController is triggered to ask the NetworkView to render the user interface.

3.1.2 User Interface Module

The user interface module presents graphical interfaces that interact with users through gestures. Due to the specific functionality of activity files which can control and render the graphics components, Model-View-ViewModel (MVVM) is an alternative software architecture to be implemented for the mobile client.

Figure 3 below is the comparison of MVC and MVVM.

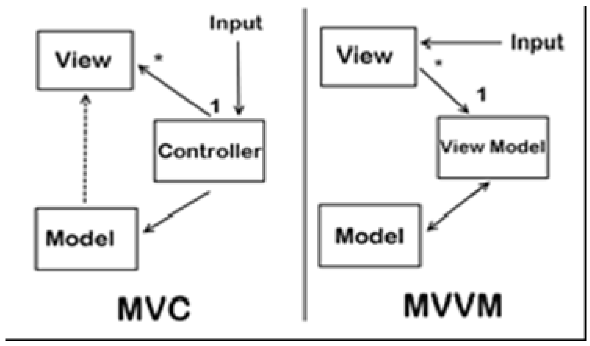


Figure 3: MVC Vs. MVVM [2]

In MVVM architecture, the responsibility of the model is similar to that of MVC. The model stores data and business logic, which determines the behaviors of the UI components. For example, the receipt list model should be capable of refreshing the list when it detects any changes. The variables inside the model are declared as observable type, which would be triggered automatically when its value changes in the view model. [3]

The view model interacts with the model and prepares the observable data for the view. For example, it binds the receipt list data into the receipt list view, when the GUI is rendered, the view displays the bind data. The view model defines the reactions and functions for the view when events happen. For example, the receipt list should be refreshed when the items change. [3]

The view is also similar to that of MVC. The view is only responsible for displaying the interactive user screen. It only cares about the color, font size, and padding of the receipt list item. The business logic of events handling functions is decoupled entirely to the model. [3]

The main difference between MVC and MVVM refers to the functionality between controller and view model. As figure 2 shows above, the system input enters the controller, which manages and triggers the other two components. However, the majority of the data comes from the user interaction with the screen, which is the view component. Using MVC in the UI module may require redirecting the input from the view to the controller, that would lower the decoupling. MVVM is more adaptive for the UI module as the input enters the view. In terms of complexity, it is mandatory to import an Android lifecycle ViewModel library to establish an MVVM structure. This new library may add time consumption for setting up and learning about a new library. MVC does not require a new library, and it is easy to implement. As mentioned before, MVVM matches more naturally with the UI module in terms of the input entry. It is more maintainable for future development. In MVC, the input redirection may lead to unknown problems as the application iterates.

According to the discussion above, a decision matrix comes up based on the aspects of decoupling, complexity, and maintenance.

Table 3: Decision matrix for MVC Vs. MVVM

	Decoupling	Complexity	Maintenance	Total
MVC	0	1	0	1
MVVM	1	0	1	2

From table 3, MVVM is the more appropriate architecture to be implemented in the mobile client.

3.1.3 Design Justification

The mobile client uses OpenCV build-in functions to crop and stitch captured images. After the image manipulation, the mobile client calls the Highcharts APIs to visualize receipts comparison over different periods and communicates with the back-end for changes of receipts and users' information. This design pattern for mobile client follows the Android framework for screen adaption and system compatibility and sticks to Material Design Document to design the user interface.

3.2 Receipt Recognition Engine Design

The design and implementation of the receipt recognition engine are described in this section. The recognition engine consists of an image pre-processing program to enhance the quality of the input photo of receipts, an image segmentation program to segment the input photo to characters and digits, and a text recognition program to transfer the characters and digits on the input photo to texts.

Convolutional neural network (CNN), an artificial neural network that is commonly used for computer vision projects and is frequently used in this project, is introduced in the first chapter. The three main programs for building the receipt recognition engine are described in the following chapters.

3.2.1 Introduction to Convolutional Neural Network

Convolutional Neural Network is a feed-forward artificial neural network uses convolutional operations for extracting most representative features from each region of the input data and applies mathematical computations on the features to perform tasks. A typical CNN consists of multiple convolutional layers for extracting features from each region of the input data, each of which is followed by a pooling layer to speed up the computation by reducing the dimension of the input data. Several fully-connected layers are added at the end of the network to classify the extracted features. [4] The structure of a typical CNN is shown in Figure 4.

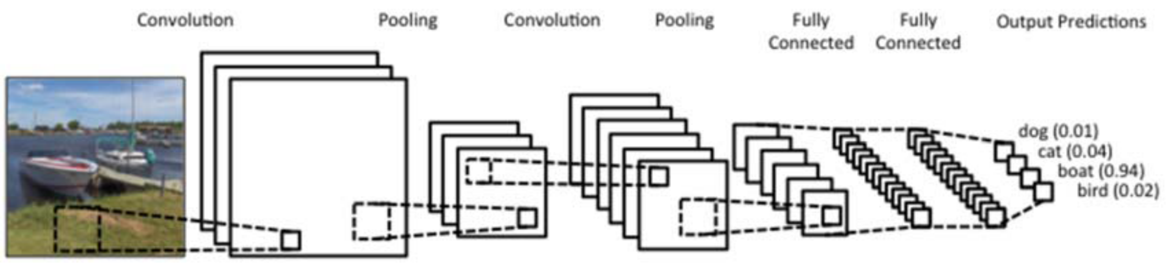


Figure 4: Structure of a Convolutional Neural Network [5]

The methods for a CNN to extract features from input data is determined by the number of neural, the size of the receptive field and the size of the convolutional kernel of neurons at each convolutional layer. A neuron in a convolutional layer is a group of mathematical functions that simulates the performance of neurons in a biological neural network. Each neuron has a particular receptive field, which is the range of functionality of the neuron when extracting features from the input data. The convolutional kernel is defined as the basic unit of a convolutional operation, which is a matrix to be applied to each region of the input data in convolutional operations. Weights are given to each element of a convolutional kernel. For a typical convolutional layer, the sizes of the convolutional kernel and receptive field of a neuron are the same. [4] An example of the convolutional operation is given in Figure 5. I, K and I*K in Figure 5 represent the input data, the convolutional kernel, and the output matrix respectively.

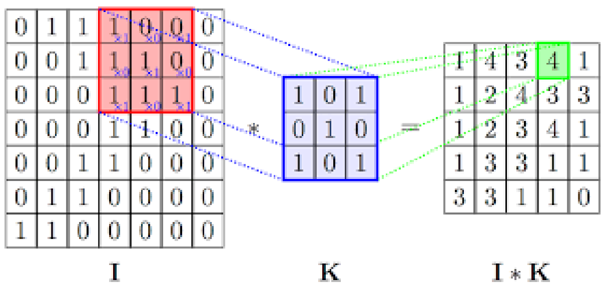


Figure 5: Example of Convolutional Operation [6]

3.2.2 Image Pre-Processing

A photo of receipts taken by a phone or a scanned copy of a receipt is considered the input of the receipt recognition engine. To increase the accuracy of the recognition results, the quality of the input photo is enhanced in the image pre-processing procedures. Image pre-processing for this project consists of image rotation, receipt detection, affine transformation, noise reduction and binarization.

Firstly, the input images are rotated to horizontally align the receipt in the image. The receipt is a rectangle and in white or similar colors in most cases, which differentiates the receipt from the surrounding environment, so the program first searches for bright and rectangular areas. Border detection techniques can be used to localize borders of the area and image is rotated until the detected top, bottom and side borders of the area are horizontally and vertically aligned respectively. According to previous experiences, Canny Edge Detection provided by OpenCV has the most acceptable performance on related tasks, which is utilized in the receipt aligning process. [7]

Secondly, an affine transformation is applied to images if needed. Affine transformation is a mapping process in an affine space that preserves straight lines, which can be used to map pixels in a parallelogram into a rectangle. [8] A bright parallelogram detected in the input image can be identified as a distorted receipt, which can be transformed into the front view of the receipt by applying an affine transformation.

Thirdly, an object detection application is applied to the rotated image for detecting the receipt. This process aims to identify if the bright and rectangular area in the input image is a receipt. A convolutional neural network is used for model training. The training data for this model is a number of images with or without receipts, which has labels that contain the existence of the receipt and the coordinate information of the receipt if existing. The expected output of images processed by the model contains rectangles that localize the receipt in the image if existing. The image is then cropped to only remain the rectangle that contains the receipt.

For images containing noises, proper noise reduction methods are applied. A band-pass filter can be applied to filter out noises with different pixel values with the text information and the background. For the noise pixels with similar values to the text information, an auto-encoder can be applied to reduce the noise. Auto-encoder is a type of neural network that has the same input and output size. The most representative features of the input are extracted by applying convolutional operations on the input data in the encoding process. The features extracted from the encoding process are then re-organized and upsampled to form the output data that is with the same size as the input data. [9] A sample auto-encoder is indicated in Figure 6.

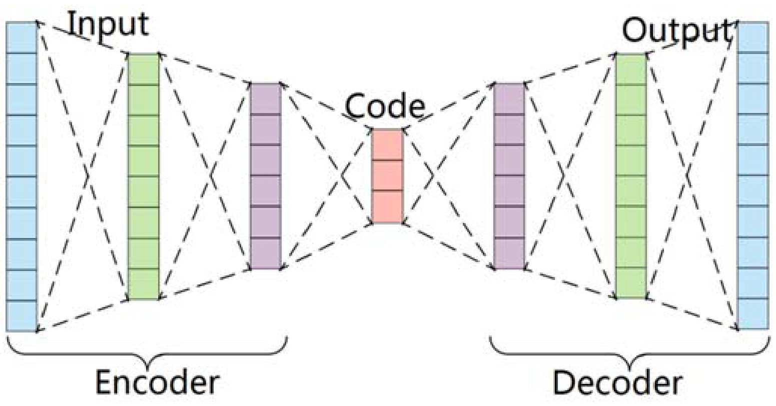


Figure 6: Structure of a Typical Auto-Encoder [10]

Binarization is the process to transfer grey images to binary images. Each pixel of the input image is either set to be 0, the minimum value for a pixel, when its original value is smaller than a particular threshold or set to be 255, the maximum value for a pixel, when the original value of the pixel is greater than the threshold. With an appropriate binarization threshold, the resulting image is expected to have the text information in black and the background in white, which is expected to promote the image segmentation processes since the contrast between the text information and the background of receipt is enhanced. An example of binarization is indicated in Figure 7.

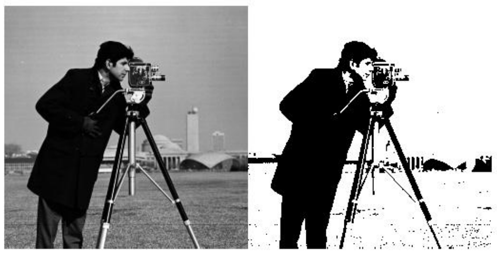


Figure 7: Example of Binarization [11]

3.2.3 Image Segmentation

The image segmentation process aims at detecting the characters and digits from the receipt image and segmenting the image to regions that contain a single character or digit for the text recognition model to transfer those regions into text format separately. Two main methods are considered for the image segmentation process. The first method is using a grid for separating characters and the second method is to find connected components from the input image by using border detection methods of OpenCV. All methods are tested and analyzed in this chapter for selecting the optimal solution for the image segmentation task.

Since the width and height of characters on a receipt are the same, the images can be equally divided into regions by using a grid. Each region is supposed to contain one character. This method results in an acceptable performance on flat receipt without foldings. However, for distorted images or folded receipts, it is impossible for the grid to perfectly fit the characters on the receipt. In addition, if the image is not properly cropped in the pre-processing procedures, the grid may be dislocated when fitting characters on the receipt. An example of the usage of a grid on image segmentation is provided in Figure 8.

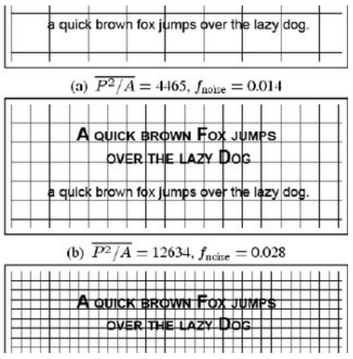

a quick brown fox jumps over the lazy dog.
(a) $\overline{P^2/A} = 4465$, $f_{\text{noise}} = 0.014$
A QUICK BROWN FOX JUMPS OVER THE LAZY DOG
a quick brown fox jumps over the lazy dog.
(b) $\overline{P^2/A} = 12634$, $f_{\text{noise}} = 0.028$
A QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Figure 8: Example for Text Segmentation with Grid [12]

The segmentation process with the border detection method is similar to the image rotation process introduced in chapter 3.2.2. Canny Edge Detection method is applied to detect the borders of each character to find the contour of characters and then create a rectangular region that contains only one character based on the detected contours of characters. [7] Then, the k-nearest neighbor algorithm is applied to the regions for combining characters into words. This method has an acceptable performance on scanned images without detectable noise while the performance is negatively affected when inputting a noisy image taken by mobile devices.

A hundred images containing receipts are randomly selected for testing the performance of the two methods above. The accuracy is measured as the percentage of characters that are successfully segmented into regions. Each test case is executed three times and the final result for a test case is calculated as the average performance over the three executions. The distribution of performance is shown in Figure 9.

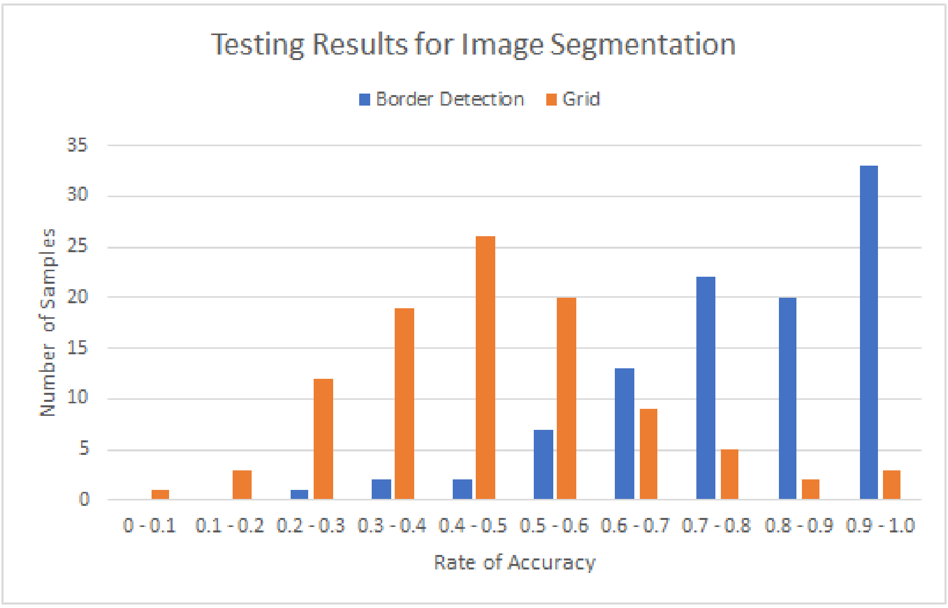


Figure 9: Testing Results for Image Segmentation

From the testing results above, it is obvious that the border detection methods have a significantly better performance than the grid method. So, the border detection method is selected as the image segmentation method for recognition engine development.

3.2.4 Training Data and Text Recognition

The training data consists of images that contain a single character or digit in the fonts that are most commonly used by receipts. According to the functional specification, the neural network is expected to train a model that reaches the required accuracy with no more than 200,000 training samples with a size smaller than 200 * 200 pixels, so we set the original size of training set as 200,000 and normalize samples in the training set to 100 * 70 pixels initially. The dataset consists of characters in the most common fonts that are used by receipts, which are directly extracted from a number of receipt images. The label of the dataset is the character contained in each image, which is one-hot encoded. In the one-hot encoding process, each ASCII symbols is considered a category and each label of the dataset is converted to an array with a length 95 (the number of ASCII symbols). The element in the label array that represents the category the associated image belongs to is set to be 1 while other elements in the array are set to be 0. The original dataset is randomly shuffled before dividing into a training dataset, a validation dataset and a testing dataset that contain 80%, 10% and 10% of the original dataset to ensure that all of the training, validation, and testing datasets are in the same distribution.

Two main methods can be applied to the training dataset for the training process of the text recognition model. A decision tree can be applied to classify the characters, which is considered the first solution for the text recognition task. The second method is using a convolutional neural network to train an image classification model for characters. The decision tree method is easy to implement and the requirement on the size of the training dataset is relatively small. However, since there are more than 90 categories in the text recognition task, the number of false positives and false negatives will be higher than the expectation. The solution with a convolutional neural network may lead to high computational costs and long training time, and the requirement on the size of training dataset is expected to be higher than using a decision tree. But comparing with using a decision tree, the convolutional neural network has a better and more consistent performance on categorizing data into a large number of categories and the convolutional neural network has a relatively better performance on processing image data.

A decision matrix is provided in Table 4 for analysing the properties of the two methods described above. Based on the final scores, the convolutional neural network method is selected as the solution for building up the text recognition model.

Table 4: Weighted Decision Matrix

Criteria	Weight	Decision Tree	CNN
Dataset Requirement	30	30	20
Speed	10	10	5
Computational Cost	10	10	5
Accuracy	50	20	50
Total	100	70	80

3.2.5 Design Justification

One of the major functionalities is recognizing characters on receipts. In the project, Convolutional Neural Network (CNN) is applied to train the character classifier. CNN takes a matrix as the input and applies convolutional operations at each layer to extract features from images for a softmax classifier to identify the type of the features and the whole image, which makes the classifying of characters accurate and efficient. According to the experimental results, Convolutional Neural Networks reach the highest performance on image classification and is expected to reach a precision and recall higher than 80%.

At the same time, edge detection with the Canny function and the K-nearest neighbour algorithms is applied to the scanned receipts. The K-nearest neighbour algorithm with proper parameter settings are available to identify words and lines of words based on the distances between characters. Also, the Canny edge detector demonstrates a high accuracy on edge detecting tasks on binarized images. In summary, the edge detection method is able to efficiently separate at least 80% of characters and digits and detect the borders of the receipt in the input image.

Also, the training and validating datasets used in the process are limited to less than 200,000 images with sizes equal to or smaller than 200*200 pixels. The requirements aim at reducing the computational cost during the training and inference processes and minimize the GPU memory requirements. According to the experimental results, when following these requirements, the GPU memory usage is smaller than 64 GB for the whole time. At the same time, since each image in the datasets contains a single character, limiting the size of images to be smaller than 200*200 pixels will not affect the performance of the model and the size of the dataset, 200,000, is sufficient for the training process from the experimental results. In addition, limiting the size of images significantly reduces the inference time for receipt text recognition, which highly contributes to enhancing the recognition efficiency.

3.3 Server Design

This section contains the analysis of different server frameworks and detailed API design. The server will be deployed using AWS Elastic Beanstalk with blue/green deployment to achieve scalability and minimal downtime.

3.3.1 Server Framework

The server is used for serving client requests, including saving and retrieving receipt information, generating reports and connecting to the receipt recognition engine. To efficiently process client requests, choosing the right API framework is extremely important. The group members are most experienced with Node.js/Express and Ruby on Rails. Those two frameworks will be evaluated based on performance, scalability, and development difficulty. Since performance and scalability are much more important, performance and scalability are given weights of 4 respectively, and development is given a weight of 2. During the analysis, each solution will be given a score from 1 to 10 according to how well it meets each criterion.

First of all, Node.js has better performance than ruby on rails. Node.js is built on top of google chrome V8 engine, which is written in C++ and highly optimized. This makes Node.js more than 20 times faster than Ruby on Rails. Therefore, Node.js is given a score of 10 and Ruby on Rail is given a score of 5. Additionally, Node.js is more scalable than ruby on rails. Node.js has a single-thread, non-blocking, event-driven architecture, which allows Node.js handles multiple concurrent requests efficiently. Whereas, Ruby on Rails has a blocking I/O nature and uses limited threads to handle multiple requests. It is wasting CPU time and memory while waiting on I/O operations and performing context switching. Therefore, Node.js scores a 10 on scalability and Ruby on Rails only scores a 5.

Last but not least, Ruby on rails has a faster development time because it is a complete framework. It has more out of box tools, including database connection and migration, which makes it easier to build a CRUD application. Whereas, in Node.js, it takes more time to find the right module and integrate the module into the application. Therefore, the scores of Node.js and Ruby on Rails on development are 6 and 10.

Table 5: Weighted Decision Matrix

		Node.js		Ruby on Rails	
Criteria	Weight	Score	Weighted	Score	Weighted
Performance	4	10	40	5	20
Scalability	4	10	40	5	20
Development Difficulty	2	6	12	10	20
Total	10	26	92	20	60
Final Score			9.2		6

As shown in Table 5, Node.js has a total score of 9.2 and Ruby on Rails has a total score of 6. It is obvious that Node.js has a higher score; therefore, Node.js is the better option.

3.3.2 API Design

This section illustrates the design of APIs using HTTP as the transport layer protocol. All access to APIs is required to contain an OAuth header to prevent hacking attempts. Each set of endpoints are designed with the constraints of REST architecture in mind to maximize readability and scalability.

Table 6 below shows the set of user APIs which support the basic user flow such as user creation and the update of essential information including email, name, and password.

Table 6: User API

HTTP Verb	Endpoint	Description	Sample Response
POST	/user	Create a new user given name, email, and password.	{ id: new_user_id }

GET	/user/{user_id}	Retrieve information of a user given their ID.	{ id: user_id name: xxx }
PUT	/user/{user_id}	Update user information given their id.	N/A

Table 7 illustrates a list of receipt APIs which enables displaying extracted information from various receipts and allows users to update receipt details if the recognition engine fails to pull out critical texts correctly. Table 8 contains product APIs, which are primarily used by receipt APIs internally to store or load product information.

Table 7: Receipt API

HTTP Verb	Endpoint	Description	Sample Response
POST	/receipt	Create a new receipt with a list of product information	{ receipt_id: 2 }
GET	/receipt?user_id={user_id}	Retrieve all receipts of a particular user.	{ receipts: [...] }
DELETE	/receipt/{receipt_id}	Delete a receipt given receipt id.	N/A
GET	/receipt/{receipt_id}	Retrieve details of a particular receipt	{ receipt_id:1, user_id: 10, products:[....] }
PUT	/receipt/{receipt_id}	Update product information within a particular receipt	N/A

Table 8: Product API

HTTP Verb	Endpoint	Description	Sample Response
GET	/product?product_id={product_id}&user_id={user_id}	Retrieve a particular product of a given user.	{ product_id: 3, price: 100, name: "Chocolate Bar", ... }
POST	/product?user_id={user_id}	Retrieve all products of a given user.	{ products: [...] }
PUT	/product	Create a new product given name, price, quantity, user, etc.	{ product_id: 2 }

Table 9 demonstrates how Report APIs are implemented. The first API contains the business logic of computing how much money the user has spent during an arbitrary period of time. The Mobile Client will call it multiple times to compare expenditures over different periods of time as requested by users and visualize them via charts. Therefore, various actions, including the creation of expense reports of arbitrary date range and the deletion of existing one are well supported.

Table 9: Report API

HTTP Verb	Endpoint	Description	Sample Response
POST	/report?user_id={user_id}&start={start_date}&end={end_date}	Generate an expense report of arbitrary duration for a given user.	{ report_id: 100 }
GET	/report/{report_id}?user_id={user_id}	Retrieve an existing report given user id and report id.	{ report_id: 100, start: "2018-01-03", end: "2018-02-03", ... }
DELETE	/report/{report_id}?user_id={user_id}	Delete an existing report given user id and report id.	N/A

3.3.2 Design Justification

The backend server is carefully designed to meet the project specifications. Node.js is chosen to be the server framework to maximize server performance and scalability. To ensure data security, an authentication service is designed to encrypt the user password and use the OAuth header to prevent hacking attempts. Additionally, the Receipt and Product APIs are designed to allow users to edit receipt information. Report API is designed to generate expense reports and expense comparison. In all the API endpoints, the user inputs are validated, and the network failures are correctly handled to achieve better server resilience. Moreover, to improve service stability, the backend server is deployed on the Heroku Cloud Application Platform. Multiple service instances are used to distribute workload and reduce single points of failure. In conclusion, the server design is closely aligned with project specifications and functionalities.

3.4 Database Design

Database subsystem is a crucial part of the project as all of the users, user-related receipts and purchase history information will be stored in databases for the use of generating individual bookkeeping reports. This section of the report will examine the database design in details by analyzing the database structure from its Entity-Relation Model. With the considerations of project management, specification and database structure, a set of requirements and weighted criteria will be introduced for the database selection. Alternative solutions for database selection will be evaluated according to the requirements based on their weighted criteria in a Decision Matrix to find out the most suitable database for this project.

3.4.1 Database Selection

The actual choice of the database management system has a significant influence on application performance and maintainability. Most team members are proficient in MySQL and PostgreSQL, the two most popular choices of the relational database management system (RDBMS) in the market. To come up with the preferred choice, two RDBMSs are examined based on a list of criteria:

performance, scalability and built-in features. Since the performance and scalability of the database are major factors dictating the theoretical limit of how fast the entire application can run, they are both assigned a weight of 4 out of 10. A larger number of built-in features enables faster development; however, it hardly affects the performance of the application and thus received a weight of 2.

When comparing the scalability between MySQL and PostgreSQL, both MySQL and PostgreSQL has the master to slave replication. However, MySQL offers replication from master to master, which can be proven useful when scaling out reads. [13] This shows that MySQL has better scalability when compared to PostgreSQL, which results in a 9 (out of 10) for MySQL and 5 for PostgreSQL in the scalability criterion.

On the memory perspective, MySQL uses a thread pool to handle requests which make MySQL light-weight and memory friendly. On the other hand, Postgres forks off a child process to establish a connection, and it can take up to 10 MB per connection. The memory pressure is more significant compared to MySQL's thread-per-connection model, where the stack size of a thread is at 256KB on 64-bit platforms. [14] This shows that MySQL has better memory management when compared to PostgreSQL, which results in an 8 (out of 10) for MySQL and 4 for PostgreSQL in the memory criterion.

Finally, when looking at the SQL features in MySQL and PostgreSQL, MySQL provides a reasonable set of features, such as built-in SQL functions, that follow the 80/20 rule: It has the 20 percent of SQL capabilities that are needed for 80 percent of database applications. Developers of simple applications can live without the remaining features. PostgreSQL, differently, provides more features than MySQL. These include more SQL functions, server-side procedural languages, and sophisticated methods for date manipulation. [15] This can save developers lots of time during the development cycle, which is a plus. This shows that PostgreSQL has more features than MySQL, which results in a 9 (out of 10) for PostgreSQL and 4 for MySQL in the features criterion. All the above values will be calculated in a Decision Matrix shown in Table 10.

Table 10: Decision Matrix for Database Selection

Criterion	Weight	MySQL	PostgreSQL
Scalability	4	9	5
Memory	4	8	4
Features	2	4	9
Total	10	76	54

From the Decision Matrix, it is clear that MySQL is the winner which will be selected as the database for the project.

3.4.2 Database Structure

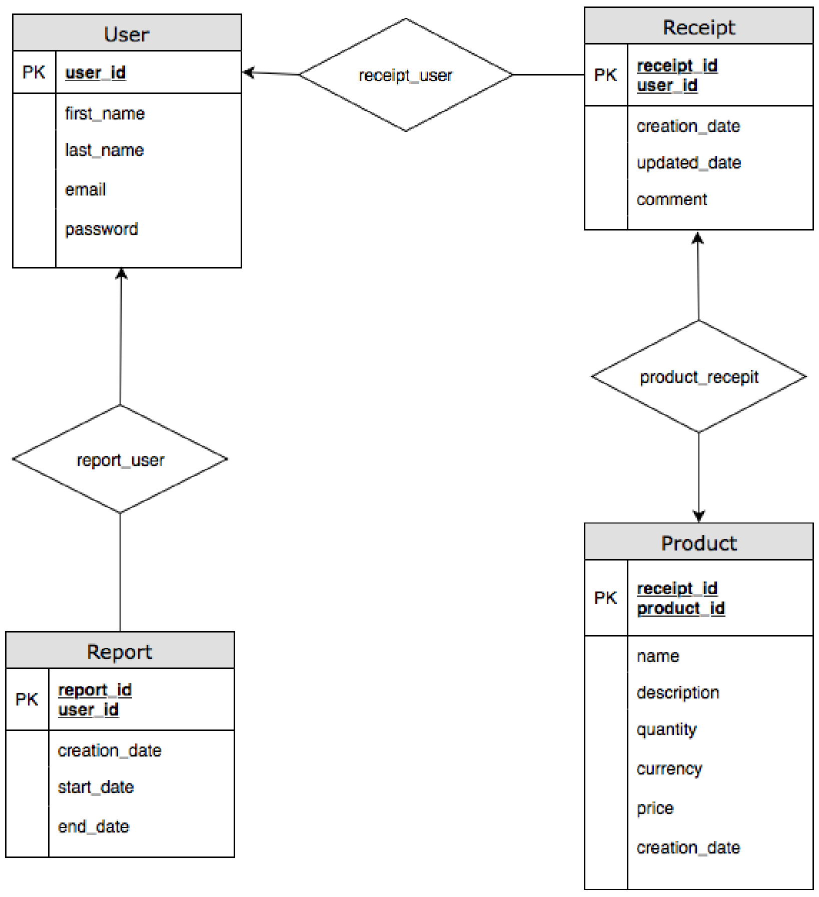


Figure 10: Entity-Relation Model for Database Structure

All the information gathered from users and receipts will be distributed into four tables in the database for this project. This includes “User”, “Receipt”, “Product”, and “Report” tables.

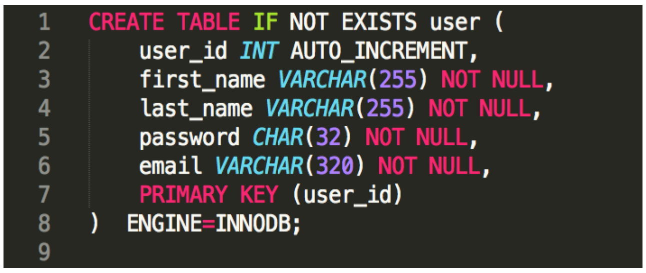
The “User” table is used to store general information for registered users such as their first, last name, email address and their password. A user identification number will be assigned to each registered user, and it will act as the primary key for the “User” table for indexing purposes.

Whenever a user scans one of his/her receipts, the application records the creation date of that receipt. If the user requested a manual update to the details of the receipt or general comment to the receipt, then those updates and comments will be stored to database accompanied with a receipt identification number into the “Receipt” table. The “Receipt” table use the receipt ID and its associated user’s ID to form a composite primary key. This user ID will act as a foreign key to the “User” table to reference user’s information.

Once the receipt data is set, all of the purchased items on the receipt will be interpreted by the receipt recognition engine and the extracted products will have their corresponding information including the product name, description, quantity, price, currency creation date stored in the “Product” Table. Each purchased product will be given a unique product identification number to keep tracks all the product. The receipt ID in the “Product” Table is a foreign key for the “Product” Table that references the receipt information in the “Receipt” table.

Once a period, the application will generate an expense report of the users. This report requires the start date and end date of that given period, which is the reason why the start date and end date needs to be stored in the "Report" table. When a report is created, creation date and a report identification number will be recorded in the "Report" table. This report ID will be the primary key for the "Report" table, and the user ID attribute will be used as a foreign key to the "User" table to reference user's information.

Since a user can have multiple receipts and multiple reports, the relationship between the "User" table and "Report" table or "Receipt" table should be a one-to-many relationship. Each product has a unique product ID with the reference of the unique receipt ID, the relationship between the "Product" table and "Receipt" table should be a one-to-one relationship. All these entity and relation information for the database structure will be summarized with the Entity-Relation Model in Figure 10.



```
CREATE TABLE IF NOT EXISTS user (
    user_id INT AUTO_INCREMENT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    password CHAR(32) NOT NULL,
    email VARCHAR(320) NOT NULL,
    PRIMARY KEY (user_id)
)  ENGINE=INNODB;
```

Figure 11: Code Snippet for the User Table Creation

A sample Code Snippet for creating the "User" Table according to the ER Model is shown in Figure 11. The creation of other tables follows a similar fashion.

3.4.3 Design Justification

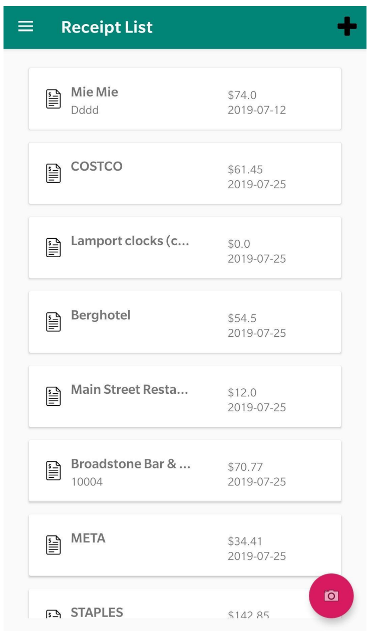
One of the major functionalities of the application is to allow users to visualize their expense history in an organized fashion. The Entity-Relation Model for the Database is designed in the way that it stores product information for the application to gather and analyze the user data. With the user data, the application is capable of comparing expenses and displaying the results on the user interface. There is also a dedicated "Report" table to store the necessary information to generate a detailed expenditure report for an arbitrary period. These thoughtful designs on Entity-Relation Model are used to satisfy the project specification.

4. Prototype Data

4.1 Prototype Images

The figures below are the screenshots of the mobile client, ReceiptIt. Figure 12 demonstrates receipt history from a user. Each block in the list outlines the name of the receipt, total expense, purchased date. It also shows the postal code where the receipt is from if the data is available. In the top right corner, the plus button provides two ways to create a new receipt: manually generate or scan a receipt. The camera button in the bottom right corner supports the user to scan a receipt directly from the device's camera. When the top left button is clicked, the right figure pops up.

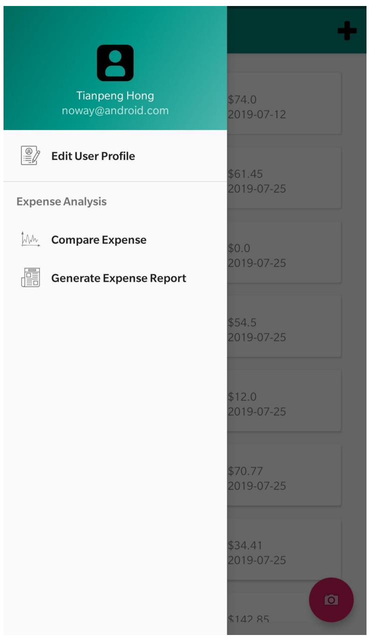
Figure 13 demonstrates a navigation drawer that presents general information of the user and provides several buttons. Users could change their username, account, and password via the Edit User Profile button. The Compare Expense button concludes all purchased products over a specific period and their total cost. The Generate Expense Report button generates a pie chart composed of expenditure over different periods.



Receipt List

Mie Mie Dddd	$74.0 2019-07-12
COSTCO	$61.45 2019-07-25
Lamport clocks (c...	$0.0 2019-07-25
Berghotel	$54.5 2019-07-25
Main Street Resta...	$12.0 2019-07-25
Broadstone Bar & ... 10004	$70.77 2019-07-25
META	$34.41 2019-07-25
STAPLES	$142.85

Figure 12: Receipt History



Tianpeng Hong
noway@android.com

Edit User Profile

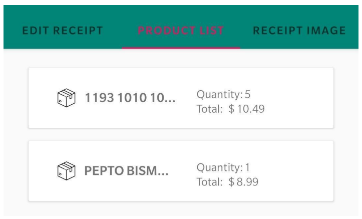
Expense Analysis

Compare Expense

Generate Expense Report

Figure 13: Main Screen

When a receipt item is clicked, the client shows detailed information about the receipt as Figure 14 shows. It includes the general receipt information, products under the receipt, and the receipt image if available. In the product list, each block displays the product name, quantity, and the total cost.



1193 1010 10... Quantity: 5 Total: $ 10.49

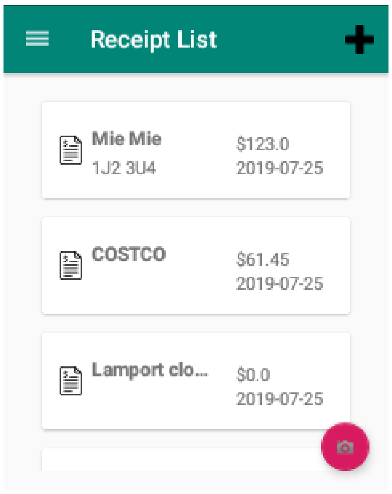
PEPTO BISM... Quantity: 1 Total: $ 8.99

Figure 14: Product List

4.2 Prototype Discussions

4.2.1 Application Screen Adaption

The figures below are the screenshots of the mobile client running in two devices with different screens. As Figure 15 demonstrates, the client is compatible with running in a display with 240*320 resolution and 120 dpi screen density. The target device's screen of the client is shown in Figure 16, with 1440*2560 resolution and 640 dpi screen density.

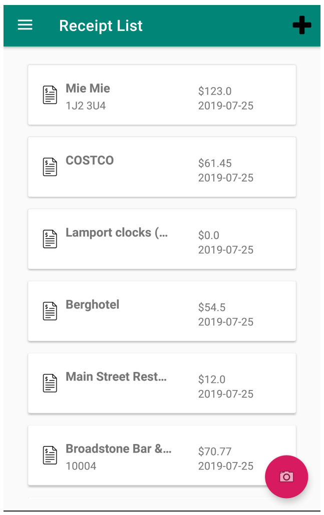


Mie Mie 1J2 3U4 $123.0 2019-07-25

COSTCO $61.45 2019-07-25

Lamport clo... $0.0 2019-07-25

*Figure 15: Screen with 240*320 Resolution*



Mie Mie 1J2 3U4 $123.0 2019-07-25

COSTCO $61.45 2019-07-25

Lamport clocks (... $0.0 2019-07-25

Berghotel $54.5 2019-07-25

Main Street Rest... $12.0 2019-07-25

Broadstone Bar &... 10004 $70.77 2019-07-25

*Figure 16: Screen with 1440*2560 Resolution*

4.2.2 Server Resiliency and Stability

The easiest way to visualize the server resilience and stability is by looking at the events in the server activity chart. The application is deployed on Heroku, which provides functionality under the Metrics tab for checking the server activity. The server activity chart for the application is shown in Figure 17.

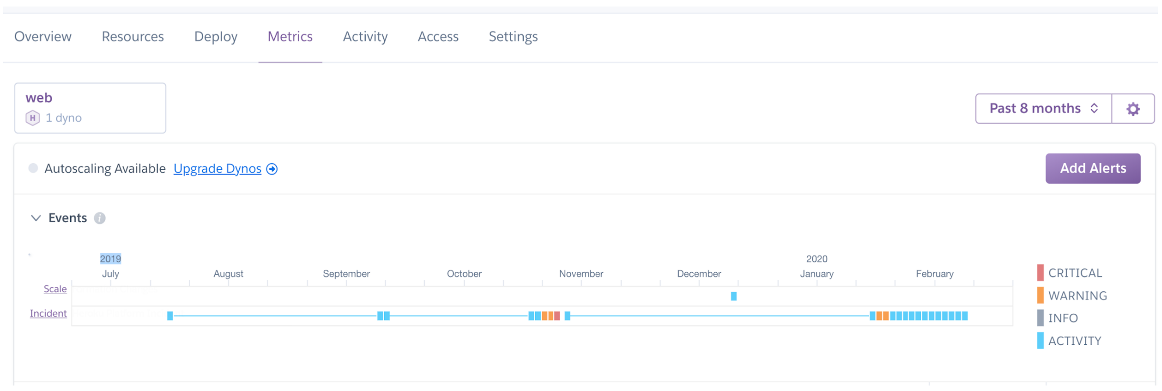


Figure 17: Server Activity

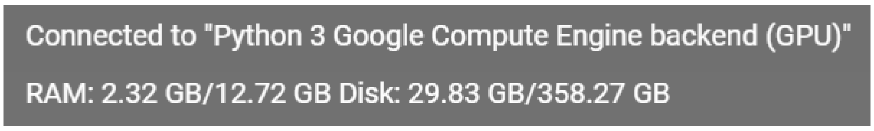
Three major events could happen on the server on a given day, and they are under different colour codes. Blue tile means there is an activity burst on that given day, orange tile means there is a warning burst on that given day, and red tile means there is a server failure on that given day. The time between an activity tile and a warning tile is considered as uptime because there is no actual failure. The time between a warning tile and a failure tile is regarded as downtime because failure is confirmed. The time between a failure tile and an activity tile is considered as downtime since the uptime is measured from the moment when an activity event is detected. From the above image, two critical moments need to be considered. First is the region from October 15th, 2019 to October 28th, 2019 and the second is the region from January 14th, 2020 to January 18th, 2020. There is an apparent failure that happened from 22th to 24th. This provides insight into the stability of the server. The percentage of uptime can be calculated by dividing the amount of downtime with the total number of days monitored. If there is a 2-day-downtime in seven and a half months, the uptime percentage for this server is 99.1%. For the second critical region, the server changed from an activity burst to a warning burst and then back to normal. This shows that the server is resilient when encountering warnings and can survive from potential server errors. The excellent percentage uptime and resilience of the server has met the requirements specified in the Non-Functional Specifications.

4.2.3 Training Data Properties

In the experiments, images in datasets are reshaped to 128*128 pixels, which is smaller than 200*200 pixels. At the same time, the size of the training dataset is 50,596, which is significantly smaller than 200,200. Since all essential requirements have been met by the model trained using the aforementioned dataset, the dataset used in the project is considered valid.

4.2.4 GPU Memory Requirement

GPUs provided by Google Colaboratory are used for training the OCR model, as shown in Figure 19 below. From the experimental results, the RAM usage is between 2 GB and 3.5 GB while the disk memory usage is always smaller than 32 GB. Hence, the GPU memory is smaller than 64 GB for all cases, which shows that the requirement for GPU memory usage is met.



Connected to "Python 3 Google Compute Engine backend (GPU)"

RAM: 2.32 GB/12.72 GB Disk: 29.83 GB/358.27 GB

Figure 18: GPU

4.2.3 Character Segmentation Accuracy

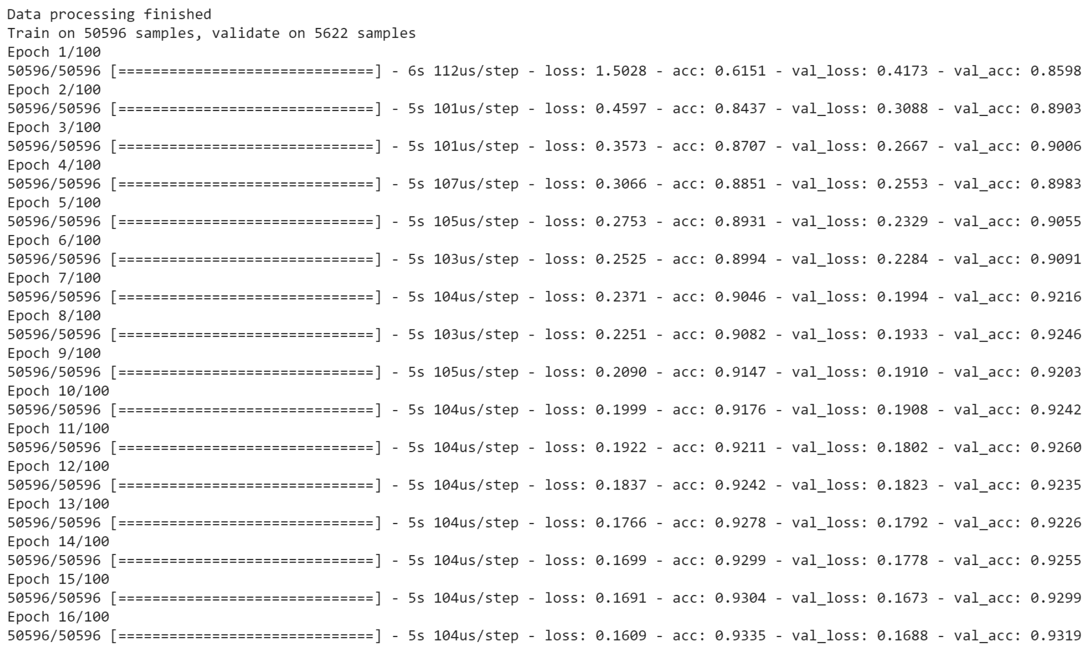
Table 12 shows the degrees of accuracies to which the recognition engine separates different characters and digits into regions for the recognition process. From the experimental results recorded in Table 12, the average accuracy is around 99%, which is significantly higher than 80%.

Table 11: Character Segmentation Accuracy

Experiment No.	Total Character Count	Segmented Count	Segmenting Accuracy
1	385	381	0.9896
2	277	277	1.0000
3	482	479	0.9938
4	420	417	0.9929
5	318	310	0.9748
Average	1882	1864	0.9904

4.2.4 Recognition Accuracy

From the experimental results shown in Figure 18, at epoch 16 of the training process, the validation accuracy has already reached 93.19%, which shows that the recognition accuracy of characters and digits exceeds the requirement of 80% accuracy.



```
Data processing finished
Train on 50596 samples, validate on 5622 samples
Epoch 1/100
50596/50596 [==============================] - 6s 112us/step - loss: 1.5028 - acc: 0.6151 - val_loss: 0.4173 - val_acc: 0.8598
Epoch 2/100
50596/50596 [==============================] - 5s 101us/step - loss: 0.4597 - acc: 0.8437 - val_loss: 0.3088 - val_acc: 0.8903
Epoch 3/100
50596/50596 [==============================] - 5s 101us/step - loss: 0.3573 - acc: 0.8707 - val_loss: 0.2667 - val_acc: 0.9006
Epoch 4/100
50596/50596 [==============================] - 5s 107us/step - loss: 0.3066 - acc: 0.8851 - val_loss: 0.2553 - val_acc: 0.8983
Epoch 5/100
50596/50596 [==============================] - 5s 105us/step - loss: 0.2753 - acc: 0.8931 - val_loss: 0.2329 - val_acc: 0.9055
Epoch 6/100
50596/50596 [==============================] - 5s 103us/step - loss: 0.2525 - acc: 0.8994 - val_loss: 0.2284 - val_acc: 0.9091
Epoch 7/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.2371 - acc: 0.9046 - val_loss: 0.1994 - val_acc: 0.9216
Epoch 8/100
50596/50596 [==============================] - 5s 103us/step - loss: 0.2251 - acc: 0.9082 - val_loss: 0.1933 - val_acc: 0.9246
Epoch 9/100
50596/50596 [==============================] - 5s 105us/step - loss: 0.2090 - acc: 0.9147 - val_loss: 0.1910 - val_acc: 0.9203
Epoch 10/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1999 - acc: 0.9176 - val_loss: 0.1908 - val_acc: 0.9242
Epoch 11/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1922 - acc: 0.9211 - val_loss: 0.1802 - val_acc: 0.9260
Epoch 12/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1837 - acc: 0.9242 - val_loss: 0.1823 - val_acc: 0.9235
Epoch 13/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1766 - acc: 0.9278 - val_loss: 0.1792 - val_acc: 0.9226
Epoch 14/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1699 - acc: 0.9299 - val_loss: 0.1778 - val_acc: 0.9255
Epoch 15/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1691 - acc: 0.9304 - val_loss: 0.1673 - val_acc: 0.9299
Epoch 16/100
50596/50596 [==============================] - 5s 104us/step - loss: 0.1609 - acc: 0.9335 - val_loss: 0.1688 - val_acc: 0.9319
```

Figure 19: Validation Accuracies

Additionally, Table 11 below indicates the accuracy at the end of each experiment and the average accuracy over 8 runs.

Table 12: Validation Accuracies

Experiment No.	Total Epoch	Validation Accuracy
1	100	0.9435
2	100	0.9498
3	50	0.9412
4	50	0.9399
5	200	0.9511
6	200	0.9489
7	250	0.9324
8	250	0.9239
Average	N/A	0.9413

4.2.7 Recognition Efficiency

Table 13 describes the amount of time the recognition engine takes to interpret a receipt over five runs. In the experiments, the completion time for the recognition process is shorter than 10s for most cases. The only experiment that complete in more than 10s is the one with 482 characters. Since from the observation, most receipts have a word count less than 400, it is available to assume that the recognition time will not exceed 10s in general.

Table 13: Recognition Efficiency

Experiment No.	Total Character Count	Total Recognition Time (s)
1	385	6.5
2	277	4.38
3	482	10.3
4	420	9.8
5	318	6.7
Average	N/A	7.536

5. Discussion and Project Timeline

5.1 Evaluation of Final Design

ReceiptIt has a propensity for meeting the project specifications. Section 3.2 Receipt Recognition Engine Design illustrates that the application is capable of extracting essential text information with accuracy being above 80%, while precision and recall are expected to be around 80%. Meanwhile, the recognition process of an arbitrary printed copy should not exceed 10 seconds. At the same time, the image segmentation method used in the training data preparation processes is expected to successfully segment at least 80% of the characters and digits on each receipt to regions for the recognition process. Also, the GPU memory used by the neural network during training is expected to be smaller than 64 GB. In addition, the model is expected to reach the required accuracy using 200,000 training samples with sizes that are equal or smaller than 200 * 200 pixels to reduce the computational cost. Section 3.3 Server Design, in conjunction with Section 3.1 and Section 3.4, proves the viability of essential functions including the generation of expense reports and the editability of receipt information by diving into the implementation of backend APIs and the integration with the user interface in detail. Section 3.1 Mobile Client Design demonstrates that the implementation of the user interface is built with compatibility and adaptability in mind to cope with the non-functional requirements of supporting screen resolutions ranging from 240 x 320 to 1440 x 2560 px and Android OS version 7.0 or higher. Overall, the project final design is feasible and should meet all defined specifications.

5.2 Use of Advanced Knowledge

The project uses advanced knowledge from various upper-year ECE courses, including ECE 356 Databases, ECE 358 Network, and ECE 457A Adaptive and Cooperative Algorithms.

The design of the Database subsystem makes use of Entity-Relationship Diagram and Data Definition Language (DDL) covered in ECE 356.The Database subsystem utilizes MySQL and it is the primary data storage solution in this project. The data models and the way they interact are firstly illustrated via Entity-Relationship Diagrams, upon which the initial designs of database tables are built. Tables in the database are built by using DDL.

The communication between the Server and Mobile Client is made through RESTful APIs, which in turn uses HTTP as the communication protocol. HTTP is a TCP/IP based protocol and is introduced in ECE 358. REST architecture makes use of HTTP verbs to manipulate and retrieve resources, such as GET, PUT and POST.

The content of ECE 457A and ECE 356 partially cover the knowledge required for building up the receipt recognition engine. Searching algorithms introduced in ECE 457A are expected to be applied to the searching process of connected components when combining the contours of characters to words. Also, the decision tree introduced in ECE 356 is considered a classification method when training the classification model for characters and digits, which is used for the text recognition process.

5.3 Creativity, Novelty, Elegance

Comparing with current receipt managing applications, this project makes it faster to recognize text information from photos of receipts. Also, this project might be the first application that provides an all-in-one service on generating expenditure reports directly based on the receipts of purchases by the users, which made it convenient for the users to track their purchasing history and make a plan for the future expenditures.

An example of the elegant part of this project is the separation of the text information on receipt into characters. The segmentation of the text information transfers the text recognition tasks to a simple character categorization task, which reduces the computational cost and enhances the recognition accuracy.

5.4 Quality of Risk Assessment

It appears that we successfully identified the source of potential risks in our risk assessment, which was the conflicts of availabilities. We overcame the existence of time conflicts and the difficulty of communication due to different geographic locations of our internships. Google hangout meetings were scheduled on a timely manner where concerns were exchanged and addressed. We were able to mitigate the risk of falling behind the schedule by making the plan ahead of time and checking whether we met the objectives in each meeting. As the result of accurate risk assessment and effective communication, we finished up the technical implementation at the end of our co-op term.

5.5 Student Workload

Table 14 below illustrates the number of hours each student has worked on this project. The estimated percentage of the overall workload done by each student over the lifecycle of the project is calculated according to the percentage of individual hour over the overall hours. As showed in the table, the workload is evenly distributed, and everyone contributes to around 20% of the overall workload.

Table 14: Student Workload

Student	Total Number of Hours	Percentage of the Overall Workload
Boyang Cheng	57	19.06%
Jinming Zhang	61	20.4%
Tianpeng Hong	60	20.06%
Zhidong Zhang	58	19.4%
Ziyan Liu	63	21.07%

References

[1]   Visa Canada, “Budgeting,” *Practical Money Skills*, Jun. 2019;
https://practicalmoneyskills.ca/personalfinance/savingspending/budgeting/.
[2] D. A. Lussier, *MVVM Compared To MVC and MVP*, Jun. 2019;
http://geekswithblogs.net/dlussier/archive/2009/11/21/136454.aspx.
[3] R. A. Ahmadi and R. A. Ahmadi, “MVVM -How View and ViewModel should communicate?,” *AndroidPub*, Mar. 2019;
https://android.jlelse.eu/mvvm-how-view-and-viewmodel-should-communicate-8a386ce1bb42.
[4] S. Khan, et al., A Guide to Convolutional Neural Networks for Computer Vision, Morgan & Claypool Publishers, San Rafael, 2018.
[5] “An Intuitive Explanation of Convolutional Neural Networks,” blog, Aug. 11, 2016;
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets.
[6] P. Veličković, “Deep Learning for Complete Beginners: Convolutional Neural Networks with Keras”, Mar. 2017; https://cambridgespark.com/content/tutorials/convolutional-neuralnetworks-with-keras/index.html.
[7] Doxygen, “Canny Edge Detection,” *Open Source Computer Vision*, Jun. 2019;
https://docs.opencv.org/master/da/d22/tutorial_py_canny.html.
[8] MathWorks, “Linear Mapping Method Using Affine Transformation,” Jun. 2019;
https://www.mathworks.com/discovery/affine-transformation.html.
[9] X. Lu, et al., “Speech Enhancement Based on Deep Learning Autoencoder, ” 14th Annual Conf. of the Int’l Speech Communication Association (INTERSPEECH 2013), 2013, pp. 436-440.
[10] A. Dertat, “Applied Deep Learning - Part 3: Autoencoders, ” Oct. 2017;
https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798?gi=c282ae043d1a.
[11] DeepAI, “Binarization, ” Jun. 2019;
https://deepai.org/machine-learning-glossary-and-terms/binarization.
[12] B. Biggio, et al., “Image Spam Filtering Using Visual Information, ” Oct. 2007; 14th Int’l Conf. on Image Analysis and Processing (ICIAP 2007), 2017, pp. 105 - 110.
[13] “Sqlite vs. MySQL vs. PostgreSQL: A Comparison of Relational Databases,” *Logz.io*, Nov. 2018;
https://logz.io/blog/relational-database-comparison.
[14] K. Ejima, “Showdown: MySQL 8 vs PostgreSQL 10,” *Dumper Blog*, blog, May. 23, 2018;
http://blog.dumper.io/showdown-mysql-8-vs-postgresql-10.
[15] “PostgreSQL vs. MySQL,” *Web Techniques: PostgreSQL vs. MySQL*, Jun. 2019;
https://people.apache.org/~jim/NewArchitect/webtech/2001/09/jepson/index.html.
-a-developer/.