



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jingyu Liu

TEXT SEARCH WEB APPLICATION

Technology and Communication

2020

ACKNOWLEDGEMENTS

I would like to express my gratitude to VAMK University of Applied Science and Wärtsilä, particularly Nishant Redekar and Vesa Mustonen. They gave me a chance to learn through working with Robotic Process Automation (RPA) team and provided me with the Optical character recognition (OCR) Validation dissertation project. All the members of the RPA team are well-versed and amicable. With their help, I accomplished some achievement in Wärtsilä and my experience in software development has been enhanced by training in Wärtsilä.

I must thank my undergraduate student mentor, Dr. Moghadampour Ghodrat, for his tireless education and guidance. I am grateful for his four years of teaching which brings me much knowledge about software and many technologies of server designing. I also extend my appreciation to Mr. Timo Kankaanpää who passed his knowledge of Django and Python to me. The Django is the project framework language of my thesis project.

Finally, thanks to all the people who helped me.

ABSTRACT

Author	Jingyu Liu
Title	Text Search Web Application
Year	2020
Language	English
Pages	63
Name of Supervisor	Moghadampour Ghodrat

This project provides a solution for Wärtsilä to store the documents including contracts and proposals as well as extract the texts from them.

Text Search Web Application project was built in the Django framework and Pytesseract Python module. This thesis project was made in four main parts. The first part is the Django framework. Django is an open-source web framework based on Python language. It lightens the workload of designing database-driven websites. The second is MongoDB which is a Not Only Structured Query Language (NoSQL) database. With the help of MongoDB, the scanned files can be stored in a file storage system locally that will reduce the stereo matching time and the information storage significantly. The third is Asynchronous JavaScript and XML (AJAX), a set of web development techniques. AJAX helps project POST data from web forms to the backend server when Django was not able to complete this. The last is the Pytesseract Python module. Pytesseract is an optical character recognition command library developed for Python. It retrieves the text data from image files. It is a wrapper for Google's Tesseract-OCR Engine and in this thesis, it deploys words extracting and fetching positions in files as a simple module.

The web application was implemented with client-server model web architecture. The web application allows users to register their accounts for storing the documents.

CONTENTS

ABSTRACT

1	INTRODUCTION.....	10
2	WÄRTSILÄ OYJ ABP	11
2.1	Strategy	12
2.2	Wärtsilä Marine Business	12
2.3	Wärtsilä Energy Business	13
2.4	RPA Team in Wärtsilä	13
3	RELEVANT TECHNOLOGIES.....	14
3.1	Python	14
3.2	Django.....	15
3.2.1	Features of Django	15
3.2.2	Architecture of Django.....	17
3.2.3	Model	18
3.2.4	View	19
3.2.5	Controller	20
3.2.6	Core Modules of Django	20
3.3	OpenCV	21
3.4	Deep Learning OCR Engine	21
3.5	Tesseract OCR	23
3.6	Training Tesseract.....	24
3.7	JavaScript.....	24
3.7.1	Dropzone.js	25
3.7.2	Maphilight.js	25
3.8	AJAX	26
3.9	MongoDB	26
3.9.1	Comparison between SQL and NoSQL	27
3.10	Docker.....	29
4	APPLICATION DESCRIPTION	30
4.1	Quality Function Deployment.....	30
4.2	Use-case Diagram	31

4.3	Class Diagram.....	32
4.4	Sequence Diagram	34
4.5	Component Diagram.....	38
5	DATABASE AND GUI DESIGN	39
5.1	Database Design.....	39
5.2	GUI design	40
6	IMPLEMENTATION	44
6.1	Dockerize the Project.....	44
6.1.1	Dockerfile.....	44
6.1.2	Docker-compose.yml	45
6.2	Classes in Django Framework	46
6.3	Media and Static files and Template Folder	48
6.4	Tesseract	48
6.4.1	Installation.....	48
6.4.2	Image pre-processing	49
6.4.3	Applying the Tesseract.....	50
6.5	Initializing the Dropzone.js.....	52
7	TESTING	54
7.1	Registration Page	54
7.2	Login page	56
7.3	Create Directory Project page.....	57
7.4	File Listing page	58
7.5	Validation page	59
8	SUMMARY	60
9	CONCLUSIONS	61
9.1	Future work.....	61
	REFERENCES.....	62

LIST OF ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BSD	the Berkeley Software Distribution
GPL	GNU General Public License
GUI	Graphical User Interface
GW	Gigawatt
Html	Hypertext Markup Language
iOS	iPhone Operating System
LSTMs	"Long Short-term Memory" Units
MB	Megabyte
NoSQL	Not Only Structured Query Language
OCR	Optical Character Recognition
OOP	Object-oriented Programming
OpenCV	Open source Computer Vision
PDF	Portable Document Format
px	pixel
R&D	Research and Development
RPA	Robotic Process Automation
tiff	Tagged Image File Format
UI	User Interface
URL	Uniform Resource Locator
URLconf	Uniform Resource Locator dispatcher
VM	Virtual Machine

LIST OF FIGURES, TABLES, AND CODES

FIGURES

Figure 1. Wäertsilä's strategy. /1/	12
Figure 2. Rapid Application Development Methodology. /6/	15
Figure 3. Screenshot from stackshare.io. /8/	16
Figure 4. MVC Pattern Diagram. /17/	17
Figure 5. Django MVT Pattern. /9/	19
Figure 6. Optical Character Recognition process. /18/	22
Figure 7. Tesseract OCR engine architecture. /4/	23
Figure 8. Mahilight.js usage in OCR validation.	26
Figure 9. Comparison between Container and VM. /15/	29
Figure 10. Use-case Diagram.	32
Figure 11. Four core classes.	33
Figure 12. User class and related classes.	34
Figure 13. Register sequence.	35
Figure 14. Login sequence.	36
Figure 15. Create a Directory Project sequence.	36
Figure 16. Uploading documents sequence.	37
Figure 17. Get metadata sequence.	37
Figure 18. Component Diagram.	38
Figure 19. Application database structure.	40
Figure 20. The home web page.	41
Figure 21. The register web page.	41
Figure 22. The login web page.	42
Figure 23. The directory project creating a web page.	42
Figure 24. The OCR files listing web page.	43
Figure 25. The validation web page.	43
Figure 26. Project Django Structure.	47
Figure 27. Tesseract result with the EAST model.	50
Figure 28. Tesseract execution time with the EAST model.	50
Figure 29. Tesseract result without the EAST model.	51

Figure 30. Tesseract execution time without the EAST model.	51
Figure 31. Test Register page with empty field.	54
Figure 32. Test Register page passwords do not match.	55
Figure 33. Test Register page successfully register.	56
Figure 34. Test Login page with empty field.	56
Figure 35. Test Login page successfully login.	57
Figure 36. Test Create Directory Project page.	57
Figure 37. Test File Listing page shows all.	58
Figure 38. Test File Listing page shows the searching result.	59
Figure 39. Test Validation page.	59

TABLES

Table 1. Wärtsilä Key Figure in Five Years. /2/	11
Table 2. Comparison between SQL and NoSQL. /14/	27
Table 3. Corresponding terminology and concepts between MongoDB and MySQL. /14/	28
Table 4. Quality Function Deployment table.	31

CODES

Code 1. Example of models.py.....	18
Code 2. Example of views.py.....	19
Code 3. Dropzone.js is established in forms page.....	25
Code 4. jQuery function for invoking Maphilight.js.	25
Code 5. The script of the Dockerfile.	44
Code 6. The script of the Docker-compose.yml file.....	46
Code 7. Invoke the Tesseract.....	48
Code 8. Install the Tesseract libraries for Ubuntu.	48
Code 9 Install the Tesseract libraries for Mac.	49
Code 10 OpenCV pre-processes the images.....	49
Code 11. Get texts result.	51
Code 12. Process texts result.	52
Code 13. Dropzone.js was initialized.	53

1 INTRODUCTION

Wärtsilä is a Finnish company, which provides power sources services and manufactures marines' engines. With the continuous evolution of Artificial Intelligence, more companies have their automation projects and teams. As one of the world-leading company, Wärtsilä has made some achievements in automation fields for five years. However, it has been a relatively empty state in terms of automatic document storage and processing. Wärtsilä stores thousands of documents every year manually.

Under the circumstances, Wärtsilä needs a solution to store documents ordered automatically so they can extract texts from these documents. The application should be easy to use and easy to deploy. There should not be unfriendly user interfaces or functions. The constructor of the database of the application needs to be simple.

The best approach is to implement a web application to help Wärtsilä storing documents in order. The web application should keep the metadata of the documents. The documents need to be saved under many specific locations. The locations seem like directories for the operation system. The directories make the documents storage system more organized. Every document will be converted to an image format. In the image format, the documents can provide a preview for users to select corresponding files. It also allows the application to capture text information better from the documents. If there is an error in word recognition, the application allows users to edit the misrecognized field.

This application aims to create a web application for searching text from the documents of Wärtsilä. This web application has the functions of storing image and PDF format files and the function of extracting the text data from these files. The project should allow users to register accounts and permit users to upload files, delete files, and search files.

2 WÄRTSILÄ OYJ ABP

Wärtsilä is specialized in providing the world's leading products, lifelong services, and design solutions in the marine power and energy market. The main market of Wärtsilä is manufacturing and serving power sources and other equipment of marine and energy. The core products in the Energy field of Wärtsilä are gas power, multi-fuel power, liquid fuel power, and biofuel power. For the technology in the Marine field, Wärtsilä provides the services covering from the cruise, ferries to fishing vessels, offshore, and yachts. One out of every three marines sailing on the global ocean is powered by Wärtsilä, and one out of every two ships was serviced by Wärtsilä.

Wärtsilä sales and service networks are covered in over 200 locations in more than 80 countries all over the world. Wärtsilä regards developing long-term relationships with its global suppliers as important. Therefore, Wärtsilä cooperates with approximately 1,250 direct global suppliers. There are approximately 19,000 employees from 140 nationalities working for Wärtsilä. /1/ The net sales are approximately 5,170 billion Euros in 2019.

Table 1. Wärtsilä Key Figure in Five Years. /2/

Five years in figures, MEUR					
	2019	2018	2017	2016	2015
Net sales	5 170	5 174	4 911	4 801	5 029
of which outside Finland, %	98.5	98.9	97.4	97.5	97.8
Exports from Finland	1 933	2 145	1 953	1 804	1 936
Personnel on average	19 110	18 899	17 866	18 332	18 565
of which in Finland	3 868	3 766	3 521	3 482	3 580
Order book	5 878	6 166	5 100	4 696	4 882

2.1 Strategy

As the global ecology deteriorates further, the leader in global energy companies, Wärtsilä realizes quickly that clean and flexible energy is important. Efficient and safe transportation is also important. Customers are also concerned about this. To achieve these, Wärtsilä made a corresponding strategy, which was Smart Energy and Smart Marine. /1/

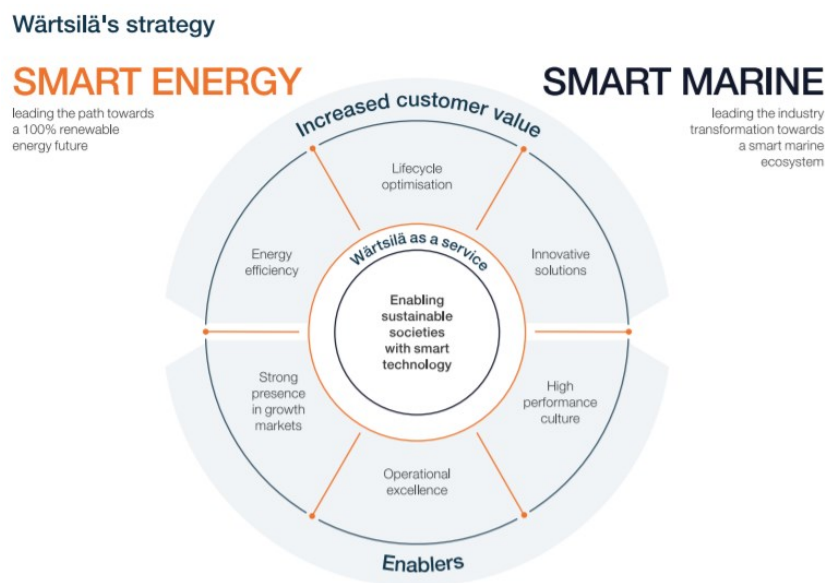


Figure 1. Wärtsilä's strategy. /1/

2.2 Wärtsilä Marine Business

Wärtsilä is now focusing on creating a Smart Marine Ecosystem. There are two aspects Wärtsilä decided to pay effort on, one is the maritime industry using only the cleanest available fuels. The other is on-board power production which optimizes and designs the routes to avoid navigation malfunctions and nautical traffic accidents causing time waiting.

Wärtsilä has a great impact on the world in the marine field. There are more than 50,000 vessels operated with Wärtsilä products and more installed. Nearly half of the vessels on the global oceans are serviced by Wärtsilä.

2.3 Wärtsilä Energy Business

Wärtsilä is gradually moving towards the use of one hundred percent renewable energy. Wärtsilä also helps their customers to unlock the value of energy transformation. The products of Wärtsilä in the energy field are flexible power plants, energy management, and storage system, also including lifecycle services. Wärtsilä has 72 GW of installed power plant capacity in 180 countries.

2.4 RPA Team in Wärtsilä

The Robotic Process Automation was set up in 2016 by Vesa Mustonen. He is the Project Manager in the Wärtsilä RPA team. The goal of the team is to raise the digital transformation to the general staff level and empower all businesses and functions in Wärtsilä to the next phase of automation. /3/

Now, including Mr. Mustonen, there are eight experts in the IM Process Automation team. They contribute to the team in their respective fields of expertise. They help each other and share experience to work together to achieve the team's goals and values.

3 RELEVANT TECHNOLOGIES

This web application was developed with the Django web framework and the OCR functions were invoked with Pytesseract and OpenCV. In addition, the web application involved in some JavaScript libraries including Dropzone.js and Maphilight.js. Data transferred between the front and back ends of the web application was also applied to Ajax. The MongoDB was used to store data in this application. Finally, the whole web application was implemented by Docker.

3.1 Python

Python is an elegant and robust programming language. It inherits the power and versatility of traditional compiled languages. It is also easy to use like scripting languages and interpreted languages.

Python was developed by Guido van Rossum during Christmas in 1989. To pass the Christmas boredom, he created Python, a newly interpreted scripting language. The name of Python was from the popular comedy serial “Monty Python” on the BBC at the time. The first public release of Python was released in 1991. It is purely free software. The source code and interpreter (CPython) follow the GNU General Public License (GPL). /19/

Python is a fully object-oriented programming (OOP) language. Functions, modules, numbers, strings, and all built-in types are all objects in Python. Python class supports advanced OOP concepts including polymorphism, operator overloading, and multiple inheritances. Besides, Python’s unique concise syntax and types make OOP very easy to use.

Python is designed to be extensible. Not all features and functions of Python are integrated into the core of the language. Python provides a wealth of APIs and tools so that developers can easily use C or C++ languages to write extension modules. Python also provides a very complete basic code base, including regular expressions, networking, multithreading, GUI, and database. In addition to the built-in libraries, Python has many third-part libraries for developers to use directly. /19/

3.2 Django

Django is an open-source large and complete web application framework, written in python language. Django adopts the MVC pattern, in some features, Django is also said called to be under the MTV pattern. It is maintained by the Django Software Foundation, an independent organization established as a non-profit.

Django was originally developed to manage the websites under the Lawrence Journal-World newspaper. The developers were Adrian Holovaty and Simon Willison. They began using Python to build the program which was a Content Management System software. It was released under the BSD license in July 2005. This framework was named after Django Reinhardt, a Belgian gypsy jazz guitarist.

3.2.1 Features of Django

Django is an open-source framework for backend web application based on Python. There are five main features of Django which are fastness, the abundance of packages, security, scalability, and versatility.

One of Django's main features is to simplify the workload for developers. The philosophy is rapid development, which means that developers can execute multiple iterations at once without having to start the entire schedule from scratch. And Django provides developers the reuse of existing code and focuses on unique code implementation. /6/

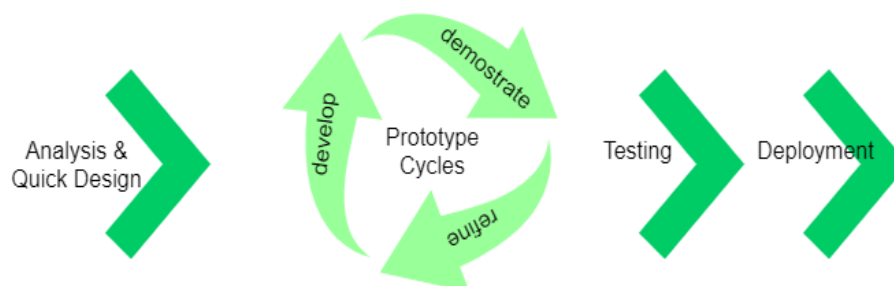


Figure 2. Rapid Application Development Methodology. /6/

Another important feature of Django is that Django includes several extra packages for handling common Web development tasks. Django prepares the extra packages for developers.

Security is also a high priority for Django. It helps developers avoid many common security issues with its out-of-the-box security systems. Among the problems are clickjacking, cross-site scripting, and SQL injection. Its user authentication system offers a safe way to manage user account and passwords. /7/

For the scalable of Django, it uses “shared-nothing” architectures, which means developers can add hardware at any level. Django scales to meet the heaviest traffic demands quickly.

Django has reached many fields from the content management system to social networks to scientific computing platforms. Django occupies a place on the Internet. It does not only serve developers but is also famous in many companies. Figure 3, shows there are 2042 companies reportedly using Django in their tech stacks.

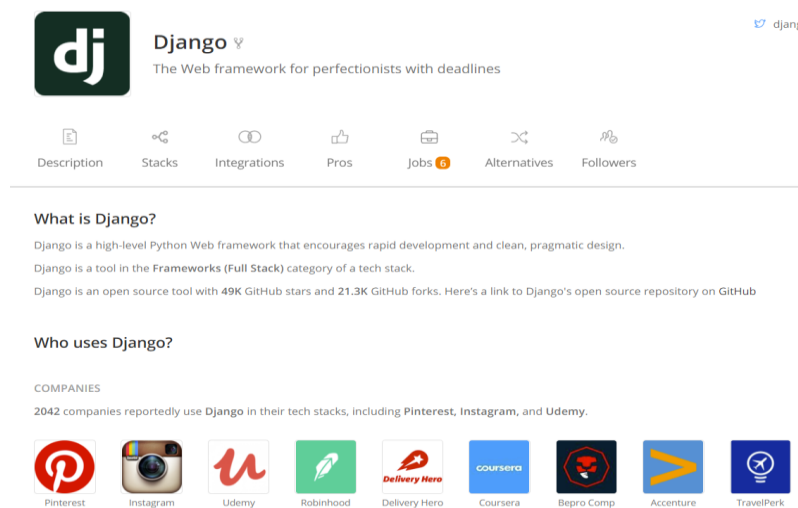


Figure 3. Screenshot from stackshare.io. /8/

3.2.2 Architecture of Django

Django architecture is based on the Model View Controller (MVC) pattern. The Django MVC architecture addresses many of the problems that existed in traditional Web Development approaches.

The MVC is a kind of framework model. It does not introduce new features but is used to guide developers to improve the application architecture. It separates the application model and views for getting better development and maintenance efficiency. In the MVC pattern, the application is divided into Model, View, and Controller, three parts. The Model includes the business processing layer and data persistence layer. The View is used to visualize the outputting data. The Controller is responsible for adjusting the model and view. It needs to be chosen which model to handle the related business based on the user's request and which view responds for the user for the final presentation form.

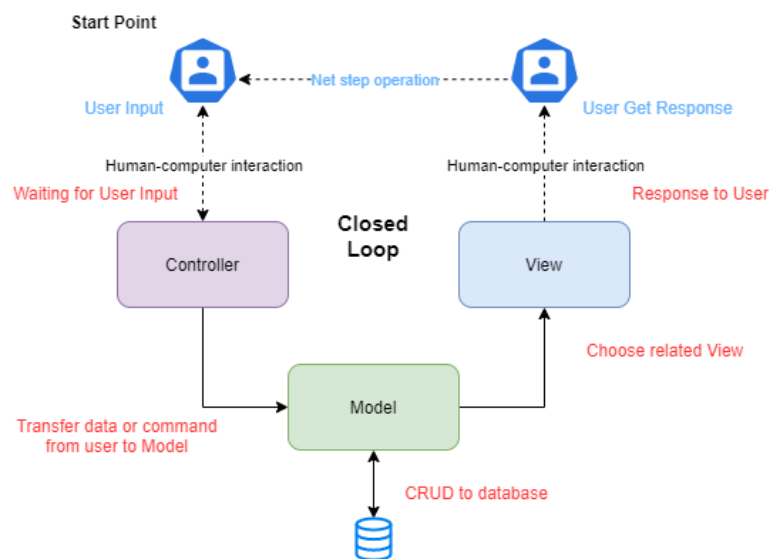


Figure 4. MVC Pattern Diagram. /17/

This difference between components helps the developer to focus on the application and each function will be better testing, debugging, and scalability experience. /9/

3.2.3 Model

The Model is one component of the web application that connects the site interface and database. One model in Django corresponds to a table in the database. In Django, the Model is represented in the form of classes and contains some basic fields with some behaviors of the data.

```
from Django.db import models
from datetime import datetime

class DirProject(models.Model):
    name = models.CharField(max_length=50)
    creator_id = models.IntegerField(default=0)
    date = models.DateField(default=datetime.now, blank=True)
    description = models.TextField()

    def __str__(self):
        return self.name
```

Code 1. Example of models.py.

The Django model uses an object-relational mapping (ORM) layer to implement the mapping between the objects and the database. It hides the details of data accessing and interacts with the database without writing SQL statements.

The Model is the component that contains Business Logic within Django architecture. /9/ It follows the MVC pattern closely, however, in the Django framework, inner URLconf works as the controller. It receives the request from the user and forwards the request. Django is more concerned about Model, Template, and Views, so, it is also called the Django MTV pattern.

The process of the Django MTV pattern is as follows: the Django framework receives the user's request and parameters, then, the URL is matched with a regular expression and forwards it to the corresponding View for handling. The View calls the Model for processing the data and uses the Template for returning the UI to the browser.

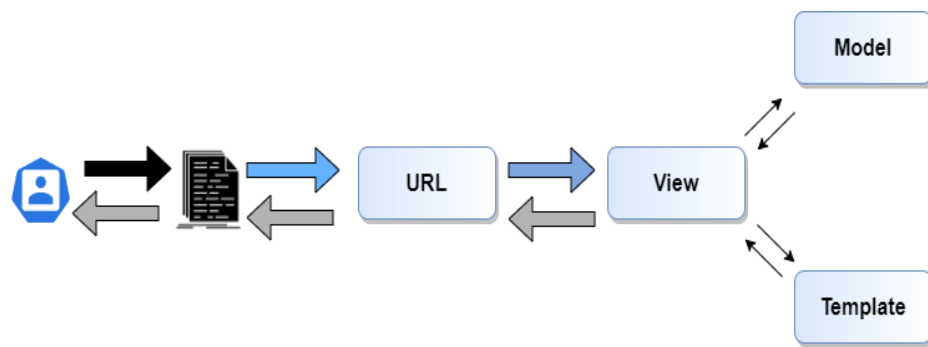


Figure 5. Django MVT Pattern. /9/

3.2.4 View

The View is the component, which contains the UI logic in the Django architecture. It is callable and it takes a request and returns a response to the front end. This can be more than a function, and Django provides some classes for developers to use as views. These allow developers to structure views and reuse code by using inheritance and mixing for tasks, as well as some generic views. The views only need to be designed to own a reusable view structure to suit all cases needed. /10/

Code 2 is from views.py of account class. It shows how to get a request from users and send responses back.

```
def login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = auth.authenticate(username=username, password=password)
        if user is not None:
            auth.login(request, user)
            messages.success(request, 'Hello!{0}'.format(username))
            return redirect('dashboard')
        else:
            messages.error(request, 'Invalid credentials')
            return redirect('login')
    else:
        return render(request, 'accounts/login.html')
```

Code 2. Example of views.py.

3.2.5 Controller

The controller is the component that works as a selector. It processes request and response and process the interaction between Model and View.

3.2.6 Core Modules of Django

- `Urls.py`

The `urls.py` is an URL entry, which is associated with a function (or generic classes) in the corresponding `views.py`. The parameters in URL patterns are written in regular expressions.

- `Views.py`

The `views.py` processes user requests from related `urls.py` and renders templates into a web page by sending the response back. It shows the web corresponding to user interaction.

- `Models.py`

The `models.py` does the operations corresponding to the database. It inserts or reads the data from the database.

- `Forms.py`

The `forms.py` is a form, which the user submits the form data from the browser. It can authenticate the submitted data and generate the input fields automatically.

- Templates Folder

The functions in `views.py` render the Html files in the templates folder to get web pages with dynamic content. It also can be used as a cache to increase the speed.

- `Admin.py`

The `admin.py` is enabled in the default project template used by the “startproject” command. It uses several lines of codes to achieve backstage management.

- Settings.py

The settings.py is a configuration file. It configures the project such as DEBUG on or off, static file location, and language changing.

3.3 OpenCV

OpenCV (Open Source Computer Vision) is a group of functions bound together as a library and it focuses on real-time computer vision. OpenCV provides more than two thousand five hundred algorithms for many areas of OpenCV's application. OpenCV also contains some machine learning library for supporting these areas.

Computer Vision is an important part of OpenCV, it works like the human eyes and brain. Eyes send signals about what they see, and the brain recognizes the objects, motions, and colors. Computer Vision wants to achieve this by computers and other electronic equipment fetch information from digital images and videos. The OCR engine is one of the applications using Computer Vision to extract data.

The main application areas of OpenCV are image processing, video analysis, face recognition, and object detection. Although they are related to Computer Vision, there are still needed libraries for real-time applications. This is the intention of OpenCV.

Nowadays, OpenCV is widely used in many personal or commercial areas. For robotics, it can enable robots to interact with human and avoid obstacles. For medicine, it helps to detect cells or tumors, builds three-dimension models of organs and vision-guide robotic arms for sugaring. For security, it helps drivers avoid sudden accidents and protect property by face recognition.

3.4 Deep Learning OCR Engine

Optical character recognition is a way of transforming two-dimensional words on images or papers into plain text. The words can contain computer printed fonts or handwriting text. With OCR, several subprocesses are possible to be achieved. For

example, there are text localization, character segmentation, and character recognition. These subprocesses above can vary, but these were roughly the steps needed to implement automatic character recognition in OCR software. The OCR software has a primary purpose which is to identify and capture all unique words using different languages from literal characters. /18/

Conventional OCR systems always face the problem that trouble reading a few fonts and page formats. And conventional OCR has never had a marginal impact on the total number of documents that need to be converted to digital form. /4/



Figure 6. Optical Character Recognition process. /18/

The next-generation OCR engine uses the latest research in the field of deep learning to well solve the problems. By using a combination of deep learning models and available large data sets, the engine can achieve the most advanced accuracy for the tasks. It is also possible to generate synthetic data with different fonts using generated adversarial networks. /4/ When the text is geometrically distorted in unconstrained environments, the OCR engine could use deep learning modules, such as OpenCV to overcome the problems. Because of the various use cases of deep learning-based OCR, for example, the technology still has great potential.

3.5 Tesseract OCR

LSTMs are the most powerful type of artificial neural network for recognizing and learning data sequences. However, when there are too many states in a sequence, the speed is not so fast. So, the long-term training is better. Tesseract is developed from Python's OCRopus model. CLSTM is a branch of LSMT in C++ and an implementation of the LSTM recursive neural network model in C++ that use the Eigen library for numerical calculation. /4/ In addition, Tesseract also has Android and iOS wrappers which makes it useful for the smartphone application.

Tesseract features traditional step-by-step pipeline architecture illustrated in Figure 7. There are six steps, in the beginning, the image is processed with adaptive thresholding and converted into an image in binary format. Then connected component analysis was processed to give a character outline. The following steps for finding characters and combining characters into words from the outlines are completed. In the end, two-pass word recognition is done by clustering and classification. The Tesseract always consults with both the language dictionary and user-defined dictionary for deciding the result of word recognition.

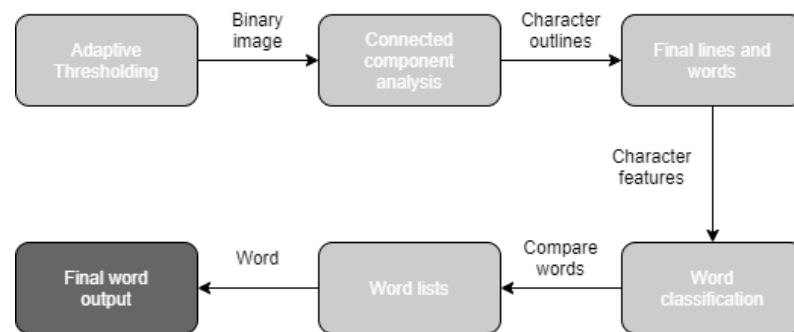


Figure 7. Tesseract OCR engine architecture. /4/

3.6 Training Tesseract

In the training part, Tesseract needs a tiff or a pdf file of a text written in the same language for being recognized. Every character in the learning text has four different feature vectors extracted by Tesseract. Each character is constructed into a model by using the clustering technique. These models are used in the classification phase for being chosen which character should be recognized.

There are several different features that can be tested including font of the text, size of the characters, and content of texts for pretesting the training texts. These three problems are done with tests. In the first method, the training text is written in a font composed of high-quality images of individual characters. In the second method, the font of the training text is like the font of the images. In the third method, the character sizes are from sixteen px to forty-eight px. Regarding the content of the training text, two different methods are tested. /5/

3.7 JavaScript

JavaScript was created by a Netscape engineer named Brendan Eich in 1995. In early 1996, JavaScript was first applied to the Netscape two browsers. Netscape changed its name from the original LiveScript to JavaScript because of the rise of Sun Microsystem's Java language. /11/

Unlike most programming languages, JavaScript has no concept of input or output. It is a scripting language that operates under the host environment. Browsers are the most common hosting environment, but JavaScript interpreters are also included in many other programs. It includes Adobe Acrobat, Adobe Photoshop, Yahoo!'s Widget engine, NoSQL database, server-side environments like Node.js, and embedded computers.

JavaScript is a multi-paradigm dynamic language that contains types, operators, standard built-in objects, and methods. The syntax comes from Java and C language. There are many common grammatical features between JavaScript with Java and C. JavaScript supports object-oriented programming through prototype rather

than classes. JavaScript also supports functional programming. Functions can also be stored in variables and passed like other objects.

3.7.1 Dropzone.js

Dropzone.js is an opensource in JavaScript library. It supports AJAX asynchronous upload function.

There are two ways to achieve files uploading by Dropzone. One is using the form, another is using div. Code 3 was from the thesis project. It shows that the Dropzone is used in the form. This method is easier than using div. The Dropzone will search all the class attributes from form elements that contained “dropzone”. By adding dropzone functions into these forms automatically, the forms upload the dropped files to the action destination.

```
<form action={% url 'createnew' %} method="post" class="dropzone"
id="myDropzone" enctype="multipart/form-data">{% csrf_token %}
  <div class="form-group">
    <div class="col-lg-10">
      <div class="fallback">
        <input name="file" type="file" multiple />
      </div>
    </div>
  </div>
</form>
```

Code 3. Dropzone.js is established in forms page.

3.7.2 Maphilight.js

Maphilight.js is a jQuery plugin that adds visual highlights to image maps. In this thesis project, the Maphilight was used for showing corresponding positions of recognition texts. It provides a single jQuery function:

```
$( '.Foo' ).Maphilight()
```

Code 4. jQuery function for invoking Maphilight.js.

Figure 8 shows how the Mahilight.js was used on the Validation page. The left side in Figure 8 is a screenshot of the code written.



Figure 8. Mahilight.js usage in OCR validation.

3.8 AJAX

Asynchronous JavaScript and XML (AJAX) is not a programming language but a new method of using existing standards. AJAX is the art of exchanging data with the server and updating part of the web page without reloading the retired page.

Ajax is a technology for creating fast and dynamic web pages. Through a small amount of data exchange with the server in the backend, AJAX can enable the web page to be updated asynchronously. /12/ Many famous applications are using AJAX including Google Maps, Sina website, and YouTube.

3.9 MongoDB

MongoDB is a scalable, high-performance NoSQL database. It is written in C++ language, designed to provide a data storage solution for web applications. There are several main features of MongoDB, the first is supporting dynamic query and a full index. Users can easily query the embedded objects and arrays in the document. The second is set-oriented storage. MongoDB is easy to store object-type data, including document embedded objects and arrays. The third is efficient data storage.

It supports binary data and large objects including photos and videos. It also supports cloud-level scalability for automatic sharing and supports horizontal database clusters. /13/

3.9.1 Comparison between SQL and NoSQL

The Comparison table below shows the difference between the SQL database and the NoSQL database.

The differences are separated into six parts. They compare the types, data storage model, pattern, scalability, consistency, and query function between the SQL database and the NoSQL database.

Table 2. Comparison between SQL and NoSQL. /14/

	SQL Database	NoSQL Database
Types	All types support the SQL standard	Many types including document storage, key-value storage, and column database
Examples	MySQL, SQL Server, Oracle	MongoDB, HBase, Cassandra
Data storage model	The data is stored in rows and columns of the table, each of which has a specific type. Tables are usually created according to standardized principles	Use joins to retrieve data from multiple tables. The data model depends on the database type. NoSQL's data model is flexible, but the SQL database is not.
Pattern	Fixed structure and mode. So, any changes to the pattern involve modifying the database	Dynamic mode. It can adapt to new data types or structures by expanding or modifying the current pattern. New fields can be added dynamically.

Scalability	The vertical expansion method is used in SQL. This means that as the load increases, the larger and more expensive servers need to be established.	The horizontal expansion method is used in NoSQL. This means that the data load can be distributed to multiple inexpensive servers.
Consistency	Strong consistency.	Depends on the product. Some products provide strong consistency, while some not.
Query function	Can be used through a simplified GUI interface.	Require programming expertise and knowledge for querying. Unlike UI, it focuses on functions and programming interfaces.

Comparison Table 3 shows the conceptual difference between the MongoDB database and the MySQL database.

Table 3. Corresponding terminology and concepts between MongoDB and MySQL. /14/

MongoDB	MySQL
Database	Database
Collection	table
Document	row
Field	Column/Field
Index	Index
Lookup, embedded documents	table joins
primary key	primary key

3.10 Docker

Docker is the world's leading software container platform. Docker uses the Go language to develop and implement. It is based on the Linux kernel's cgroup, namespace, and UnionFS technologies. It encapsulates and isolated processes, which is a virtualization technology at the operating system level. Because its isolated process is independent of the host and other isolated processes, Docker is also called a container. Docker can automate repetitive tasks, including setting up and configuring a development environment. Users are allowed to create and use containers and put their applications in the containers easily. Containers can also be used for version management, copying, sharing, and modification. /15/

The traditional virtual machine technology virtualizes a set of hardware, run a complete operating system on the host, and then executes the required application processes on the system, while the application process in the container runs directly on the host's kernel. There is no kernel inside the container and there is no hardware virtualization. Therefore, containers are lighter than traditional virtual machines.

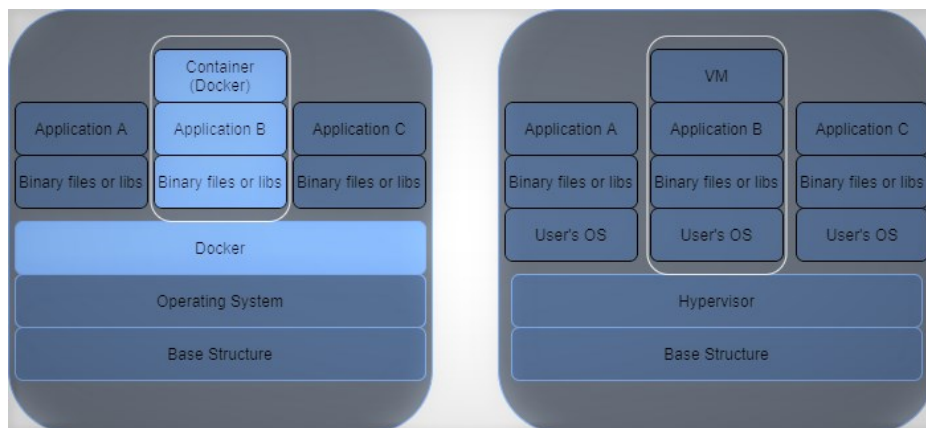


Figure 9. Comparison between Container and VM. /15/

4 APPLICATION DESCRIPTION

The idea of the application is to provide a solution for Wäritsilä to store documents easily and recognize text. All the documents are stored in the host. Information on users extracted words and metadata are stored in MongoDB. The whole project is under the Django web framework. It involves the Tesseract for words recognition and extraction.

In the application, users have to register for using the application. Then users need to create a project directory for storing the files. There is a “create” button on the home page for linking to the directory creating page. After uploading files (PDF and image type files allowed), users can go through all the files by searching form on the home page or forwarding to the OCR Files Listing page. Clicking the files name link under the previous files, users can get all the words and positions from the validation page from the selected document.

4.1 Quality Function Deployment

There are eleven functions in this thesis project. Four of them are under the highest priority. The web application uploads documents to the server, converts uploaded documents to image format for text recognition, extracts texts from documents, and edits misrecognition texts. These four functions constitute the core backbone of the entire web application. There are two “should have” functions which are deleting the documents from the server and locating the extracted texts from text recognized files. These functions provide a better user experience for users. And the rest five functions are nice to have. They provide the users with a complete web application experience and users can use the application intelligently.

Table 4. Quality Function Deployment table.

Priority	Functions
Must have	Upload Documents
	Convert Documents to Image Format
	Extract Texts from Documents
	Edit Misrecognition Texts
Should have	Delete Documents
	Locate Texts Coordinates
Nice to have	Registration and Login
	User Dashboard
	Search Documents
	Copy All Texts to Clipboard
	Training the Tesseract Models

4.2 Use-case Diagram

As the diagram shows, there are five core functions in this project. The first one is the register function. The users can register one account by providing a unique username, password, and some other personal information. Then the users can log in to the web application with the account they created. Users can create one or more project directories with or without uploading some documents. However, users have to upload some documents before obtaining all text information from files. When users get the texture data from documents, they can get texts locations in the files. They can edit the misrecognition parts of the texts and copy all the texts to the clipboard for further usage. The application also allows the users to upload new documents in an existing project directory or delete uploaded documents. The application provides searching with some options for users to search for documents under their accounts.

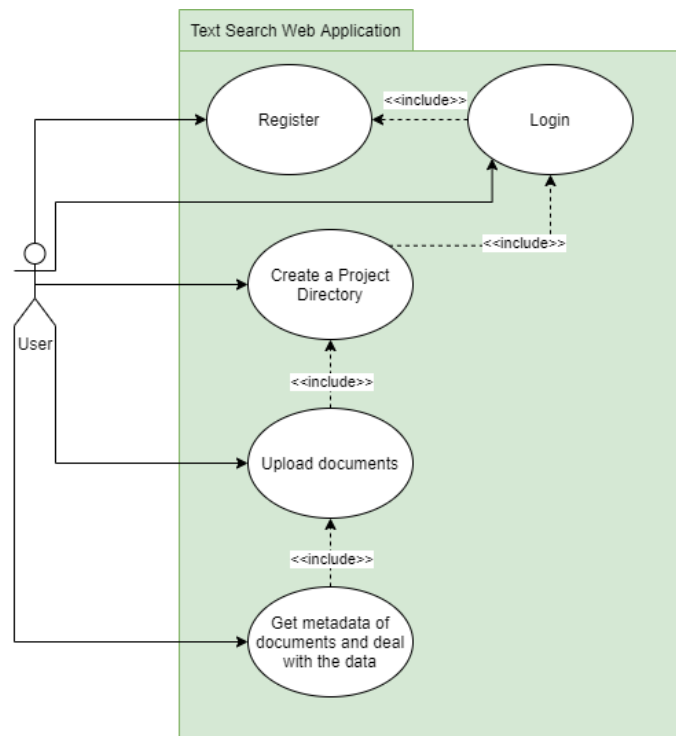


Figure 10. Use-case Diagram.

4.3 Class Diagram

This project contains four main models, which are DirProject, Ocrfiles, Validation, and OcrConvertedImage. The DirProject is used to separate uploaded files in the file storage system. It avoids the duplicated file name in the server file storage system. Ocrfiles contains three information of files including file name, file extension, and file size. It has one to one relationship with OcrConvertedImage and Validation. Files are converted to image type and inserted into the OcrConvertedImage model. Validation is used to store the extracted data from files by the Tesseract.

DirProject contains an id of users, which means that one project directory belongs to only one user. Ocrfiles has a relation with DirProject by holding a foreign key to DirProject. In this case, there are multiple OCR files in one project directory. One OCR file also has many related OcrConvertedImage tables and Validation tables.

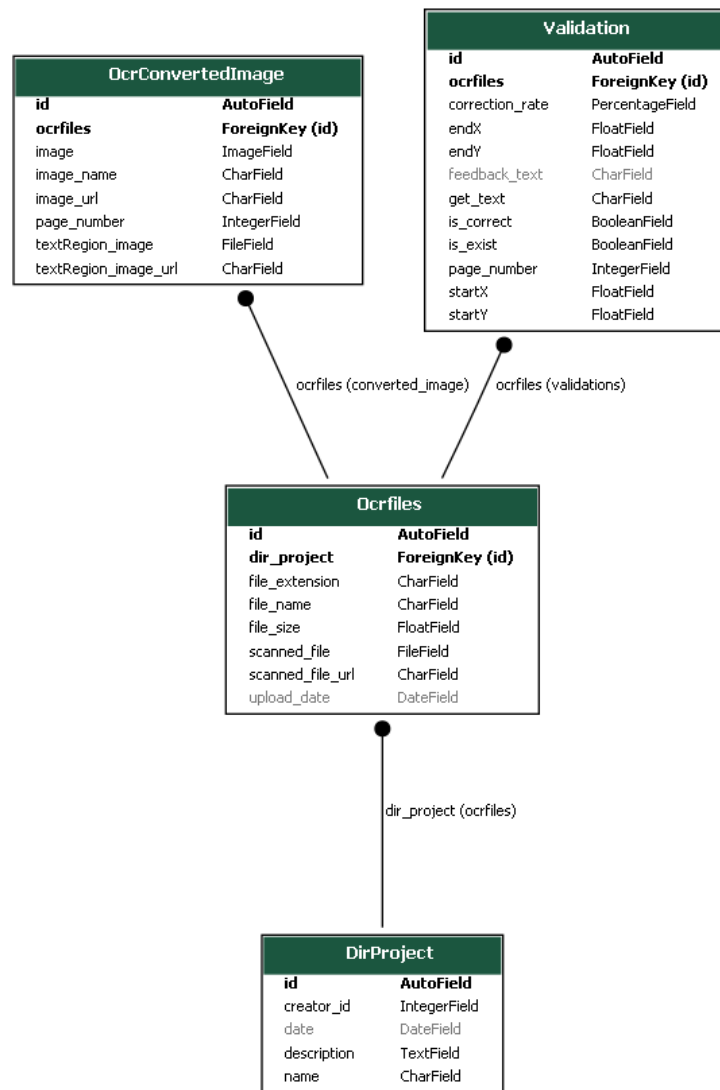


Figure 11. Four core classes.

The User class in this project is Django's native-defined user class. The User class contains eleven-member variables including id, password, username, and email. This project uses some variables for users to create accounts. The User class inherits the AbstractUser class. This class is also native-defined in the Django framework. The user class has a relationship with the group class. The group class provides permission for users that users can be an administrator and a normal user. The users do not have to join any groups, but they are given certain permission by the Permission class directory. The LogEntry class contains user logs.

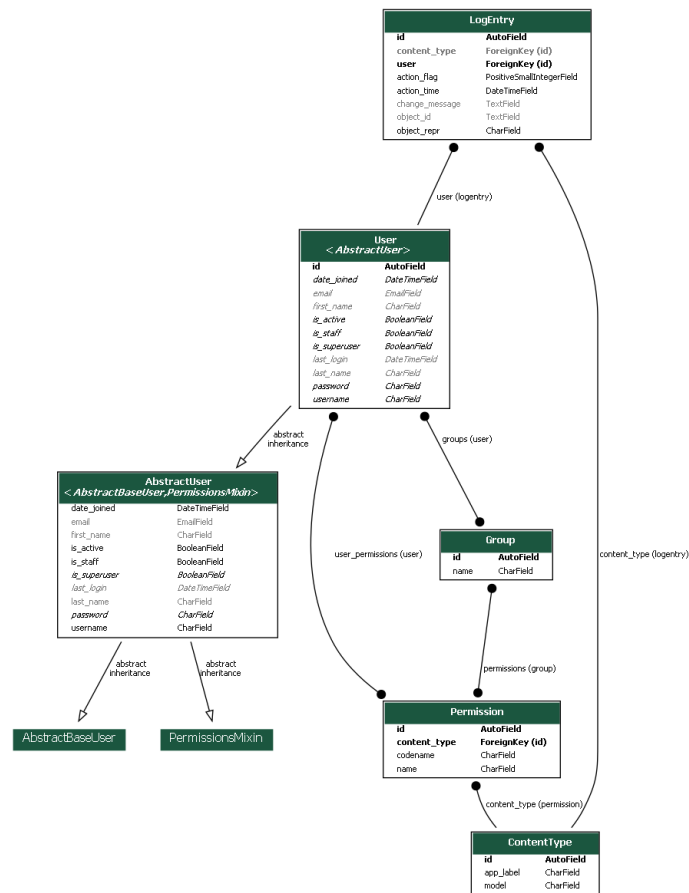


Figure 12. User class and related classes.

4.4 Sequence Diagram

There are four sequences for users handling their accounts when they use the web application. If users do not own accounts, they have to register one before using any functions in this web application. The register function will check the empty of username and password and check whether the username has been taken.

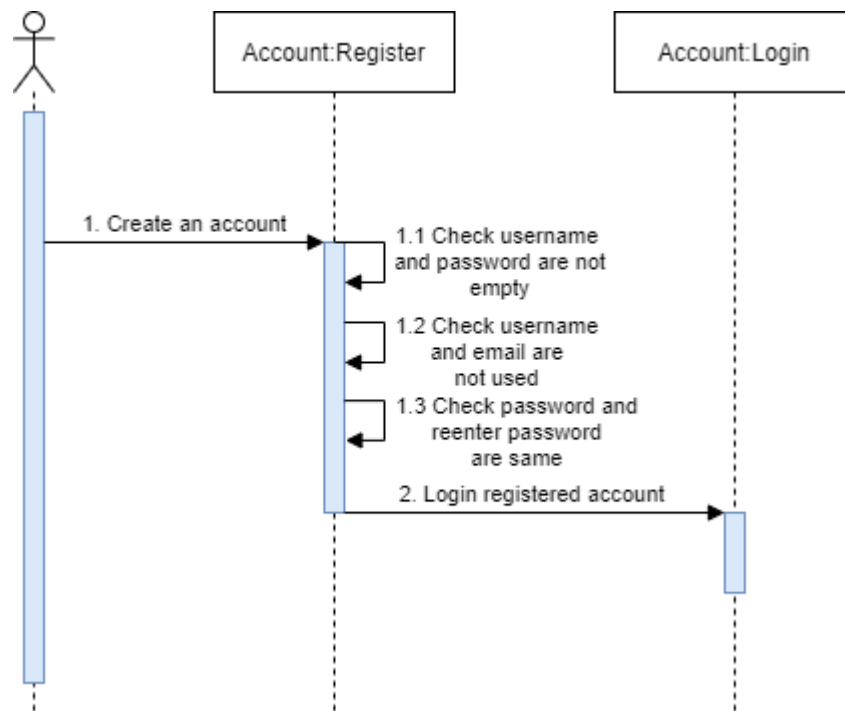


Figure 13. Register sequence.

After successfully registering, users will automatically login to the account. If users have one account, they can simply login into the account. After logging in, users can modify their Project Directories or logout.

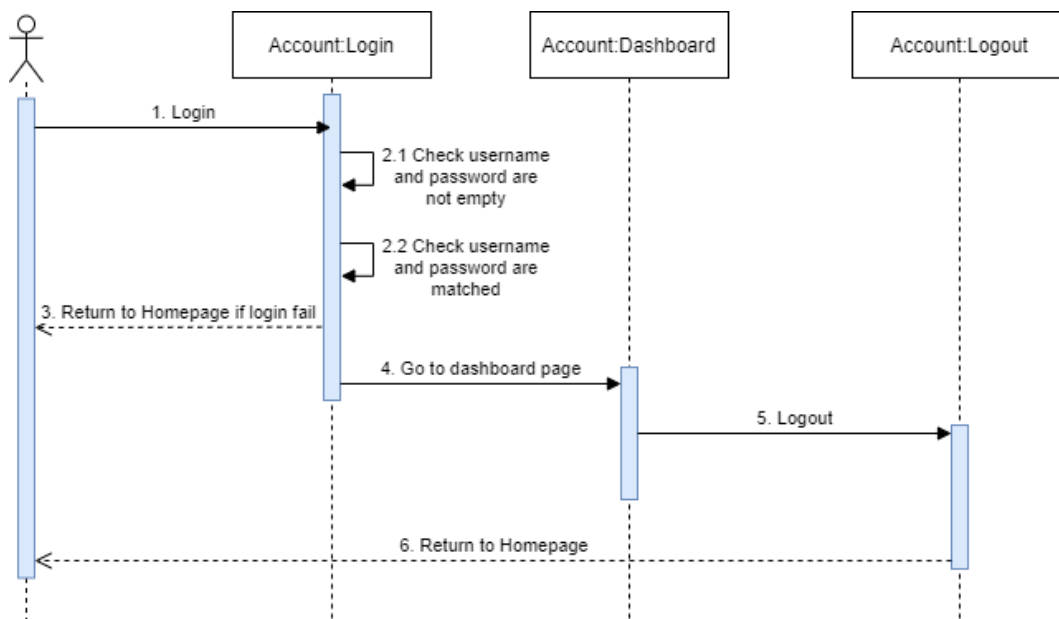


Figure 14. Login sequence.

When have users logged into their accounts, they can create a Project Directory if they do not have. They can upload some documents when they create the Project Directory or later. If the users have a Project Directory, they can search the documents by giving file name, file type, and selecting which Project Directory the documents belong to. Users can also get files list on the listing page.

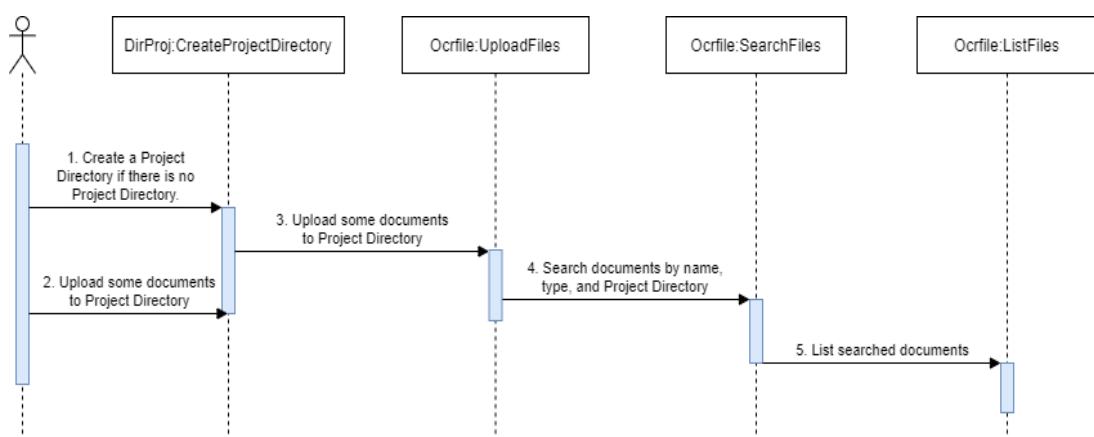


Figure 15. Create a Directory Project sequence.

In the uploading documents sequence, users can pick one or multiple documents from local machines by dropping the documents or selecting from the popup window. The application will check the format of documents and rename duplicate files with time. After uploading the documents successfully, the application jumps to the listing page for showing the preview of uploaded documents.

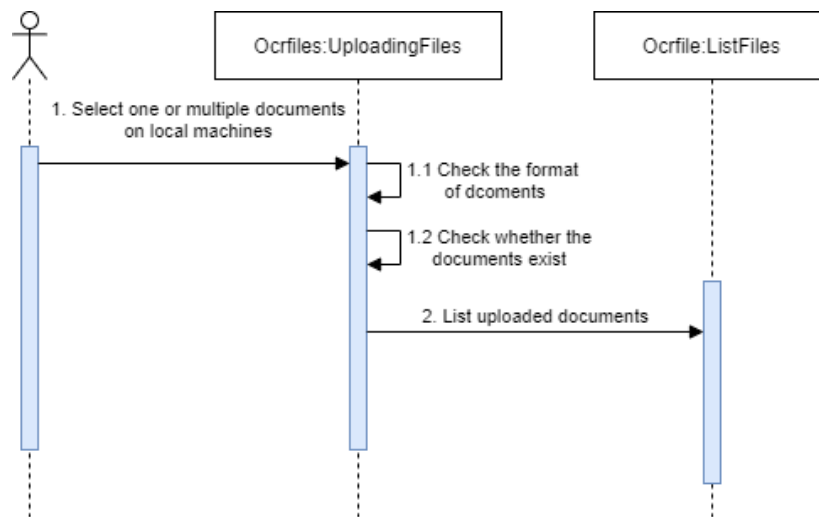


Figure 16. Uploading documents sequence.

When users find the documents and try to extract the text data from the documents, they can click the link button. The data will be presented on the validation web page. Users can edit the misrecognition texts and copy all the texts to the clipboard by clicking one button.

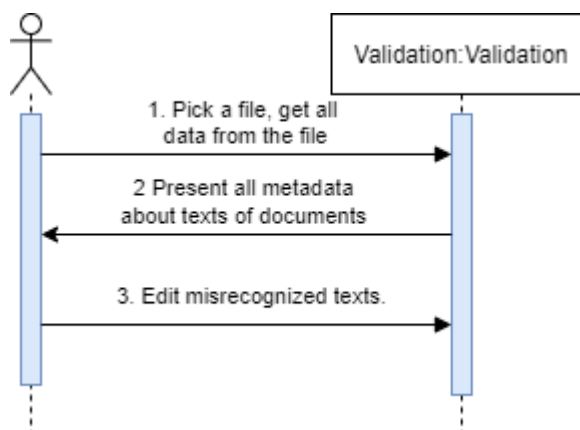


Figure 17. Get metadata sequence.

4.5 Component Diagram

The component diagram shows that all the data including account, Directory Project, and files information are saved in MongoDB. And the connection between models and databases is provided by the Django framework. The files are saved in Directory Project and extracted texts are received from converted image files. Image files are processed by the OpenCV and texts are extracted by the Tesseract.

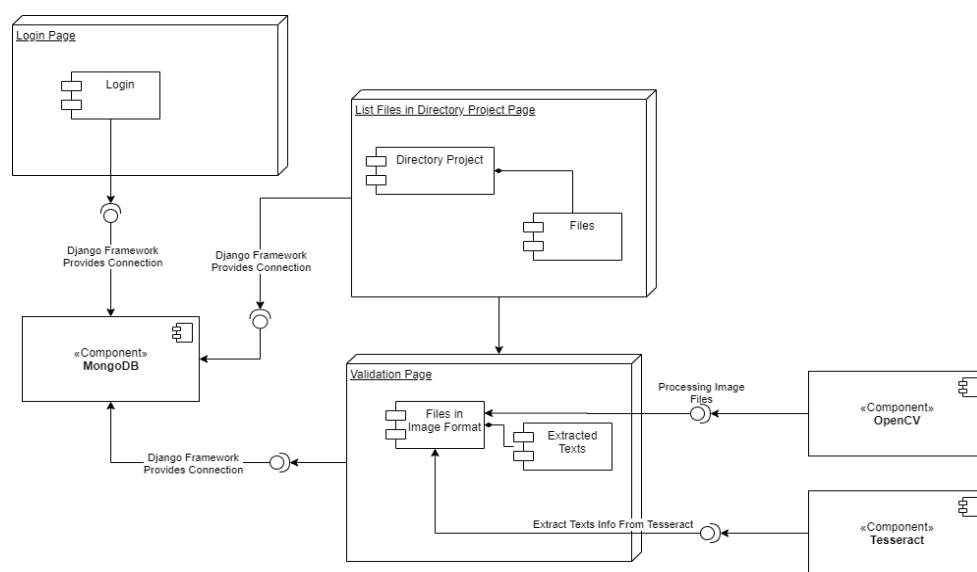


Figure 18. Component Diagram.

5 DATABASE AND GUI DESIGN

The database is deployed in MongoDB, which is a NoSQL database. There is no direct relationship between tables. The tables are connected by foreign keys. The GUI is designed by using Bootstrap mainly to arrange the page layout.

5.1 Database Design

The project uses the Django framework, which migrates the models to database tables automatically. The table of Directory Project named DirProjects contains an auto-increment integer field for ID and integer field for the user's ID who creates the Directory Project. There is a character field for name, a text field for description, and a date field for creation date.

The table of OCR files named Ocrfiles contains an ID and a foreign key for the Directory Project table. One Directory Project has multiple OCR files. The table also contains file extension, file name, file size, upload date, file storage, and storage URL.

The table of OCR converts image files named OcrConvertedImage contains an ID and a foreign key for OCR files. It ensures that one OCR file only has one OCR converted image file related. The table also contains the image file name, page number image storage, and storage URL. Text Region image storage is used for the validation page. After the Tesseract extracting the texts from images, the project will crop the image without the non-text area.

The table of validation texts named Validation contains an ID and a foreign key for OCR files. One OCR file has multiple validation texts. The table also contains correction rate, text start X, text end X, text start Y, text end Y coordinates, and text content, page number.

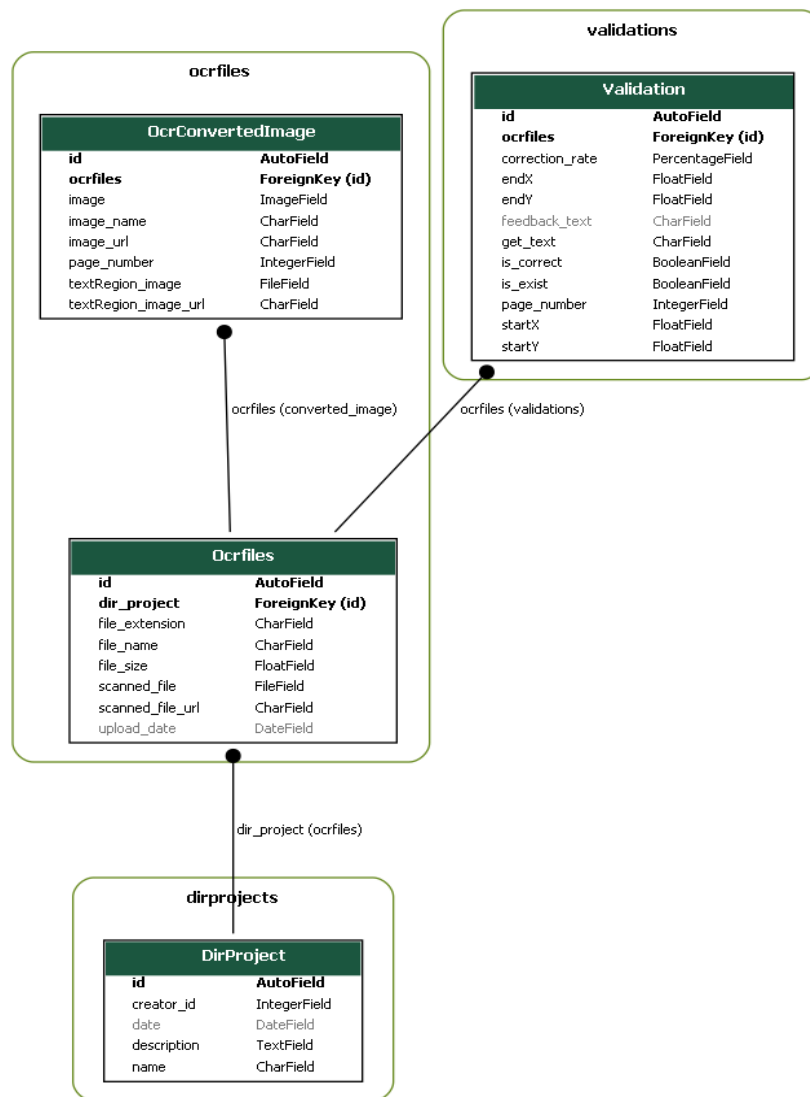


Figure 19. Application database structure.

5.2 GUI design

The main page of the project is shown with three search fields and one button for creating a new Directory Project. Users can search by file name, file type, and what Directory Project the files are in.

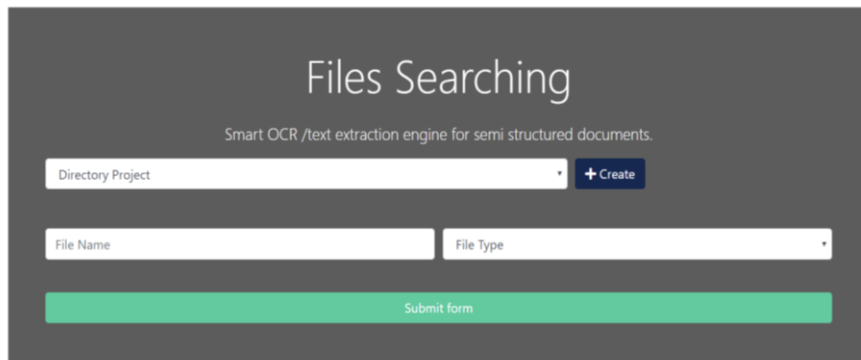


Figure 20. The home web page.

The register page requires the users to provide their first name, last name, username, email, and password. The web application will ensure the password field and confirm the password field was the same.

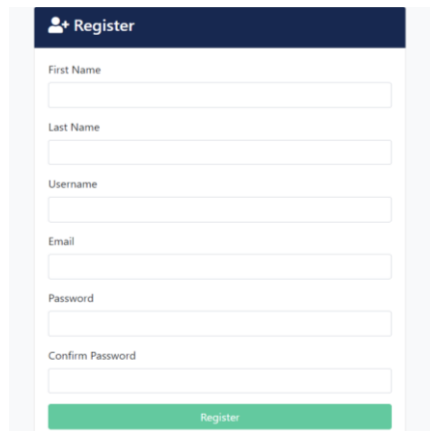


Figure 21. The register web page.

The login page contains the username and password fields. It checks none of these fields are empty and that they match each other.

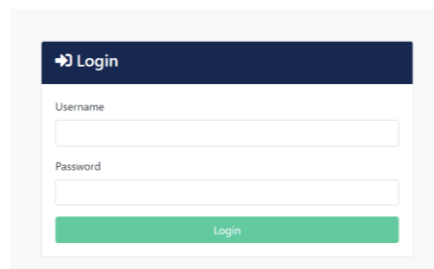

 A login web page with a dark blue header containing a white arrow icon and the word "Login". Below the header, there are two input fields: "Username" and "Password". At the bottom of the form is a green button labeled "Login".

Figure 22. The login web page.

After login to the account, users can create a new project directory for storing and recognizing the files by clicking the create button on the home page or search for needed files.


 A web page titled "Create a new Directory Project". It features two text input fields: "Project Name*" with the value "test" and "Description" with the value "test". Below these fields is a large dashed rectangular box containing three placeholder images of code editors, each with a "Delete" button underneath. At the bottom left of the page is a blue "Create" button.

Figure 23. The directory project creating a web page.

After storing files inside the directory, users can preview the files on the OCR files listing page as shown in Figure 24.

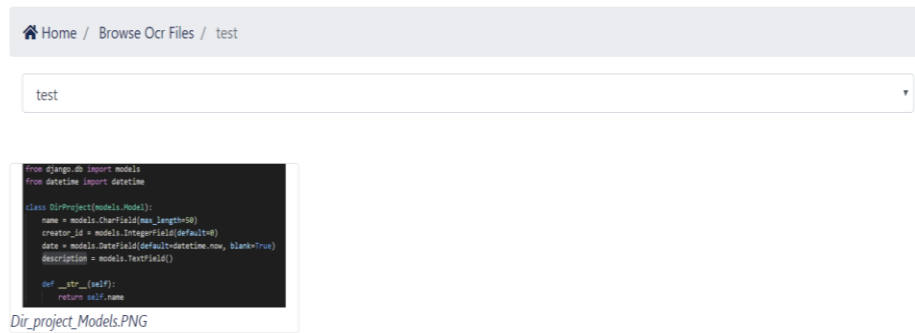


Figure 24. The OCR files listing web page.

Selecting a file for OCR takes place by clicking the name of the files under image previews. On the invalidation page, users can get all the texts from files and related positions.

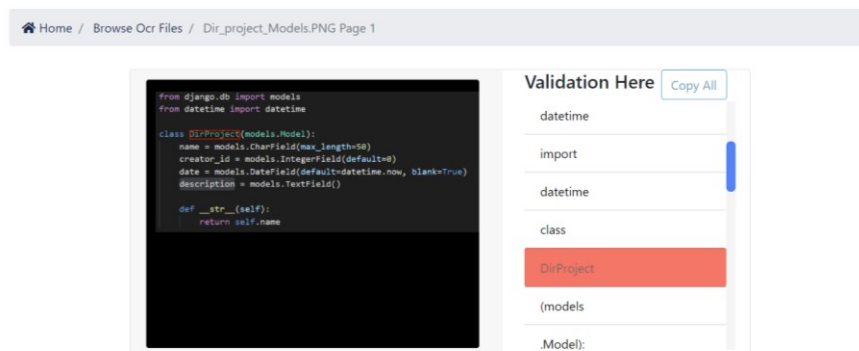


Figure 25. The validation web page.

6 IMPLEMENTATION

The project was executed under the Docker structure. Docker provides container technology that assists to package up the whole project. The project is divided into two parts. One is the Django framework based on Python. Another one is how the Tesseract extracts the metadata from raw image files.

6.1 Dockerize the Project

Two files were used when the web application was deployed by Docker. The Dockerfile provided the implementation base environment and installed the needed packages and libraries. The Docker-compose file defined and executed containers for the web application.

6.1.1 Dockerfile

The Dockerfile consists of lines of command statements and supports comment lines starting with the symbol “#”. In general, the Dockerfile is divided into four parts including basic image information, maintainer information, image operation commands, and container execution commands.

```
FROM python:3
ENV PYTHONUNBUFFERED 1
RUN mkdir /Ocr
WORKDIR /Ocr/
RUN apt update && apt install tesseract-ocr -y && apt install libtesseract-dev -y
COPY requirements.txt /Ocr/
RUN pip install -r requirements.txt
COPY . /Ocr/
```

Code 5. The script of the Dockerfile.

The script shows that Python in version three is the basic image for the project. Then “PYTHONUNBUFFERED” was set as one in the container environment variable. This meant forcing stdin, stdout, and stderr to be unbuffered. /16/ After this, Docker created a folder for storing the whole project and set it as a working folder.

Then needed the Tesseract libraries were installed and required modules of Python from the “requirements.txt” file were downloaded.

6.1.2 Docker-compose.yml

Docker-Compose is an orchestration service of Docker. It is a tool for defining and executing complex applications on Docker. Dockerfile allows users to manage a single application container while Docker-Compose allows users to define a set of related application containers in a template (YAML format).

In this thesis, five services are defined in the Docker-Compose file. The first is mongo for MongoDB. It establishes the MongoDB server on port 27017 and creates a root user for users to use the database. The second is mongo-express, which is a web UI for users checking the MongoDB database. The service is located on port 8081. The third is the web for the whole project on port 8000. The fourth is the migration for migrating the Django models to MongoDB. The last is make_migrations which makes migrations inside the Django project for preparing migrating models structures to the database.

```
version: '3'
services:
  mongo:
    image: mongo:latest
    container_name: mongoddb
    restart: always
    environment:
      MONGO_INITDB_DATABASE: ocr_db
      MONGO_INITDB_USERNAME: test
      MONGO_INITDB_PASSWORD: test
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: root
    volumes:
      # - ./init-mongo.sh:/docker-entrypoint-initdb.d/init-mongo.sh
      - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
    ports:
      - 27017:27017

  mongo-express:
    image: mongo-express
    restart: always
    ports:
      - 8081:8081
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: root
      ME_CONFIG_MONGODB_ADMINPASSWORD: root

  web:
    image: app
    restart: always
```

```

    command: python manage.py runserver 0.0.0.0:8000
    volumes:
      - ../Ocr
    ports:
      - 8000:8000
    depends_on:
      - migration

migration:
  build: .
  image: app
  command: python manage.py migrate
  volumes:
    - ../Ocr
  depends_on:
    - make_migrations

make_migrations:
  build: .
  image: app
  command: python manage.py makemigrations
  volumes:
    - ../Ocr
  depends_on:
    - mongo

```

Code 6. The script of the Docker-compose.yml file.

Code 6 shows that all the other containers have a relationship with the main container Mongo. The make_migrations will execute after the Mongo has run fully, the migration will execute after the make_migrations has run fully, the Mongo-express will execute after the migration has run fully. The volumes attribute allows the Docker to transfer the content of the files from the virtual machine to the local directory.

6.2 Classes in Django Framework

Figure 26 shows the base structure of all the project files. The accounts, dirprojects, ocrfiles, pages, and validations are classes for Django. The accounts class contains functions of login, logout, register, and dashboard. It authenticates the users through all the processes inside the web page and provides a dashboard for users to check and process the project directories.

The dirprojects class contains models for the project directory. It also processes the function of creating a new project directory.

The `ocrfiles` class contains two models which, are OCR files and converted OCR files. The OCR files model fetches the information of related project directory and basic data of uploaded files including file name, file type, file size, and file storage location. The converted OCR files model is used for images that are converted from OCR raw files.

The `tesseract` can only recognize the text from image type files. When uploaded files are saved in the file system provided by Django, the application converts the files into PNG type by `PyMuPDF` library. In order not to disorganize the `validations` class, the image type files will be saved directly in the converted OCR file model.

The `pages` class only contains home page views and URLs. It provides several choices of file types and directory project for searching.

The `validations` class contains models of OCR words data and web page for validation. It uses the `Tesseract` to extract information from image files and write the data into `MongoDB` form validation model.

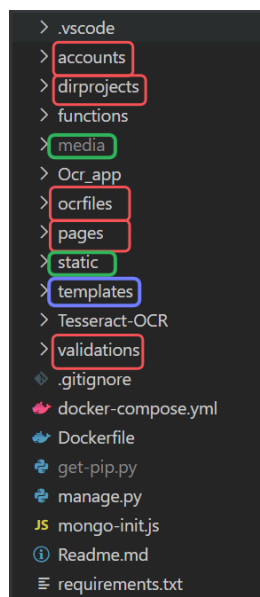


Figure 26. Project Django Structure.

6.3 Media and Static files and Template Folder

In Figure 25, the media folder and static folder are cropped inside green blocks. The static folder is for loading needed JavaScript and CSS files. It also stores images, gifs, videos, and other types of files that were used in web pages. While the media folder collects the needed files to be stored in the database. It is a file storage system under the Django structure. The files inside media folders served on a web project.

The template folder contains all the HTML files for the web application. After completing the settings.py file and urls.py file, the Django will search the HTML templates from the template folder.

6.4 Tesseract

6.4.1 Installation

Before using the pytesseract, it needs to be installed in the Tesseract libraries. For Windows system, the Tesseract can be download from GitHub URL and the Tesseract can be invoked using Code 7. Set the Tesseract path in the script before calling image_to_string.

```
pytesseract.pytesseract.tesseract_cmd
= os.path.abspath(os.getcwd()) + r'\Tesseract-OCR\tesseract.exe'
```

Code 7. Invoke the Tesseract.

For Ubuntu, it only needs to install tesseract-ocr and libtesseract-dev libraries by prompting the following command. However, in Ubuntu, there is no need to set the Tesseract path in the script (the same as in macOS) Code 8.

```
sudo apt-get install tesseract-ocr
sudo apt-get install libtesseract-dev
```

Code 8. Install the Tesseract libraries for Ubuntu.

For macOS, the Tesseract libraries also needed to be installed. Code 9 shows how to use HomeBrew to install the Tesseract.

```
brew install tesseract.
```

Code 9 Install the Tesseract libraries for Mac.

After setting the path for the Tesseract some Python packages still need to be installed including pytesseract, OpenCV-python, and pillow.

6.4.2 Image pre-processing

After the Tesseract was installed, it was ready to go with Python. However, before using the Tesseract extracting the information from images, the images needed to be processed for better results.

In this web application, the image was converted into gray scale and applied thresholding for getting only binarization. The files were saved into a temporary directory.

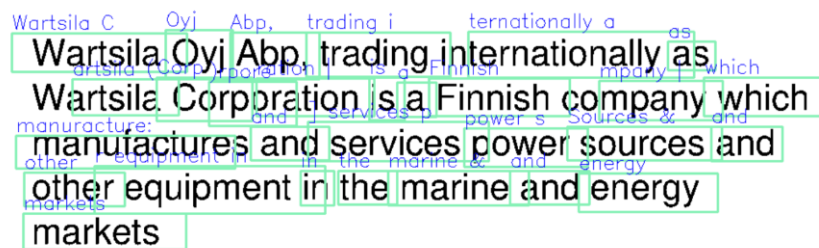
```
# image path
img_path = os.path.abspath(os.getcwd()) + r'\test1.PNG'
preprocess = "thresh"
# load the input image and grab the image deimensions
image = cv2.imread(img_path)
# convert image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Check to see if we should apply thresholding to preprocess the image
if preprocess == "thresh":
    gray = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY |
cv2.THRESH_OTSU)[1]
# check to see if median blurring should be done to remove noise
elif preprocess == "blur":
    gray = cv2.medianBlur(gray, 3)
temp_path = os.path.abspath(os.getcwd()) + r'\temp'
if not os.path.exists(temp_path):
    os.makedirs(temp_path)
# write the grayscale image to disk as a temporary file so we can
# apply OCR to it
filename = os.path.abspath(os.getcwd()) + r"\temp\{}.png".format(os.getpid())
cv2.imwrite(filename, gray)
img_gray = cv2.imread(filename)
```

Code 10 OpenCV pre-processes the images.

6.4.3 Applying the Tesseract

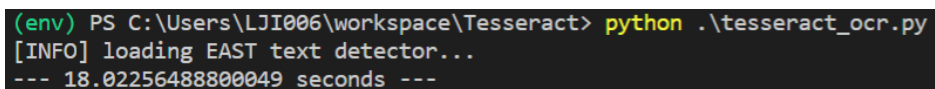
When all prework was done, the image could be recognized. The Frozen EAST Text Detection model was used before. It is a deep learning model for extracting the words from the image and scaling for the words. However, the correction rate did not achieve the desired results, and it took a long time for detecting the positions of the text.

Figure 27 shows the result of the Tesseract recognition with the EAST model and Figure 28 shows the execution time when using the EAST model.



Wartsila Oyj Abp, trading internationally as
Wartsila Corporation is a Finnish company which
manufactures and services power sources and
other equipment in the marine and energy
markets

Figure 27. Tesseract result with the EAST model.



```
(env) PS C:\Users\LJI006\workspace\Tesseract> python .\tesseract_ocr.py
[INFO] loading EAST text detector...
--- 18.02256488800049 seconds ---
```

Figure 28. Tesseract execution time with the EAST model.

Figure 29 shows the result of the Tesseract recognition without the EAST model and Figure 30 shows the execution time with only Tesseract libraries.

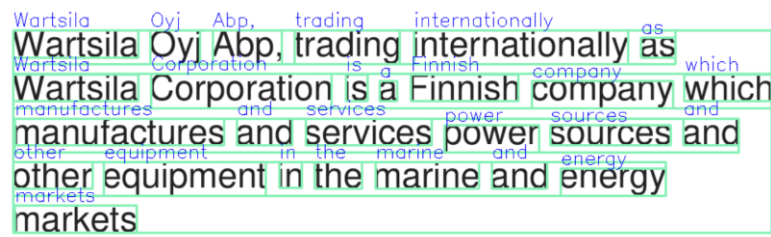


Figure 29. Tesseract result without the EAST model.

```
(env) PS C:\Users\LJI006\workspace\Tesseract> python .\tesseract.py
--- 2.2523722648620605 seconds ---
```

Figure 30. Tesseract execution time without the EAST model.

When using the EAST model, some spaces were not recognized correctly while the Tesseract accomplished it well. The EAST model took seven times of execution time than raw Tesseract. This comparison proved that the Tesseract was much more powerful than the EAST model when extracting the texts from the image.

For using the Tesseract, the `image_to_data` function was used in this project. It detects the texts in the image and converts all the data into a string. The data contains box boundaries, confidences, and text value. It also provides language selecting. The default is English.

```
res = pytesseract.image_to_data(img_gray,output_type=Output.DICT)
```

Code 11. Get texts result.

In Code 11, the `img_gray` is the pre-processing image, and `output_type` was chosen as a string. The metadata of text can be easily fetched as the following scripts. The output data was saved in the results variable and will be saved into the database.

In Code 12, the text variable got what text was and variables x, y, w, and h got what the text x coordinate, y coordinate, text width, and text height. The data was added to result in sets.

```
text = res['text'][i]
(x, y, w, h) = (res['left'][i], res['top'][i], res['width'][i],
res['height'][i])
results.append((x, y, x + w, y + h), text))
```

Code 12. Process texts result.

6.5 Initializing the Dropzone.js

There were several setting options in Dropzone. In this project, six options were defined with specific values. The files uploading field is clickable and it allows the files under five megabits. The accepted file formats are image and PDF format. And it can only upload a maximum of thirty files at a time. The script also shows that the button whose id is “button”. This button is selected as the upload selector. It presents remove buttons under ever uploaded files. The Dropzone send all files name with content to the Django framework. After all the files are uploaded successfully, the project posts the web application to a successful page.

```
Dropzone.options.myDropzone = {
  clickable: true,          // The filed can be clicked for selecting
  files from drives
  maxSize: 5,              // The maximum file size, unit in MB
  addRemoveLinks: true,    // If will add a delete link for files
  acceptedFiles: ".png,.jpg,.jpeg,.pdf", // Indicate the file
  types allowed to upload
  dictDefaultMessage: 'Upload your files here', // Message
  shows in uploading field
  uploadMultiple: true,    // Indicate dropzone allows upload mul-
  tiple files at once
  parallelUploads: 30,      // How many files will be uploaded par-
  allel rocessing
  // Function binding
  init:function(){
    var self = this;
    // upload selector to match your button
    $("#button").click(function (e) {
      e.preventDefault();
      e.stopPropagation();
      self.processQueue();
    });
    // config the remove links buttons
```

```

self.options.addRemoveLinks = true;
self.options.dictRemoveFile = "Delete";
//New file added
self.on("addedfile", function (file) {
    console.log('new file added ', file);
});
// Processing
self.on("processing",function(){
    self.options.autoProcessQueue = true;
});
// Send file starts
self.on("sending", function (file, xhr, formData) {
    console.log('upload started', file);
    $('#meter').show();
    var data = $('#frmTarget').serializeArray();
    $.each(data, function(key, el) {
        formData.append(el.name, el.value);
    });
});
// File upload Progress
self.on("totaluploadprogress", function (progress) {
    console.log("progress ", progress);
    // Progress will present as percentage.
    $('#roller').width(progress + '%');
});
// After file upload complete, there is 999ms delay.
self.on("queuecomplete", function (progress) {
    $('#meter').delay(999).slideUp(999);
});
// On removing file
self.on("removedfile", function (file) {
    console.log(file);
});
// When all the uploading process completing,
// project will go to another page.
self.on("success", function(file, responseText){
    if (self.getUploadingFiles().length === 0 && self.getQueued-
Files().length === 0) {
        window.location.replace(responseText.url);
    }
});
},
}

```

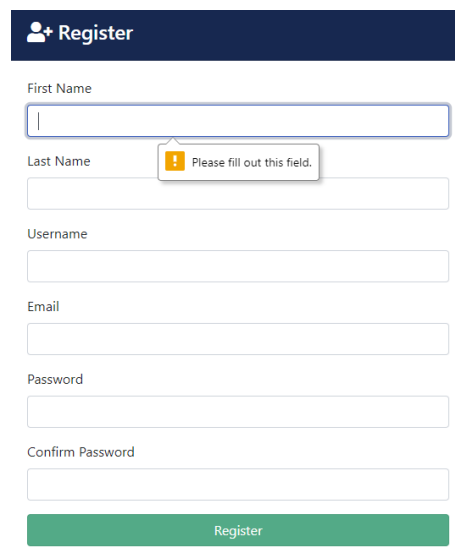
Code 13. Dropzone.js was initialized.

7 TESTING

This project is separated into five parts, which are the five pages of this project. These five pages are the register page, login page, create Directory Project page, file listing page, and validation page.

7.1 Registration Page

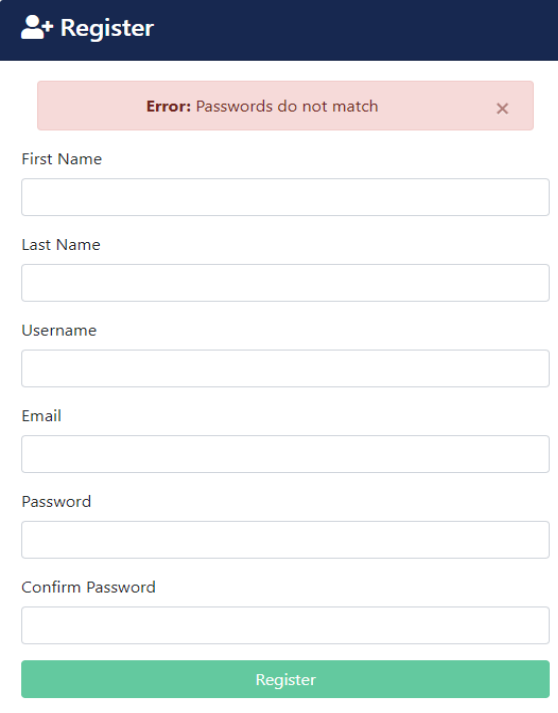
The register page allowed the users to create their accounts and the accounts information was saved into the MongoDB. The project ensured all the information fields be not empty by JavaScript as shown in Figure 31.



The screenshot shows a registration form titled "Register" with a dark blue header. The form contains six input fields: "First Name", "Last Name", "Username", "Email", "Password", and "Confirm Password". A yellow warning icon with the text "Please fill out this field." is positioned above the "Last Name" field, indicating a validation error. At the bottom of the form is a green "Register" button.

Figure 31. Test Register page with empty field.

The register page also would check that the password and confirm password was matched. The password would be posted by form request and be checked at the backend.

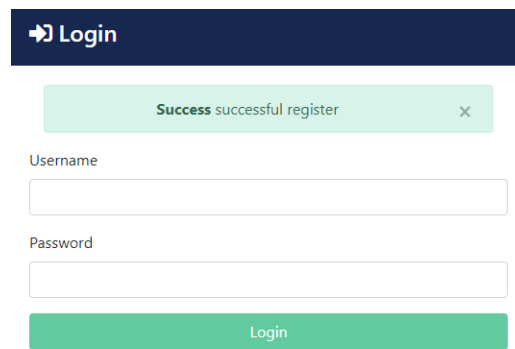


The image shows a web form titled "Register" with a dark blue header. Below the header, a red error message box displays "Error: Passwords do not match" with a close button (X). The form contains several input fields: "First Name", "Last Name", "Username", "Email", "Password", and "Confirm Password". At the bottom of the form is a green "Register" button.

Figure 32. Test Register page passwords do not match.

After checking the passwords, if the passwords do not match, the project would send back an error message to users. The project would also guarantee the username and email unused.

When the account was successfully registered, the project would jump to the login page and pop-up success prompt.

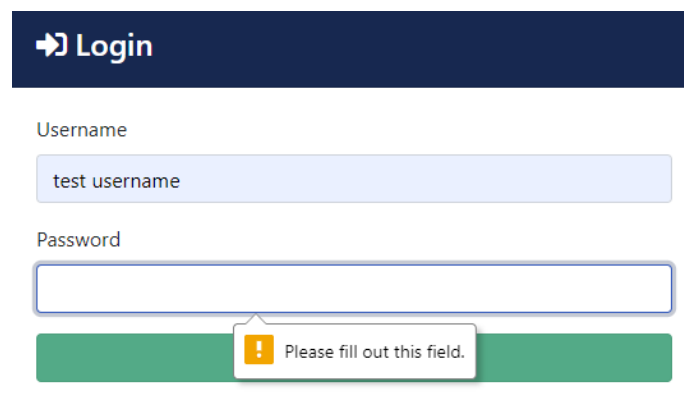


The screenshot shows a dark blue header with a white right-pointing arrow and the text "Login". Below the header is a light green success message box with the text "Success successful register" and a close button (X). Underneath are two white input fields: "Username" and "Password". At the bottom is a green "Login" button.

Figure 33. Test Register page successfully register.

7.2 Login page

The login page would also check all the fields not empty. The JavaScript guaranteed the fields at the front end as shown in Figure 34.



The screenshot shows the same dark blue header with "Login". The "Username" field is filled with "test username". The "Password" field is empty. A green "Login" button is at the bottom. A validation error message box with an exclamation mark icon and the text "Please fill out this field." is displayed over the empty password field.

Figure 34. Test Login page with empty field.

The project ensured the password and username be matched. It was achieved by the Django “auth” package. The package authenticated the account by transferring the username and password and check whether they were matched.

When the user successfully logged in, the project would jump to the dashboard page of this user.

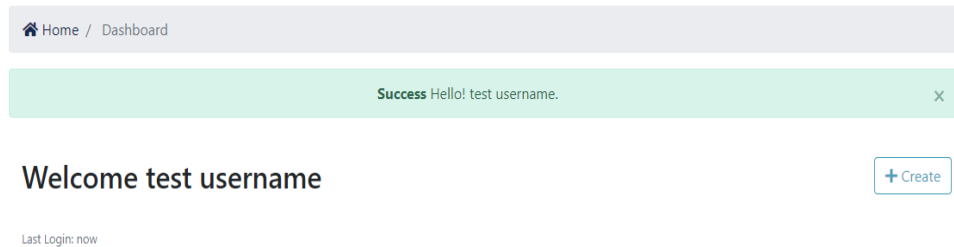


Figure 35. Test Login page successfully login.

7.3 Create Directory Project page

After logging in to the account, the user could create a Directory Project on the page shows in Figure 36. The user would be required to provide the Directory Project name and one or more files.

The form is titled 'Create a new Directory Project'. It has two input fields: 'Project Name*' with the value 'test' and 'Description' with the value 'test create Directory Project'. Below these is a dashed box containing two file upload cards. The first card shows '97.9 KB' and a 'Choose File' button. The second card shows '1.9 MB' and a 'Choose File' button. Each card has a 'Delete' button below it. At the bottom of the form is a dark blue 'Create' button.

Figure 36. Test Create Directory Project page.

7.4 File Listing page

On the file listing page, all files in one Directory Project were shown as preview mode. Users also could search the files by giving the file name and file type.

All files in one Directory Project were listing as in Figure 37.

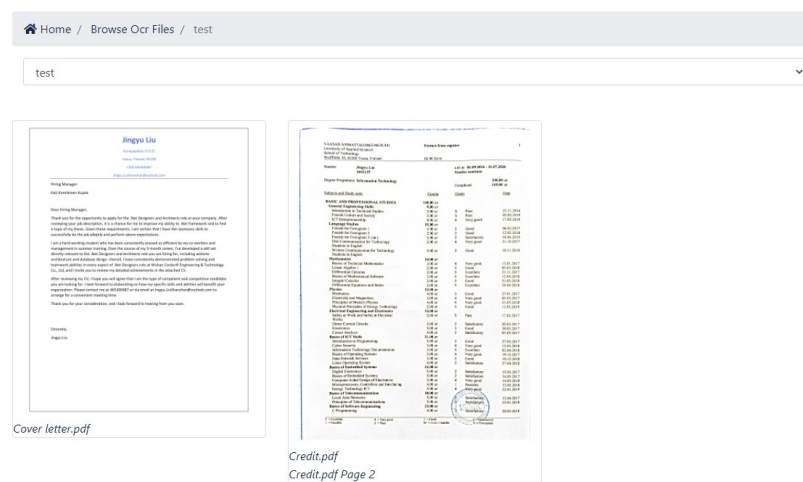


Figure 37. Test File Listing page shows all.

The page listed selected files by giving the file name in Figure 38.

Figure 38. Test File Listing page shows the searching result.

7.5 Validation page

The validation page would be reached after the user clicks one file which needed to be OCR. The project cropped the file with only a text area and listed all texts on the right side of the file. The user could edit wrong scanned words and copy all the texts into the clipboard.

Figure 39. Test Validation page.

8 SUMMARY

During the project, several problems were discovered. The first was that when users uploaded documents with the same file name, the program would only save the last document to the MongoDB. For solving this issue, the project renamed the duplicated document's file name by adding date and time automatically. The second was that the Dropzone setting codes could not be recognized in HTML files. For solving this problem, the setting codes were placed in the Dropzone source JavaScript file. The third problem was on the validation page. The recognized files were equally scaled. In this case, the original text coordinates were not correct which were extracted from full-size files. For solving this case, the project invoked the scaling ratio as parameters and recalculated the coordinates.

At present, some enterprises and personnel have already given some good solutions for Text Search Web Application. It includes Google Translate (Scanning files or taking pictures), Huawei mobile full-screen text recognition, and UiPath OCR. It helps themselves and their users much. There is no need to type all the texts from files, documents, and only a receipt. OCR technology is strong and powerful to face most cases; however, it still needs to be improved when the quality of the pictures is not satisfied.

In this project, the Django web framework and the Tesseract are introduced, and the whole process of building the Django web application. Besides, Python, JavaScript, MongoDB, and Docker are also introduced in this project. Many fields need to be improved or established in this thesis. At the same time, to make the Text Search Web Application project handier, it needs more knowledge from OCR technology and machine learning.

The web application allows users to upload multiple same documents. The application will rename the documents in the same content and file name. The accuracy of text recognition is very high even though the quality of the document is poor, or the background is dark. The project will crop the documents and shows only text fields to users which gives a better result for presenting the text searching of documents.

9 CONCLUSIONS

The project was implemented on the Wartsila Azure portal. The text searching functions, and files storage functions were well deployed. Users could upload multiple documents into one or more Project Directories and extract all the texts from the files. But the project still needed re-train the texts OCR models to achieve feedback functions. Which means that users could modify the wrong OCR fields.

The most challenging part of this project was manipulating the parameters of the Tesseract and OpenCV. It took much time for processing the image files well including grayscale the files, noise removal, sharpen kernel and thresholding.

9.1 Future work

In the future, the application needs to be unified with the front-end UI styles. And in this project, the recognized text can be manually fixed by users however it is only saved into the database rather than sent to the Tesseract for parameter optimization. It needs the Tesseract trained models being retrained once when users correct the wrong recognized fields. And need to enable users to outline the missing part of recognized texts in the pictures. Besides, the tesseract model should also provide the correct rate for separative words.

Besides, the purpose of the project is to create an application for Wärtsilä to automatically extract text information from thousands of documents. However, the application still needs users to upload the files and check the files one by one. It should complete these works automatically with only “one button” stuff.

REFERENCES

- /1/ Wärtsilä annual report. Accessed May 19, 2020. "https://wartsila-reports.studio.crasman.fi/file/dl/i/xrJERg/h3GfJgVK1kaXWKlix-TYG0A/Wartsila_Annual_Report_2019.pdf".
- /2/ Wärtsilä. 2015. "Financial-information". Accessed May 10, 2020. "wartsila.com/investors/financial-information".
- /3/ Mustone Vesa. 2020. "A new era of Process Automation Experts – Wärtsilä's Citizen Developers". Accessed May 10, 2020. "<https://www.wartsila.com/about/wartsila-185/view/a-new-era-of-process-automation-experts-wartsilas-citizen-developers>".
- /4/ Tesseract. 2019. "A comprehensive guide to OCR with Tesseract, OpenCV and Python". Accessed May 04, 2020. "<https://nanonets.com/blog/ocr-with-tesseract/#introduction>".
- /5/ Gjoreski, Martin & Zajkovski, Gorjan & Bogatinov, Aleksandar & Madjarov, Gjorgji & Gjorgjevikj, Dejan & Gjoreski, Hristijan. 2014. Optical character recognition is applied to receipts printed in the Macedonian language. 10.13140/2.1.1632.4489.
- /6/ Djangostars. 2016. "Why we use Django framework". Accessed May 07, 2020. "<https://www.djangostars.com/blog/why-we-use-django-framework>".
- /7/ Django project. 2015. "Django overview". Accessed May 07, 2020. "<https://www.djangoproject.com/start/overview/>".
- /8/ Stackshare. 2014. "Django - Reviews, Pros & Cons Companies using Django" Accessed May 13, 2020. "<https://stackshare.io/django>".
- /9/ Data Flair. 2019. "Django architecture – 3 major components of MVC pattern". Accessed May 7, 2020. "<https://data-flair.training/blogs/django-architecture>".
- /10/ Django project. 2020. "Documentation, Class-based views". Accessed May 10, 2020. "<https://docs.djangoproject.com/en/3.0/topics/class-based-views/>".
- /11/ Wikipedia. 2004. "JavaScript Wikipedia". Accessed May 13, 2020. "<https://en.wikipedia.org/wiki/JavaScript>".
- /12/ Wikipedia. 2005. "Ajax (programming) Wikipedia". Accessed May 13, 2020. "[https://en.wikipedia.org/wiki/Ajax_\(programming\)](https://en.wikipedia.org/wiki/Ajax_(programming))".

- /13/ Wikipedia. 2009. “MongoDB Wikipedia”. Accessed May 13, 2020. “en.wikipedia.org/wiki/MongoDB”.
- /14/ MongoDB docs. 2017. “Terminology and Concepts”. Accessed May 13, 2020. “<https://docs.mongodb.com/manual/reference/sql-comparison/>”.
- /15/ Andy_Lee. 2018. “Concepts of Docker Article”. Accessed May 13, 2020. “<http://dockone.io/article/6051>”.
- /16/ Docs of Python. 2012. “Command line and environment”. Accessed May 13, 2020. “<https://docs.python.org/2/using/cmdline.html#cmdoption-u>”.
- /17/ Zhihu. 2018. “Python—Django framework design ideas”. Accessed August 24, 2020. “<https://zhuanlan.zhihu.com/p/43833483>”.
- /18/ XpertUp. 2020. “Document Scanner and OCR Overview”. Accessed August 24, 2020. “<https://www.xpertup.com/downloads/document-scanner-and-ocr/>”.
- /19/ RUNOOB. 2015. “Python Synopsis”. Accessed September 10, 2020. “<https://www.runoob.com/python/python-intro.html>”.