CERTIFICATION OF APPROVAL

Web App for Tools Inventory Management with Predictive Categorization

By

Muhamad Hamzah bin Razali

25571

A project dissertation submitted to the

Information System Programme

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF INFORMATION SYSTEM (Hons)

Approved by,

[illegible]

24 March 2022

Dr. Toni Anwar
Assoc. Prof.
Computer and Information Sciences
Universiti Teknologi PETRONAS

(Assoc. Prof. Ts Dr Toni Anwar)

UNIVERSITI TEKNOLOGI PETRONAS TRONOH, PERAK

January 2022

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2022

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work that is in this report has not been undertaken or done by unspecified sources or persons.

[illegible]

Muhamad Hamzah bin Razali

ABSTRACT

The title of this project ‘Web App for Tools Inventory Management with Predictive Categorization' is proposed by Mr. Muhamad Hamzah bin Razali. The main purpose of this project is to develop a web application namely ‘Drillclinic’ that can digitalize inventory management process for tools management, by having predictive tool categorization and assigning Data Matrix code to each real-world Tool object.

A tool sharpening company is interested to have a web-based system to store information and assign a Data Matrix code to each tool object for them to get info by scanning the code. Whenever new tools enter their database, they want to have automatic categorization once their parameters are already measured. This would reduce time taken for their business process.

As a result, the literature review discusses research on Data Matrix code technology as well as tool categorization. It would help us gain better understanding the characteristics of tools and Data Matrix technology as the scope for this project. A raw dataset of tool categorization has been provided from the company for it to be analyse, but it is generalized and simplified for this project to protect companies’ intellectual property. With the combination of machine learning knowledge and information as well as Data Matrix technologies, this proposed system hopes to bring significant improvements to their business operation.

Hence this project, aims to bring all these elements together into a dynamic web framework that allows users (specifically Product Receiver staffs) to build a customer repository, manage and edit Tools information as well as having predictive tool categorization, all within the same system. The chosen methodology for this project is the waterfall methodology with reason, to achieve phase by phase development with result indirectly reduce the risk of project failure. As a junior developer, it is best to select this methodology as objectives and expectation can easily planned phase-by-phase. This document thoroughly describes the feasibility studies, literature review, project methodologies and result of the project.

ACKNOWLEDGEMENTS

Firstly, I am very grateful to God for giving me health and wellbeing that I’m finally able to complete this project.

I would like to express my gratitude to my former supervisor during my internship, Dr.-Ing. Mohanad El-Haji for providing me a generic dataset from a tool maintenance company for me to analyse for my project purposes. He also emphasizes me to always be pragmatic and never overengineer your work. Those ideology and mindset are what I applied for this project in order to fulfil the requirements on time.

Additionally, I would also extend my thanks to Assoc. Prof. Ts Dr Toni Anwar for his patience, tolerance, and knowledge in advising me through my project for providing support and encouragement for the project I am doing, and for assessing my project progress. His knowledge and sharing have been very helpful in making progress in this project. I also would thank Assoc. Prof. Said Jadid Abdul Kadir as the course coordinator for final year project that help me and fellow colleagues to follow project guidelines, provide adequate references and specifies the submission requirements thoroughly.

Not forgetting want to convey my heartfelt thanks to my course mates, no matter senior or junior who help me to revise and apply the learned knowledge for this project. I was fortunate to have beneficial network with talented and skilled people in my life. This project is unlikely to succeed without dedicated people in the university. The university deserves a tremendous appreciation.

Finally, I would like to thank my family for their undying spiritual and mental support throughout the project and in general. They understand my limitation and provide necessities for me to continue finishing my project.

Table of Contents

Abbreviations and Nomenclatures

Item/Part/Tool : A real world object (tool/bits) that is will be engraved by Data Matrix code that is assigned to it in this web application system

Tool category: A category that groups together multiple tools based on the tool's parameter values. For example, drillers, mills, cutters or countersink.

Customer: The tool owner by with company name. Important to have this info, to identify which tool belongs to who and to send back the tool together in a correct batch.

Task: Tools have to be processed in a certain way defined by their category (e.g. sharpening, harden, fabricate). After processing the item is sent back to the owner / customer.

Static parameters: Item parameters that are given by the item's existence and do not change during the lifetime of the item (e.g. name and Data Matrix code of the drill bit)

Dynamic parameters: Item parameters that change during the lifetime of the item due to usage of the item, the customers will request a processing of the item, to achieve a certain parameter value of these dynamic parameters (e.g. cutting edge radius of the drill bit; it gets bigger due to usage of the item, then the customer will send a request to sharpen the drill bit again, this requires a certain value changes of the cutting edge radius).

1. INTRODUCTION

1.1 Background study

For this project, I wanted to contribute a prototype application for my previous company that I did my internship with, EDI GmbH - Engineering Data Intelligence, Karlsruhe, Germany. This application is based on one of the requests to propose a web application for Tool Inventory Management System for a tool sharpening company. Listed below is a quick summary of the business process by that company.

- Expectation

They know each single tool (Category, ID, etc) and identify every incoming tool from their clients (customer). They know how often this part was already in their company, they know how to process this part in their company. They know how much it costs and They have all billing information.

- Reality

At present, they identify tool parts as the customer sent to them manually from the catalogue, maintained by Product Receiver Department. Human error may occur due to a lack of information and experience. Every time the part comes, they have to do the same thing again. All information has to be shared manually, even using paper with bar code on it.

- Consequences

They are unable to deliver product to be replaced or repaired as expected at the best time. Because of this, their customer may get a delay or low-quality rushed products, which will adversely affect customer satisfaction. Besides, searching through the catalogue for the right product decreases productivity because it takes a lot of time. Single part of information is not updated concurrently into different catalogues

1.2 Research gap / Problem Statements

The problem statement is a description of present difficulties that must be resolved by the end of the project evaluation. The following are the issues that have arisen as a result to initiate this project:

1. Manual catalogue check

A medium business company who main business is to sharpen or coat industry tools from their client, was having issues of manually checking and updating information of all tools sent by their client companies. The company's Product Receiver Department, a cataloguer, arranges customer's tool sent to them from logistics by manually checking from the catalogue. He/she must refer and update multiple excel sheets to obtain tool parameters, tool owner information for one specific tool. This takes too long for them to identify the product, searching for related catalogues that stores the product categorization and parameters thus having significant delay in the process and eventually their business operation.

2. Information are outdated

They also struggle to have updated catalogue due to taking note manually if products have some special remarks (e.g. coating properties, special requests from customers etc.) especially the same product that have been sent to them before for repair. On top of that, new recruiters in the company are having trouble completing the process without committing mistakes.

This company is interested to have a web-based system to store information and assign a DataMatrix code to each Tool object for them to get info by scanning the code. New tools can have automatic categorization once their parameters are already measured.

1.3 Objectives & Scopes of Work

Project scopes are frequently used to determine a project's focus and to make the development process more organised and systematic.

1.3.1 Objectives

The main goal of this project is to create an application that will make it easier to tackle all of the challenges associated with tool management. As a result, the following are the objectives for achieving the goal:

- To have an Inventory Management web-based application for tools to store, manage and update their information together with parameters.
- To generate and assign each tool with a Data Matrix code in the system for them to engrave in real world object, Tools.
- To have a predictive classification for the tool category, whenever users have entered all required tool parameters

Businesses can reduce their time required by staff to search for information and identifying items. This ID will be used to show and store each item's information. The items will be categorized based on their parameter values. Further semantical links can be connected to each item (e.g. customer information in order to know which item belongs to who). This app will help the company to show the Tool's information without having to manually search the hardcopy information since it is defined and stored in the database.

The following is a short storyboard to summarize the change is business operation before having the application and after having the application.

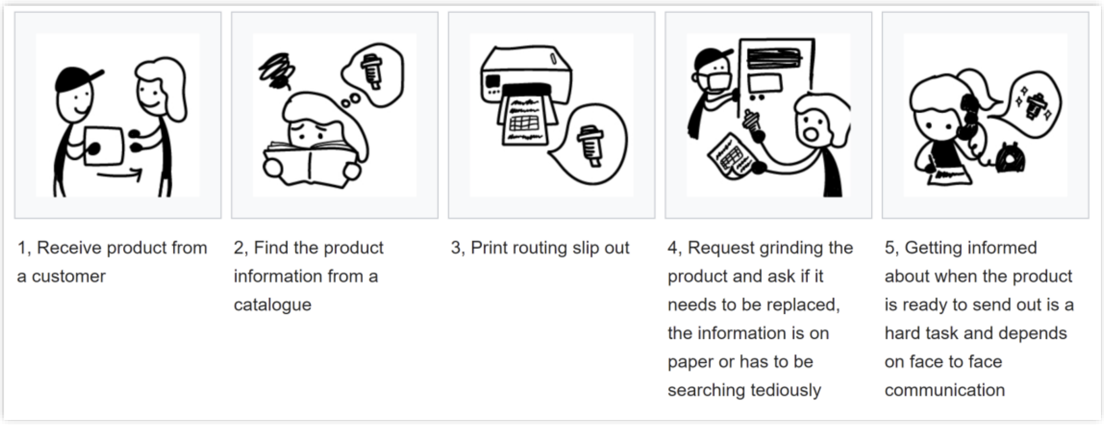

1, Receive product from a customer
2, Find the product information from a catalogue
3, Print routing slip out
4, Request grinding the product and ask if it needs to be replaced, the information is on paper or has to be searching tediously
5, Getting informed about when the product is ready to send out is a hard task and depends on face to face communication

Figure 1. Before having the application

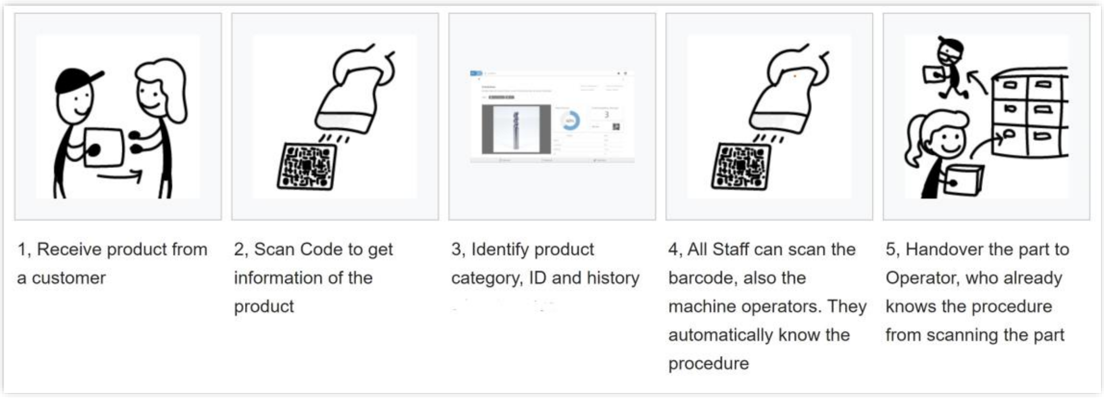

1, Receive product from a customer
2, Scan Code to get information of the product
3, Identify product category, ID and history
4, All Staff can scan the barcode, also the machine operators. They automatically know the procedure
5, Handover the part to Operator, who already knows the procedure from scanning the part

Figure 2. After having the application

1.3.2 Scope of Study

The development of web-based application prototype for inventory management with the given name ‘Drillclinic’ would be the scope of this project. End user for this app are Product Receiver Staff and Technician who will fabricate or sharpen the tools. They need the application to provide latest information by scanning Data Matrix code.

In order to have predictive classification of tools, we have to apply machine learning algorithm to analyse the dataset given and that would be the second scope of the project.

2. Literature Review

This literature review studies on previous research conducted about Data Matrix code technology and industrial tools categorization which are related to this project. This is done to assist in determining the best potential strategy and procedures for determining the project's feasibility. Despite given the requirements, there's a possibility to acknowledge more information about this technology to understand it's possible limitation and risk.

2.1 Data Matrix code

Two-dimensional (2D) matrix codes are becoming more widespread and are increasingly replacing standard linear 1D barcodes in many spheres of life. The data capacity (the ability to store more data in a smaller space) and error correcting capabilities (through the Reed-Solomon algorithm) are the two most significant benefits of 2D codes. A 2D code applied to a product may hold extensive information on the product, the manufacturer, the receiver, and the customer, and so has applications in manufacturing, inventory, distribution, sales, and maintenance.

Black and white modules (also known as cells) are used to create two-dimensional matrix codes, which are commonly organised in a square pattern. The smallest construction block is represented by one module, and in the data region, the dark module commonly encodes binary 1 and the light module binary 0. The number of modules (rows and columns) rises as more data is encoded in a 2D code.

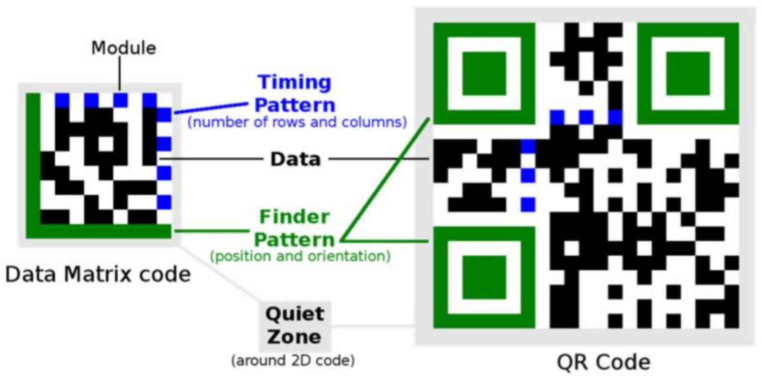


Figure 3. Data Matrix code features

Each sort of 2D code contains distinct fixed elements that are utilised to establish the code's location and orientation (Finder Pattern) as well as its dimensions (Timing).

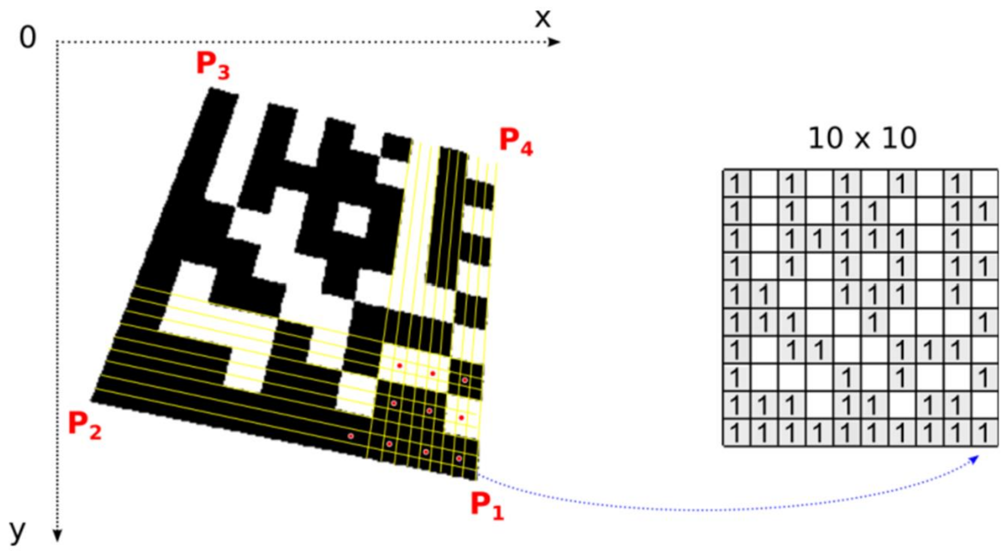
The most frequent forms of 2D codes are Data Matrix and QR Codes. Because they need less area to contain the same amount of data, Data Matrix codes are often employed to designate tiny goods such as electrical components. According to article by Ladislav Karrach and Elena Pivarčiová[1], here are a short comparison between Data matrix code and QR code.

Feature	Data Matrix Codes	QR Codes
Size	From 10 * 10 to 144 * 144 modules (increment by +2)	from 21 * 21 to 177 * 177 modules (incremented by +4)
Capacity	3116 numeric or 2335 alphanumeric or 1556 bytes	7089 numeric or 4296 alphanumeric or 2953 bytes (at lowest error correction level)
Error Correction Level	one level up to 30% damage	four configurable levels: L (Low): 7%, M (Medium): 15%, Q (Quartile): 25%, H (High): 30%
Finder Pattern	“L” shaped on the edge	at three corners of a QR Code. Each Finder Pattern is formed by an inner dark square surrounded by a dark frame
Timing Pattern	alternating dark and light modules at the edge	alternating dark and light modules placed inside a QR Code and interconnecting Finder Patterns
License	public domain	public domain

Table 1. Comparison Data Matrix code with QR code

Decoding the code.

We may translate picture points from the Data Matrix code in the image to the square binary matrix after the exact position of the four corner points, which constitute the bounding quadrilateral, is known, perspective translation is set up, and the number of rows and columns is defined (Figure 4). Modules that are dark must be mapped to binary 1 and binary 0 for light modules. The data recorded in the Data Matrix Code is decoded using this binary matrix as the input to a deterministic procedure.



1		1		1		1		1	
1		1		1	1			1	1
1		1	1	1	1	1		1	
1		1		1		1		1	1
1	1			1	1	1		1	
1	1	1			1				1
1		1	1			1	1	1	
1				1		1			1
1	1	1		1	1		1	1	
1	1	1	1	1	1	1	1	1	1

Figure 4. Data Matrix code converted from the picture domain to a binary matrix.

To decode the binary matrix, the opensource libdmtx software [2] is used, which restores the original text contained in the Data Matrix code. In a genuine picture, a module is frequently made up of multiple points. The module's dark or bright classification is determined by one centre point (red points in Figure 4 represent the central points of the modules).

Light is defined as points with an intensity above the threshold, while dark is defined as points with an intensity below the threshold. The average grey intensity in the Data Matrix coding area is used as the threshold. In addition to the centre point, we may incorporate points in its immediate surroundings in the decision-making if the modules are made up of more than 5*5 points.

2.2 Two-dimensional (2D) matrix engraved on Tool

A pilot test conducted by the company to see the ability of an engraving machine to print several types of two-dimensional (2D) matrix code on a drill bit. The below pictures are the outcomes of their test.

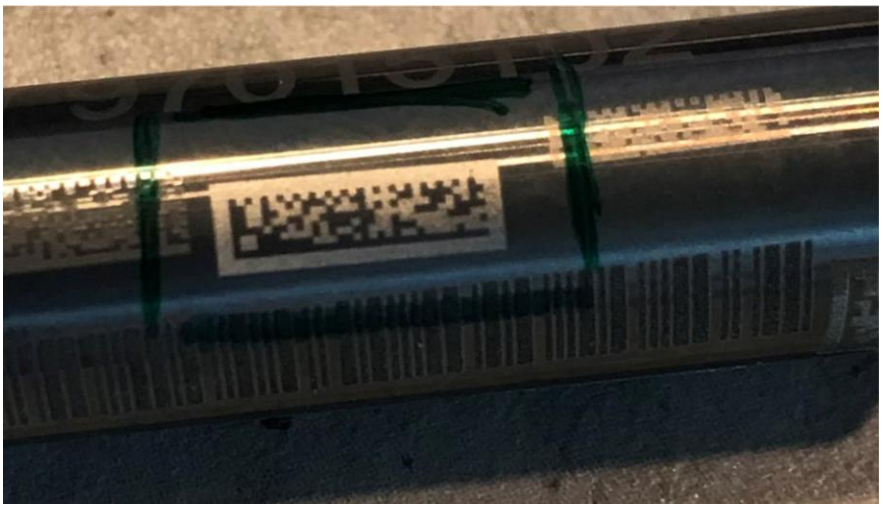


Figure 5. Other 2D code engraved

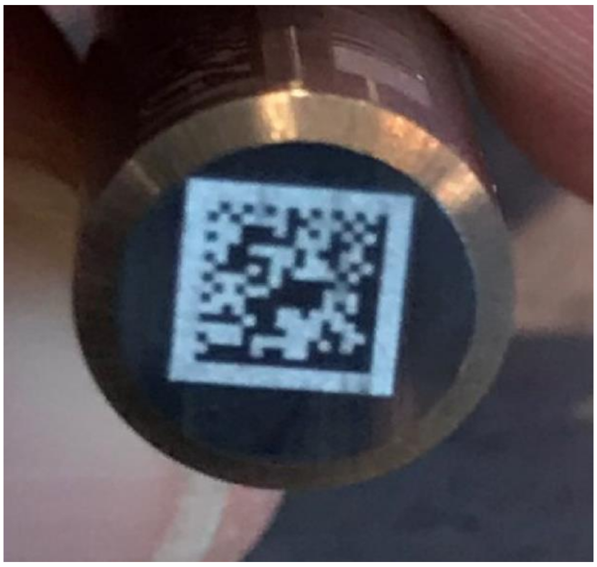


Figure 6. Data Matrix code engraved on a drill bit

They found out Data Matrix code have lesser risk of deteriorating and easier to be detected by their scanner compared to other 2D codes. Since it’s safer to engrave at the top of the drill bit, it is also helps to avoid reflective rays when attempting to scan on rounded objects.

3. Methodology/Project Work

Methodology explains the part of how the project going to be implement from an idea into a working prototype with the intention to achieve the project goals in time. It consists of multiple processes and discussions such as Requirement Gathering, Project Timeline (Gantt chart), System and UML Designs, User Interface sketches, Application Interface, Software Development (SDLC) and System Integration Test.

Since this project involved data mining process, I decided to use CRISP-DM approach as it is the most common and standard process for data mining. It has multiple phases to ensure dataset acquired can be utilized according to project objectives, more details will be explained later in this chapter.

3.1 Requirement Gathering

3.1.1 Functional requirements

Features	Description
Data Matrix Code	Generate a unique Data Matrix code image for one Tool. This will be scanned to show the overview page of the tool in the web application
Predictive Classification	Have an automatic Tool type classification once all required parameter for a Tool has been inserted
Data Storage	Able to retrieve or store object information whether it is Tool or Customer data.

Table 2. Functional Requirement

3.1.2 Non-Functional requirements

Features	Description
Security	Different user permission for specific functionalities separated by User Groups. There are only (excluding admins) 2 user groups, one is Receiver, and another is Technician

Table 3. Non-functional Requirement

3.2 Project Timeline (Gantt Chart)

3.2.1 FYP1 Gantt Chart

	Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
1	Proposal Title	🟩	🟩										
2	Project Background and Requirement Gathering		🟩	🟩	🟩								
3	Literature Review		🟩	🟩	🟩	🟩	🟩						
4	Research, Technologies			🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩		
5	Sketch UI							🟩	🟩	🟩			
6	Proposal Defense									🟩			
7	Interim Report									🟩	🟩	🟩	🟩

Table 4. FYP1 Gantt Chart

3.2.2 FYP2 Gantt Chart

	Task	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12
1	Development of Application	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩				
2	Data Mining & Modelling				🟩	🟩	🟩	🟩	🟩				
3	Integration Test & Debug					🟩	🟩	🟩	🟩	🟩			
4	Draft Final Dissertation					🟩	🟩	🟩	🟩	🟩	🟩	🟩	
5	Dissertation Submission												🟩

Table 5. FYP2 Gantt chart

3.3 SDLC

This project will practise the Waterfall SDLC approach since it is the simplest methodology for junior software developer as it follows a sequential flow phase by phase approach. In terms of development, we should not underestimate how complicated the project could be. Hence, to avoid any complication during web development or chaotic progress, it is why this approach is chosen.

3.4 System Architecture

The framework that will be used in this project is Django Web Framework. It is a server-side web framework built in Python that is famously used in industries as it has lot of features and versatile configurations. For the project whole architecture using Django as a framework, it's common to have Models, Views, Template (MVT) files in the library, it can be represented in below diagram:

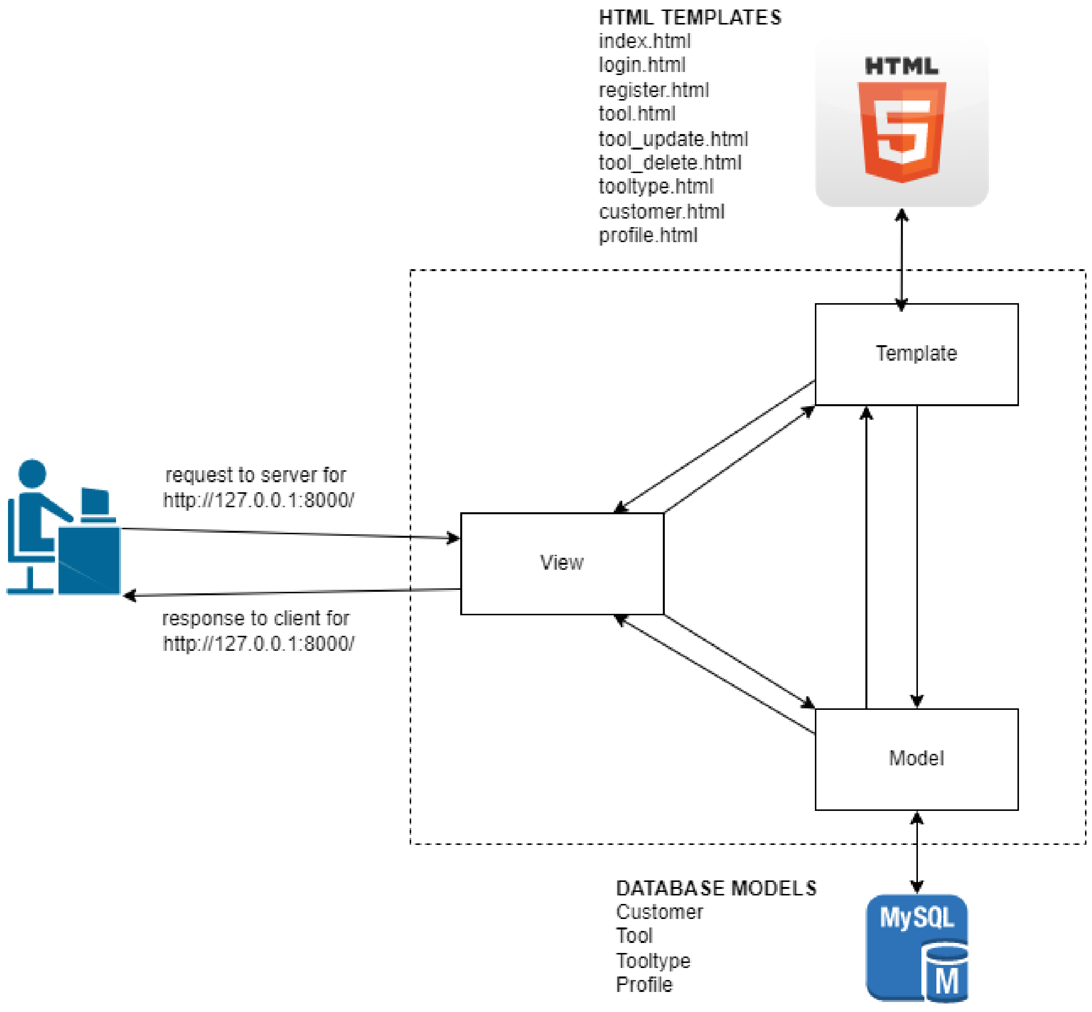


Figure 7. System Architecture

In Django, files can be separated according to diagram above to identify which files are related when performing requests. Django can store data internally in a file using the SQLite database and can be assessable via the Django admin panel. Objects can be created and saved by either using the admin panels or there's form rendered in the Views. Django will automatically perform the tedious coding for communicating with the database for you.

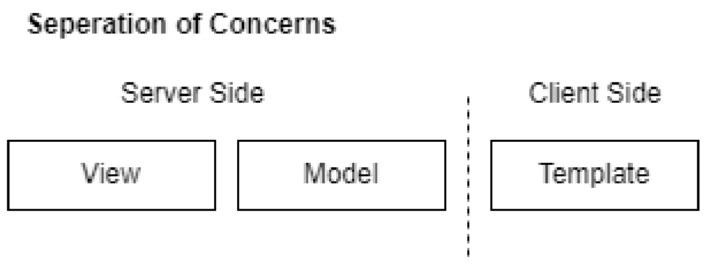


Figure 8. Separation of Concerns

This figure shows the separation of concern during implementation. At the server side, the View files is used to implement the logic such as to add an object into the database. There will be communication between the View and Model files where it contains the database model, for example show/hide data, add/delete data. Whereas, template files will be rendered in client side, which means client can only view the HTML files template.

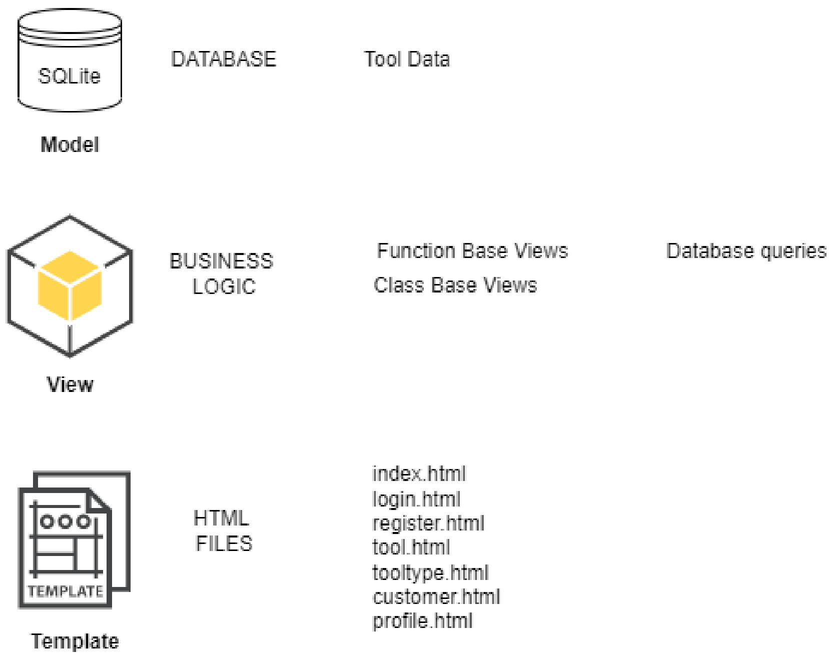


Figure 9 Django architecture.

In the above diagram, models are going to represent the database. Creating an object for example Tool object, will be added into the database model. For Views it consists of two types, Function Based Views, and Class Based Views. It handles the logic that needs to be implemented, and there will be a correlation between Views and Models to access the database. Every defined property is a column in the database model.

3.5 System Design using UML

Unified Modeling Language (UML) is a current approach to software modelling and documentation. In fact, it's one of the most widely used methods for modelling business processes. It is based on representations of software components in diagrammatic form. Non-developers and developers could help clarify potential misinformation or problems in software or business processes by employing visual representations.

3.5.1 Use Case Diagram

The following is a generic use case diagram for this project showing all the core functionalities. There are different actors having different privileges, this is set by user groups (user roles) that has defined permissions by the admins.

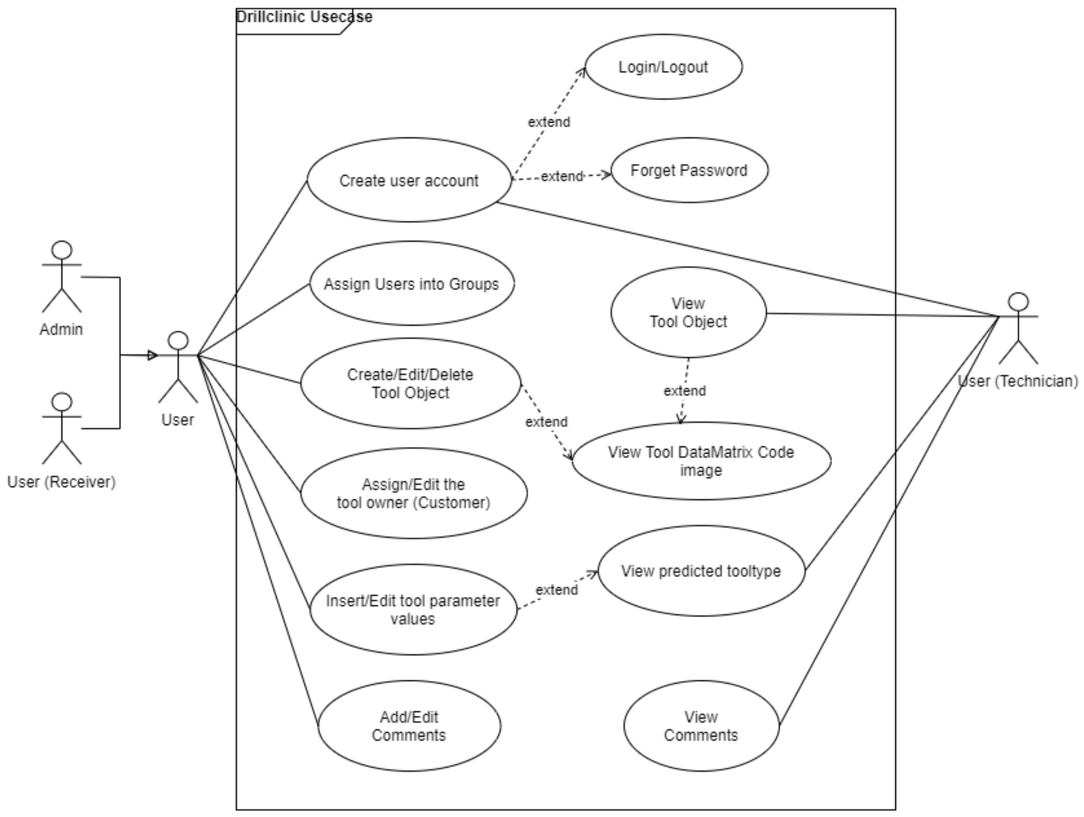


Figure 10. Use Case Diagram

3.5.2 Activity Diagram

The following is a generic activity diagram summarizing the project flow. The user (receiving dept) is the main actor for this application hence will undergo a lot of functionalities in this application.

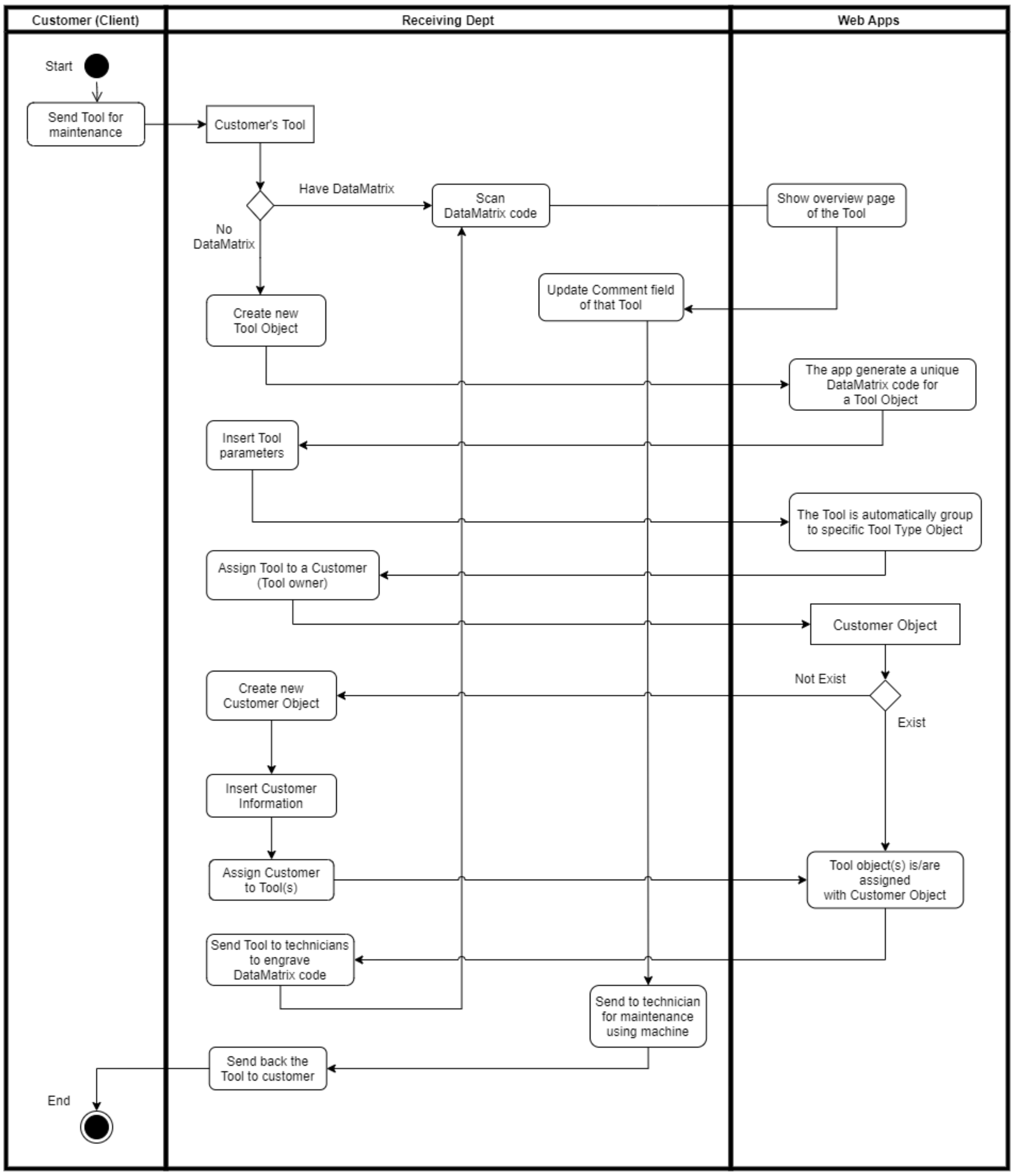

Customer (Client)
Receiving Dept
Web Apps
Start
Send Tool for maintenance
Customer's Tool
Have DataMatrix
No DataMatrix
Scan DataMatrix code
Show overview page of the Tool
Update Comment field of that Tool
Create new Tool Object
The app generate a unique DataMatrix code for a Tool Object
Insert Tool parameters
The Tool is automatically group to specific Tool Type Object
Assign Tool to a Customer (Tool owner)
Customer Object
Not Exist
Exist
Create new Customer Object
Insert Customer Information
Assign Customer to Tool(s)
Tool object(s) is/are assigned with Customer Object
Send Tool to technicians to engrave DataMatrix code
Send to technician for maintenance using machine
Send back the Tool to customer
End

Figure 11. Activity Diagram

3.5.3 Class/Domain Diagram

The diagram below will represent a generic domain diagram showing all the relationships and cardinality between classes for this application. There is a lot more classes that can be specify in this diagram including Django classes, but it's generalized for more understanding of the core applications.

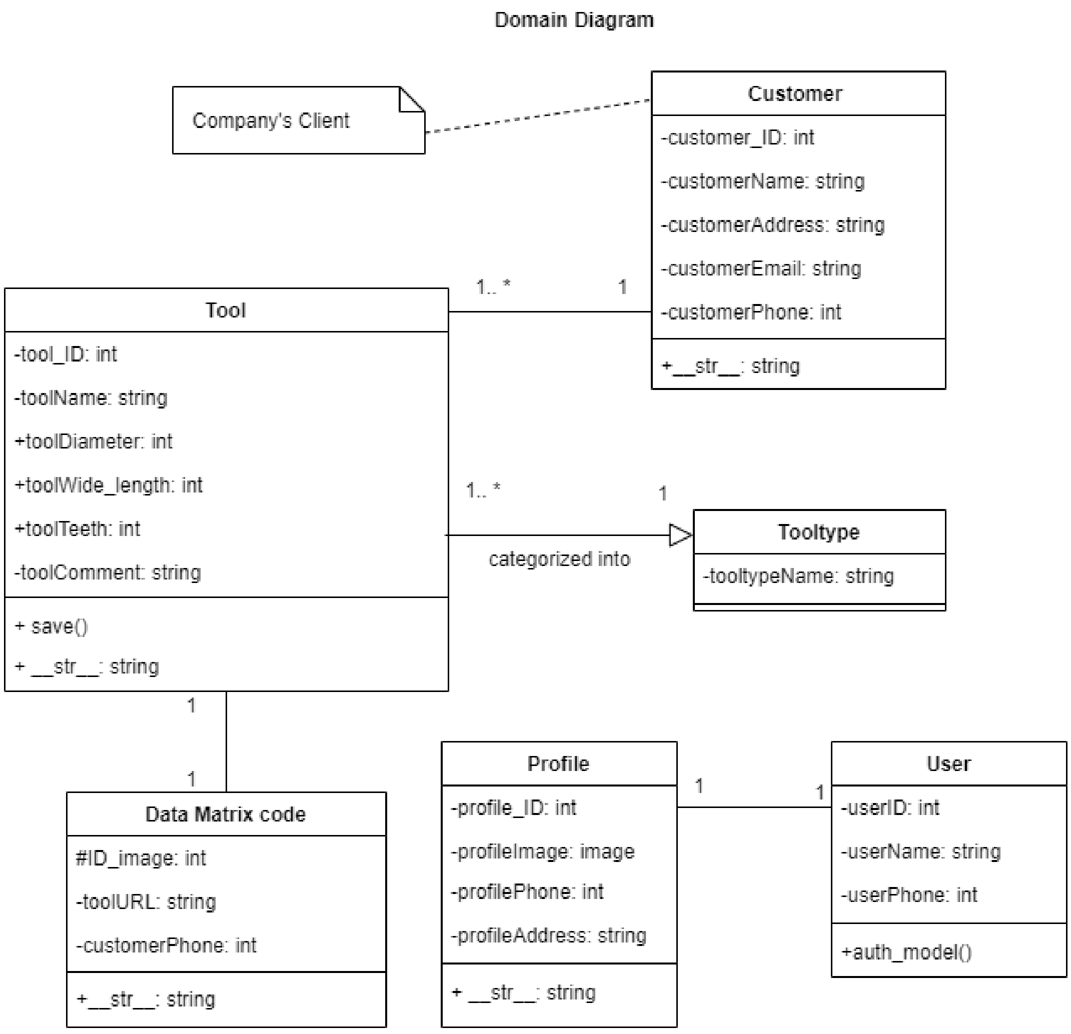


Figure 12. Domain Diagram

I decided to add user Profile class with one-to-one relationship with User app once they successfully register in the application. Having user profile will be easier for Receiver Dept to assign the user into certain User Groups in the front-end, although it can also be done through the admin interface. The Data Matrix code is separated class from Tool class as it is called from a different library and model in Django. Within each class have its own class/model name, field names, data types, methods, and their return types. More details can be seen in the code itself.

3.6 System Interface

Whenever this application is ready to be used by the company, the table below shows a summary of the interface interaction between this web application and real-life business operation and object from that company. The company have their own Data Matrix code scanner and engraving machine, will be used for the following tasks:

Item	Description	Data sent to this interface	Data received from this interface
Engraving machine Computer	Folder on this computer that also runs the Machine to engrave Data Matrix on each Item using software in that machine	300dpi resolution image Data Matrix	Tool engraved with Data Matrix
Scanner	To scan the engraved Data Matrix and get that Tool URL to display info in browser	Data Matrix code	The Tool object overview page

Table 6. System Interface

3.7 Data Mining for Predictive Analytic

This project will follow the Cross Industry Standard Process for Data Mining (CRISP-DM) approach as it is the most common and standard process for data mining. It has multiple phases to ensure dataset acquired can be utilized according to project objectives. Predictive analytics take historical data from existing database, then the suitable algorithms, statistical models, and machine learning are deployed to capture patterns in the existing dataset. The following will elaborate the stages of CRISP-DM and how it will be applied for this project.

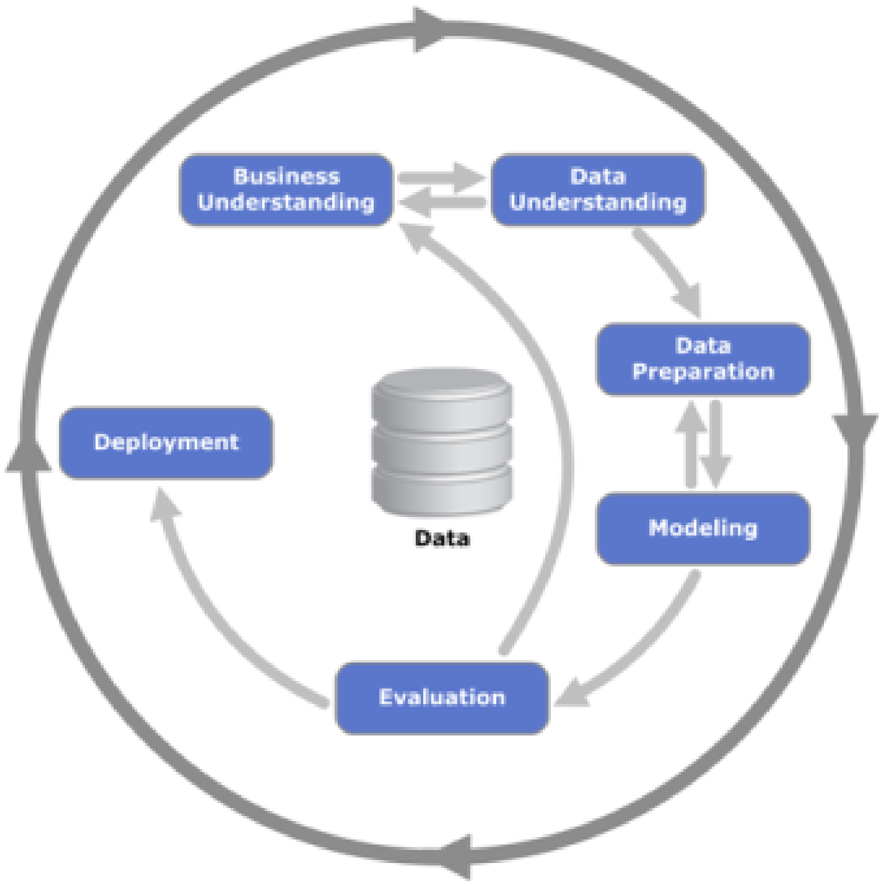


Figure 13. CRISP-DM

3.7.1 Business Understanding

The Business Understanding phase focuses on determining the project's goals and needs. The tasks in this phase are standard steps in software industries to initiate project management activities that apply to almost all projects. While many organizations rush through this step, developing a solid business knowledge is important to oversee the reasons behind it. To relate these tasks with the project, the requirements are gathered thoroughly with the company and information and business objectives are specified and acquired.

1. **Establish business objectives:**

   Thoroughly grasp, from a business viewpoint, what the client wants to achieve. and then create success criteria for the organization. The business objectives are stated previously in Chapter 1 having the same as project objectives.

2. **Establish data mining objectives**:

   In addition to identifying business goals, we should also describe what are the success criteria looks like in terms of technological data mining. The aim of data mining for this project is to have predictive classification for Tool type according to Tool parameters assigned. The accuracy score for the prediction is supposed to be more than 80% accuracy.

3. **Create a project plan:**

   Choose technology and tools, as well as specific plans for each step of the project. The environment for this project, to perform data mining is using Google Colab. It is similar to Jupiter Notebook with the same coding language, Python to execute the code. More in depth code explanation in further section.

3.7.2 Data Understanding

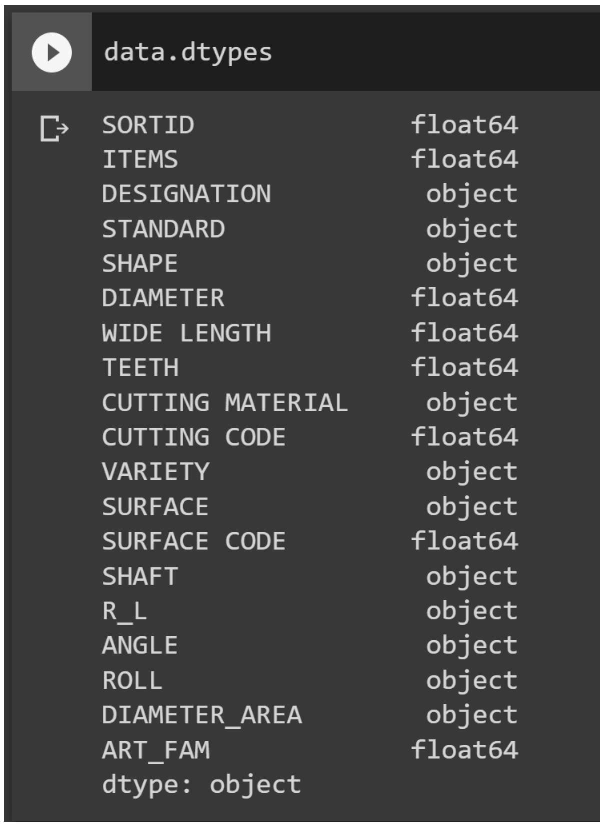
This stage generally focuses on identifying, collecting, and analysing data sets that might assist to meet the project objectives. There are several tasks in this phase:

1. **Collect preliminary data:**

   For this task, the dataset is collected from my former internship company, EDI GmbH. The dataset is modified and further generalized to protect companies' privacy policies. The dataset is originally coming from one of the clients of EDI who specialized in Tool maintenances. My former supervisor, Mohanad gave the opportunity to analyze the dataset that is beneficial for this project. In return, I have to provide several findings and knowledge based on data mining processes that may be beneficial to the company and provide business insight.

2. **Describe the information:**

   The file is originally stored using Microsoft Access. It is further modified and exported as excel file for this project. The dataset received is actually a partially full catalogue of all Customer Tools that the company receives. It contains information such as tool standards, tool designation (Tooltype), diameter, wide length and several more measurable parameters for that Tool. The following are the datatypes of each column extracted by executing a python code in Google Colab. The size of the dataset are 55251 rows × 19 columns



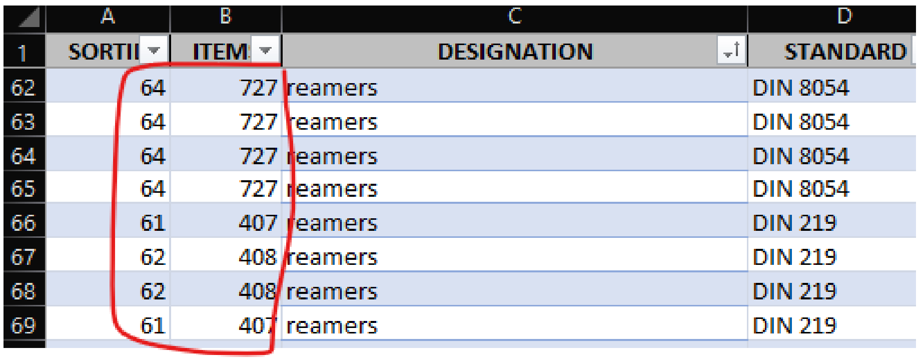
```
data.dtypes
```

```
SORTID                float64
ITEMS                 float64
DESIGNATION            object
STANDARD               object
SHAPE                  object
DIAMETER              float64
WIDE LENGTH           float64
TEETH                 float64
CUTTING MATERIAL       object
CUTTING CODE          float64
VARIETY                object
SURFACE                object
SURFACE CODE          float64
SHAFT                  object
R_L                    object
ANGLE                  object
ROLL                   object
DIAMETER_AREA          object
ART_FAM               float64
dtype: object
```

Figure 14. Data types of the dataset

3. **Investigate the data:**

   After meticulously investigating the dataset, there are several linkages can be assumed between different columns, as they provide the same meaning for one entry. For example, the column name “SORTID” and “ITEMS” carry the same meaning. There are no differences between these two columns, therefore it wouldn’t be needed for data mining.



	A	B	C	D
1	SORTII	ITEM	DESIGNATION	STANDARD
62	64	727	reamers	DIN 8054
63	64	727	reamers	DIN 8054
64	64	727	reamers	DIN 8054
65	64	727	reamers	DIN 8054
66	61	407	reamers	DIN 219
67	62	408	reamers	DIN 219
68	62	408	reamers	DIN 219
69	61	407	reamers	DIN 219

Figure 15. Columns having the same information

4. **Examine the data for quality:**

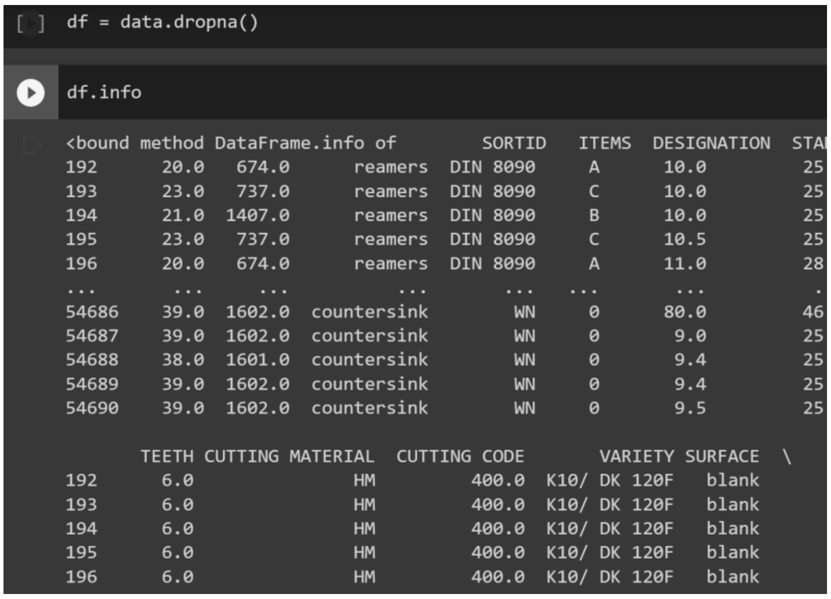
   The dataset unfortunately were very "dirty" in the sense, different language being used (German). What is worse is that the columns names appears to be in shortforms. Therefore, manually translating from German to English for the column names sometimes results inaccurate meaning from the original word. This can be assumed by the values/entries that a column has.

3.7.3 Data Preparation

This step, sometimes known as data munging or pre-processing data is responsible for preparing the final dataset for modelling. It has few tasks:

1. **Clean data:**

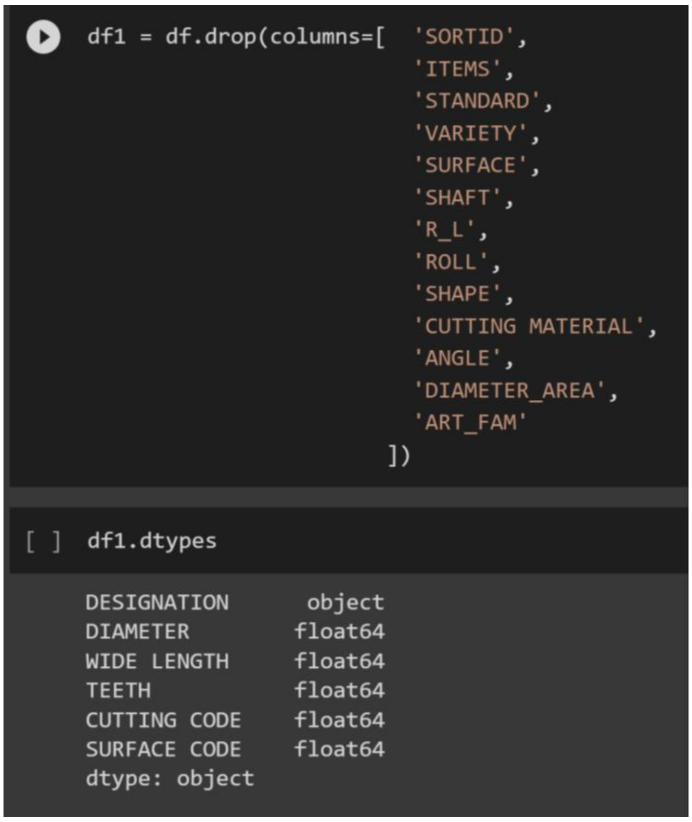
   This is often the most time-consuming operation. Erroneous data are often corrected, imputed, or removed throughout this job. The common first step is to drop all rows that contain at least one NA values.



```
[ ] df = data.dropna()

df.info

<bound method DataFrame.info of        SORTID  ITEMS  DESIGNATION  STA
192       20.0   674.0       reamers  DIN 8090     A      10.0        25
193       23.0   737.0       reamers  DIN 8090     C      10.0        25
194       21.0  1407.0       reamers  DIN 8090     B      10.0        25
195       23.0   737.0       reamers  DIN 8090     C      10.5        25
196       20.0   674.0       reamers  DIN 8090     A      11.0        28
...        ...     ...           ...       ...   ...       ...         .
54686     39.0  1602.0   countersink        WN     0      80.0        46
54687     39.0  1602.0   countersink        WN     0       9.0        25
54688     38.0  1601.0   countersink        WN     0       9.4        25
54689     39.0  1602.0   countersink        WN     0       9.4        25
54690     39.0  1602.0   countersink        WN     0       9.5        25

       TEETH  CUTTING MATERIAL  CUTTING  CODE       VARIETY  SURFACE  \
192      6.0                HM      400.0  K10/ DK 120F   blank
193      6.0                HM      400.0  K10/ DK 120F   blank
194      6.0                HM      400.0  K10/ DK 120F   blank
195      6.0                HM      400.0  K10/ DK 120F   blank
196      6.0                HM      400.0  K10/ DK 120F   blank
```

Figure 16. data frame drop NA

Then, the columns that are irrelevant to perform data mining are removed. It is important to understand what each column represent in the dataset before removing. The following are columns that are being removed, followed by the columns that remains.



```
df1 = df.drop(columns=[  'SORTID',
                         'ITEMS',
                         'STANDARD',
                         'VARIETY',
                         'SURFACE',
                         'SHAFT',
                         'R_L',
                         'ROLL',
                         'SHAPE',
                         'CUTTING MATERIAL',
                         'ANGLE',
                         'DIAMETER_AREA',
                         'ART_FAM'
                       ])
```

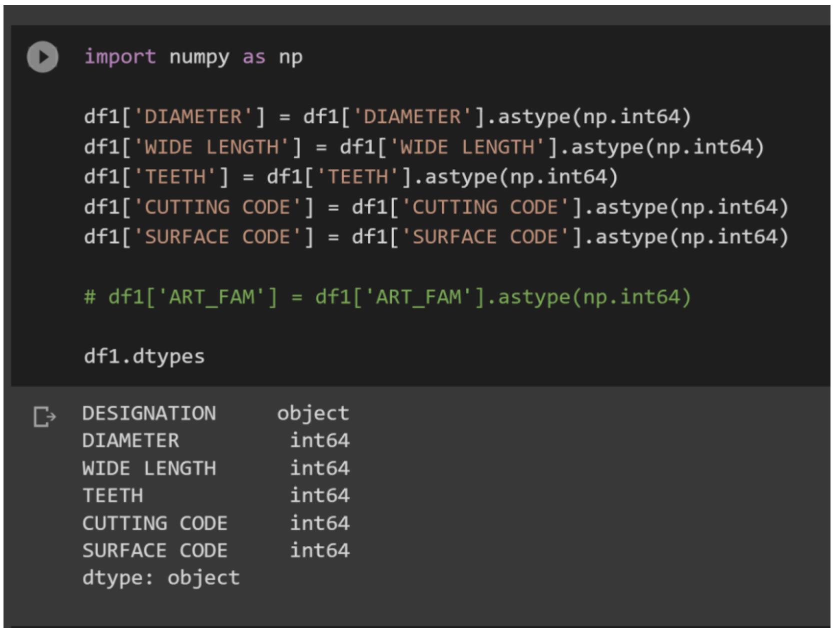
```
[ ]  df1.dtypes
```

```
DESIGNATION       object
DIAMETER         float64
WIDE LENGTH      float64
TEETH            float64
CUTTING CODE     float64
SURFACE CODE     float64
dtype: object
```

Figure 17. data frame remove columns

These remove columns are some irrelevant to perform classification, and some columns are having mix datatypes (int and string) together. I removed them anyway as it is simpler to manage the dataset as the columns that remains already suffice.

2. **Format data:**
   Whenever required, the columns are transformed text values containing integers to numeric values in order to conduct mathematical operations. For this case since we will be using classifier model using Decision Tree Classifier, it requires the values to be in integer data type. For that, several columns are transferred from float to integer values



```
import numpy as np

df1['DIAMETER'] = df1['DIAMETER'].astype(np.int64)
df1['WIDE LENGTH'] = df1['WIDE LENGTH'].astype(np.int64)
df1['TEETH'] = df1['TEETH'].astype(np.int64)
df1['CUTTING CODE'] = df1['CUTTING CODE'].astype(np.int64)
df1['SURFACE CODE'] = df1['SURFACE CODE'].astype(np.int64)

# df1['ART_FAM'] = df1['ART_FAM'].astype(np.int64)

df1.dtypes
```

```
DESIGNATION      object
DIAMETER          int64
WIDE LENGTH       int64
TEETH             int64
CUTTING CODE      int64
SURFACE CODE      int64
dtype: object
```

Figure 18. Convert data type

3.7.4 Modelling

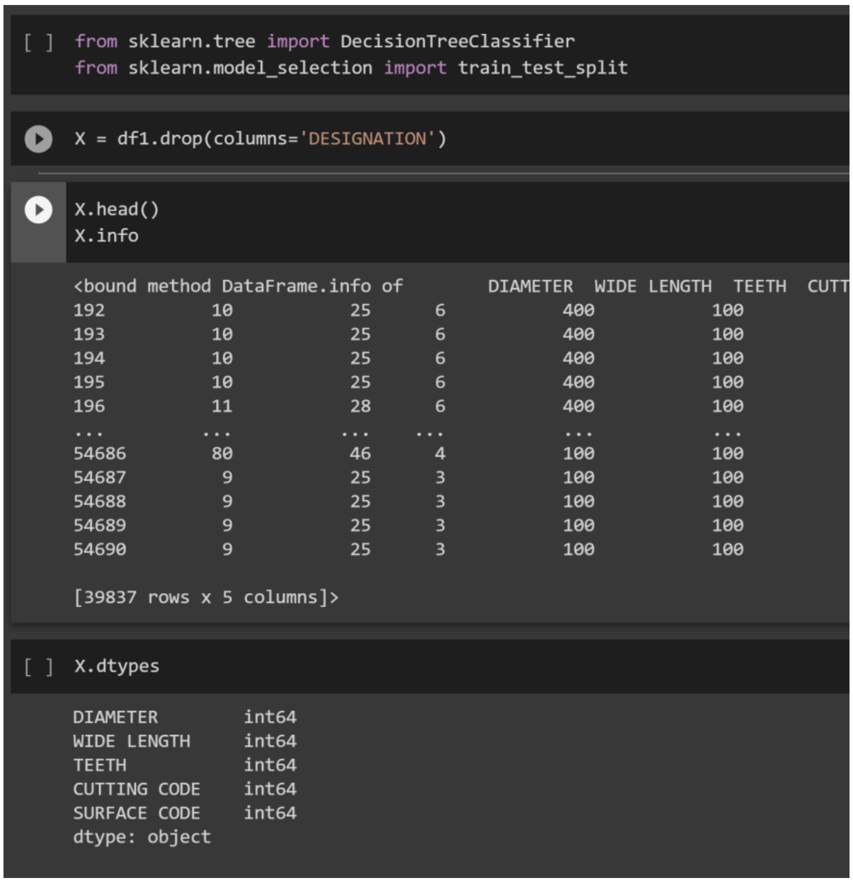
Using a variety of modelling approaches available, here is where we have to evaluate multiple models to be selected. There are several tasks in this phase:

1. **Modeling approaches to use:**
   Our dataset is labelled dataset, the machine learning technique is going to be supervised learning. For classification and regression, Decision Trees (DTs) are a non-parametric supervised learning approach. The objective is to learn basic decision rules from data attributes to develop a model that predicts the value of a target variable. A tree approximates a piecewise constant.
   The following are some of the benefits of decision trees:
   - It just takes a few minutes to prepare the data. Other procedures often need data normalization, the creation of dummy variables, and the removal of blank values. This module, however, does not handle missing values.
   - The cost of utilizing the tree (that is, predicting data) is proportional to the amount of data points needed to train it.
   - Capable of working with both numerical and category data.
   - The model is based on a white box. If a circumstance can be seen in a model, Boolean logic can simply describe the situation. In contrast, the findings of a black box model (such as an artificial neural network) may be more difficult to decipher.

DTs are chosen since our dataset size is ideal for the algorithm to obtain high accuracy.

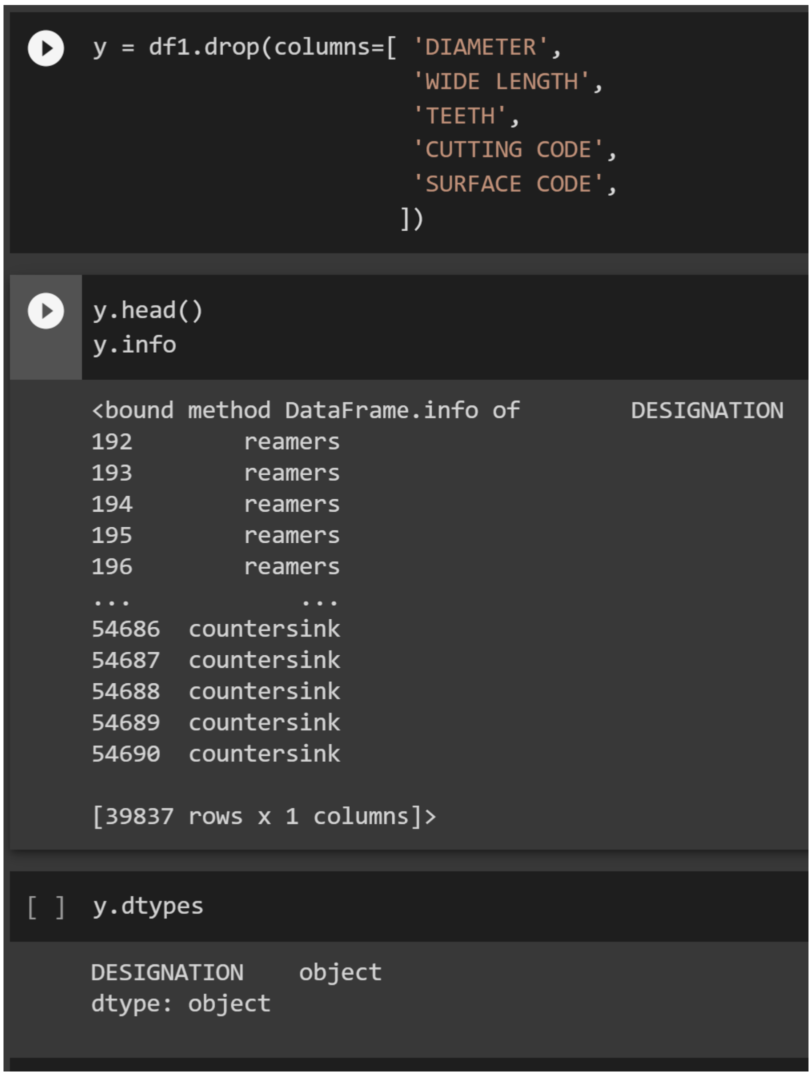
2. **Separate dataset for test:**

   In this stage dataset are separated for the purpose of testing the model and train the model. The most common ratio for test data and training data is 3:10 respectively. Before splitting the dataset is split into two data frames X and Y. X being the data frame consisting of all influencing variable, whereas Y being data frame that consist only the target variable. The code are as follows:



```
[ ] from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import train_test_split

    X = df1.drop(columns='DESIGNATION')

    X.head()
    X.info

    <bound method DataFrame.info of        DIAMETER  WIDE LENGTH  TEETH  CUTT
    192          10           25      6          400           100
    193          10           25      6          400           100
    194          10           25      6          400           100
    195          10           25      6          400           100
    196          11           28      6          400           100
    ...         ...          ...    ...          ...           ...
    54686        80           46      4          100           100
    54687         9           25      3          100           100
    54688         9           25      3          100           100
    54689         9           25      3          100           100
    54690         9           25      3          100           100

    [39837 rows x 5 columns]>

[ ] X.dtypes

    DIAMETER        int64
    WIDE LENGTH     int64
    TEETH           int64
    CUTTING CODE    int64
    SURFACE CODE    int64
    dtype: object
```

Figure 19. Split dataframe for influencing variables



```
y = df1.drop(columns=[ 'DIAMETER',
                       'WIDE LENGTH',
                       'TEETH',
                       'CUTTING CODE',
                       'SURFACE CODE',
                      ])
```

```
y.head()
y.info
```

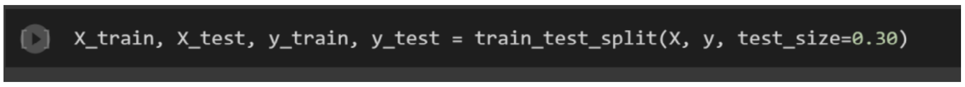
```
<bound method DataFrame.info of        DESIGNATION
192        reamers
193        reamers
194        reamers
195        reamers
196        reamers
...            ...
54686  countersink
54687  countersink
54688  countersink
54689  countersink
54690  countersink

[39837 rows x 1 columns]>
```

```
[ ]  y.dtypes
```

```
DESIGNATION    object
dtype: object
```

Figure 20. Split data frame into Target variables only

After having the two data frame, it is then the data is separated into train and test data:

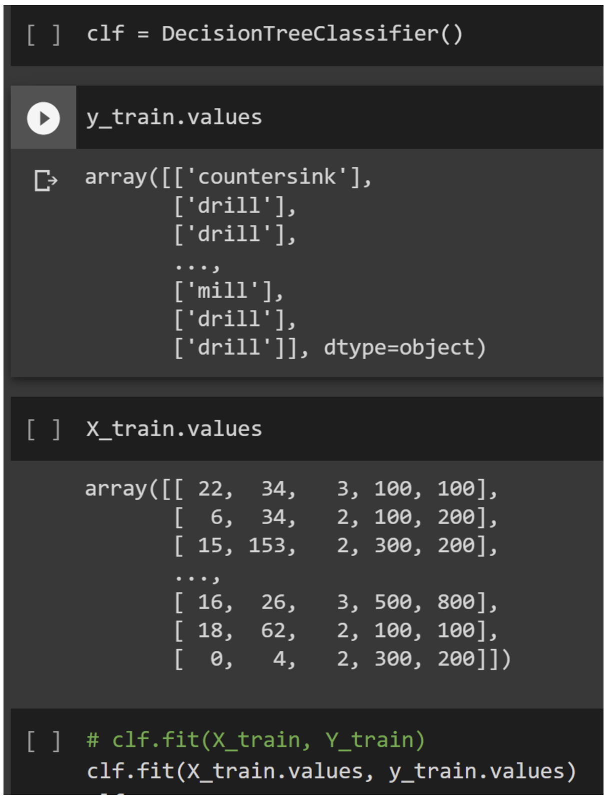


```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

Figure 21. Split train and test dataset

3. **Build model:**

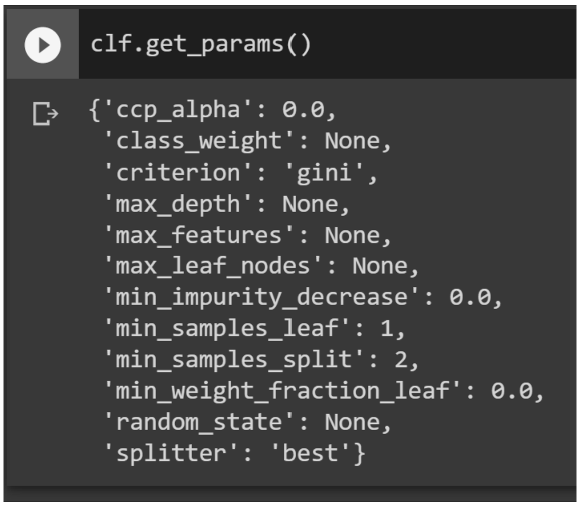
   To build a DT model, will only require few lines of codes



```
[ ] clf = DecisionTreeClassifier()

    y_train.values

    array([['countersink'],
           ['drill'],
           ['drill'],
           ...,
           ['mill'],
           ['drill'],
           ['drill']], dtype=object)

[ ] X_train.values

    array([[ 22,  34,   3, 100, 100],
           [  6,  34,   2, 100, 200],
           [ 15, 153,   2, 300, 200],
           ...,
           [ 16,  26,   3, 500, 800],
           [ 18,  62,   2, 100, 100],
           [  0,   4,   2, 300, 200]])

[ ] # clf.fit(X_train, Y_train)
    clf.fit(X_train.values, y_train.values)
```

Figure 22. Fit dataset into model

To get the classifier parameters we run this code:



```
clf.get_params()

{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'random_state': None,
 'splitter': 'best'}
```

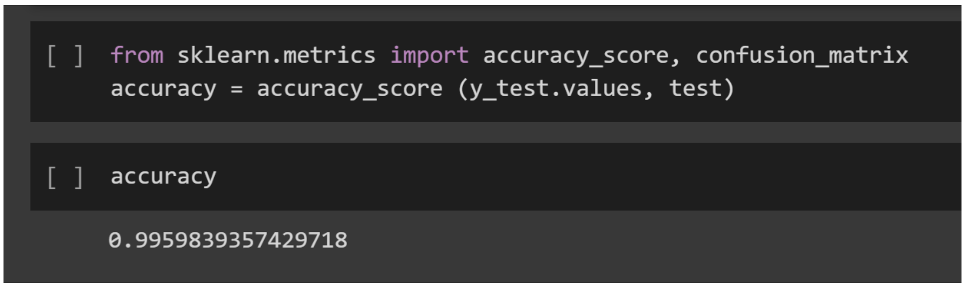
Figure 23. Get parameter of classifier

3.7.5 Evaluation

The Evaluation phase looks at accuracy score of the model built and what to do next in a broader sense.

1. **Assess the outcomes:**

   To determine whether the model which was built is accurate or not, we can perform accuracy score check

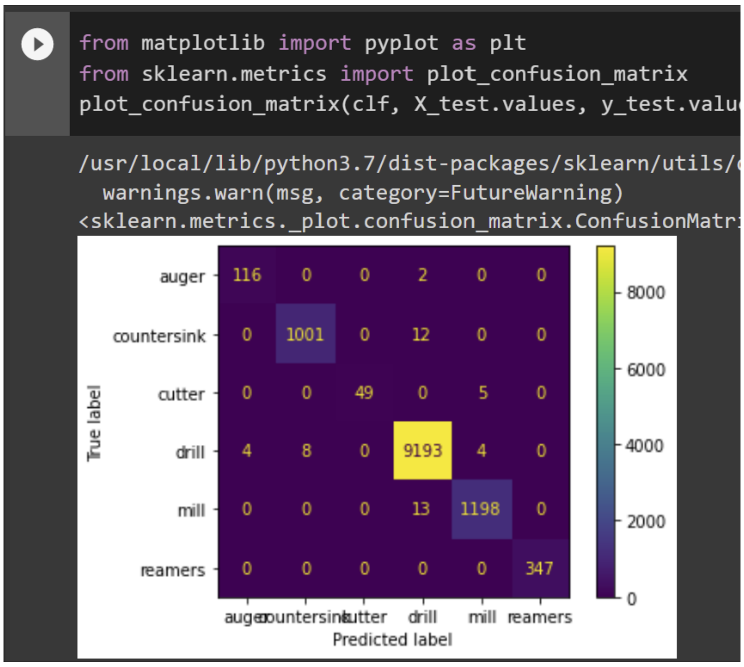


```
[ ]  from sklearn.metrics import accuracy_score, confusion_matrix
     accuracy = accuracy_score (y_test.values, test)

[ ]  accuracy

     0.9959839357429718
```

Figure 24. Calculate model accuracy score

2. **Visualize the prediction**

   We can also visualize the accuracy score for the model built by using confusion matrix.

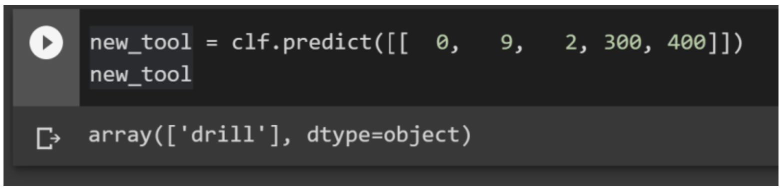


```
from matplotlib import pyplot as plt
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(clf, X_test.values, y_test.valu

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatr
```

Figure 25. Visualize the accuracy score

3. **Test the prediction**

   Randomly select and a row from the dataset and entered the values in an array to see whether the predicted value for the target variable is the same as the dataset or not



```
new_tool = clf.predict([[  0,   9,   2, 300, 400]])
new_tool
```

```
array(['drill'], dtype=object)
```

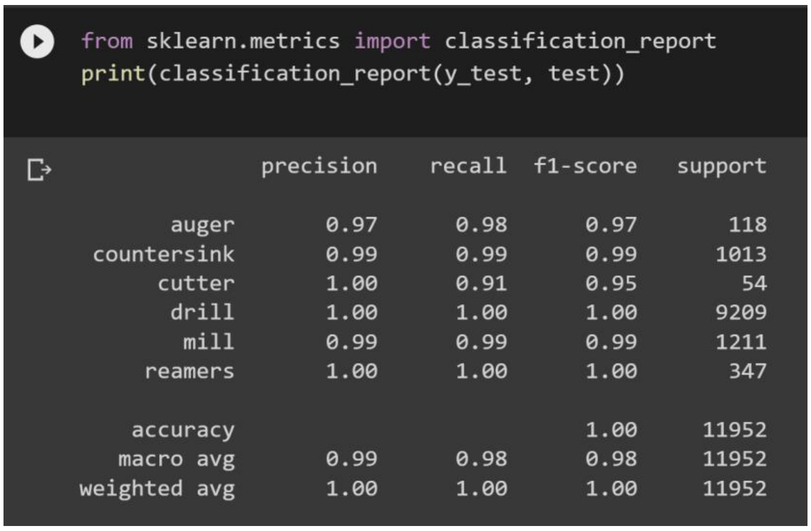
Figure 26. Retest with random values

3.7.6 Deployment

A model isn't very helpful unless the findings can be accessed by the person. This phase's complexity varies greatly. There are few tasks for the last phase:

1. **Generate a report:**

   The is command line to generate a report for that data to present data mining findings.



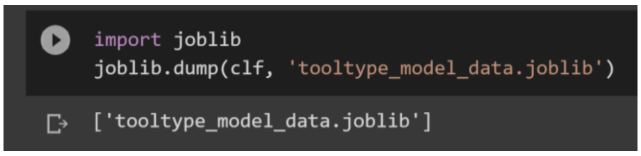
```
from sklearn.metrics import classification_report
print(classification_report(y_test, test))
```

```
              precision    recall  f1-score   support

       auger       0.97      0.98      0.97       118
 countersink       0.99      0.99      0.99      1013
      cutter       1.00      0.91      0.95        54
       drill       1.00      1.00      1.00      9209
        mill       0.99      0.99      0.99      1211
     reamers       1.00      1.00      1.00       347

    accuracy                           1.00     11952
   macro avg       0.99      0.98      0.98     11952
weighted avg       1.00      1.00      1.00     11952
```

Figure 27. Produce classification report

2. **Export to usable file**

   Model trained can be exported into .joblib file to be use in the web application for predictive classification of Tools into Tool type. The following are the code



```
import joblib
joblib.dump(clf, 'tooltype_model_data.joblib')
```

```
['tooltype_model_data.joblib']
```

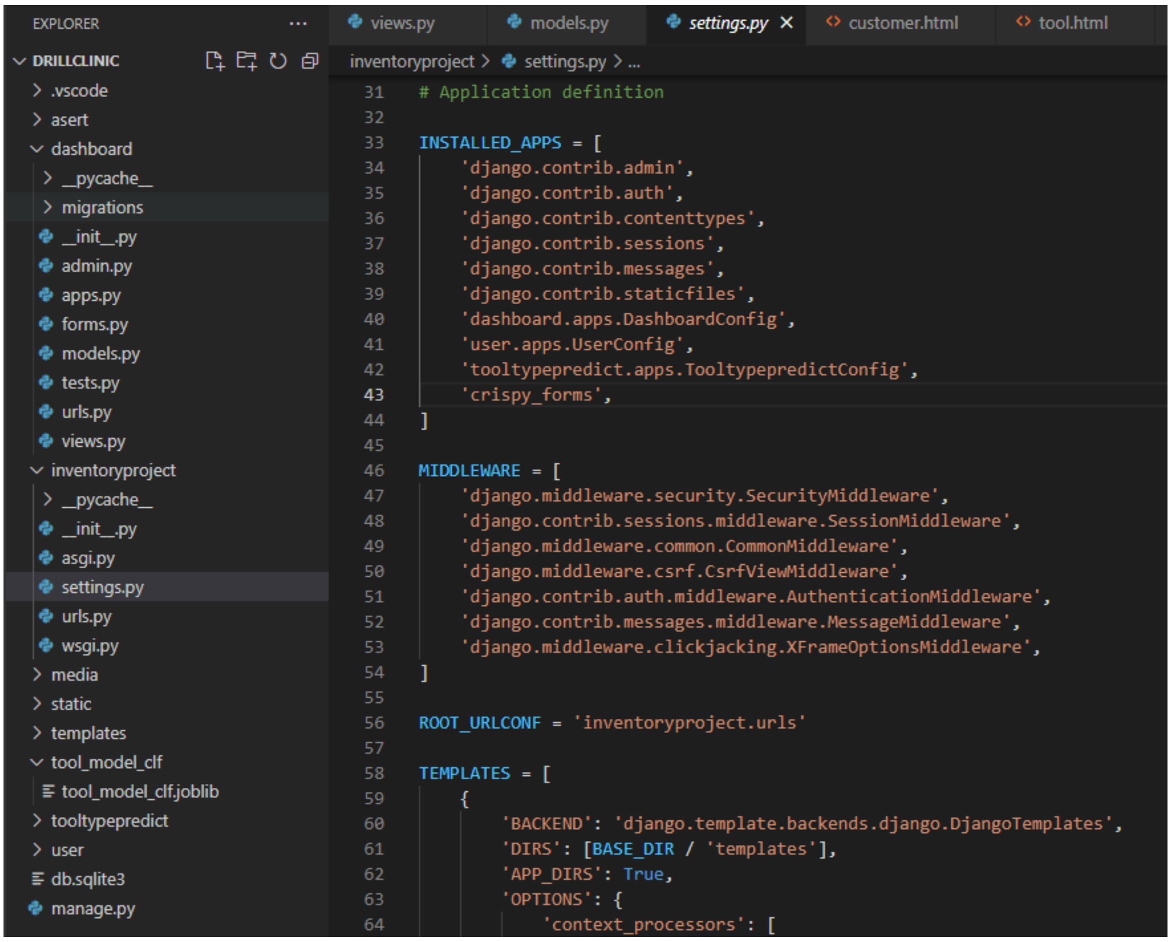
Figure 28. Export model to joblib

3.8 Web Development

As mentioned on the earlier chapters, the project consists of sub-app arranged by folders. The root project “inventoryproject” consist of several model namely:

- “dashboard”
- “user”

It is arranged according to Django documentation, to effectively keep track and understand the separation of concern while developing. It helps to know which file will take effect when implementing a functionality. This part of the coding shows the application (as models) that are installed into the projecrt

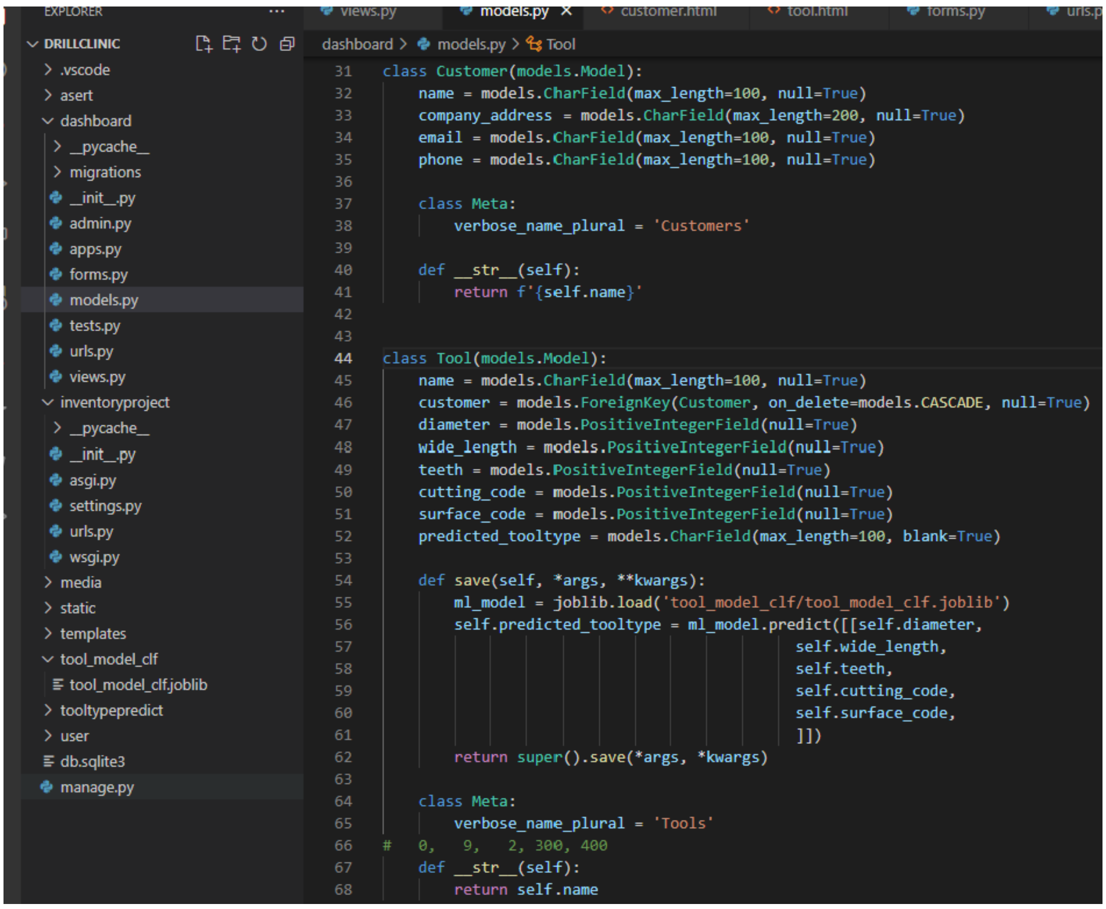


```python
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'dashboard.apps.DashboardConfig',
    'user.apps.UserConfig',
    'tooltypepredict.apps.TooltypepredictConfig',
    'crispy_forms',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'inventoryproject.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
```

Figure 29. Setting.py

3.8.1 Dashboard model

The dashboard model will store related Classes that will be displayed by Template file (index.html). This snippet shows two classes together with it's fields defined in models.py which are Customer and Tool. There are several more classes defined in here.



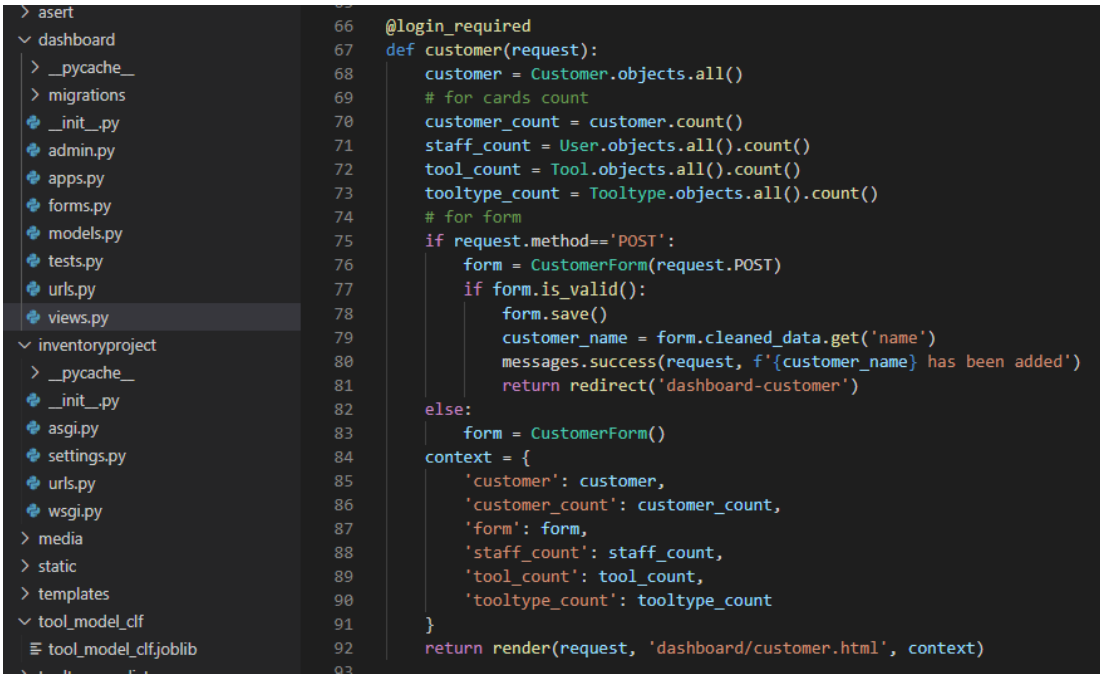
```
31 class Customer(models.Model):
32     name = models.CharField(max_length=100, null=True)
33     company_address = models.CharField(max_length=200, null=True)
34     email = models.CharField(max_length=100, null=True)
35     phone = models.CharField(max_length=100, null=True)
36 
37     class Meta:
38         verbose_name_plural = 'Customers'
39 
40     def __str__(self):
41         return f'{self.name}'
42 
43 
44 class Tool(models.Model):
45     name = models.CharField(max_length=100, null=True)
46     customer = models.ForeignKey(Customer, on_delete=models.CASCADE, null=True)
47     diameter = models.PositiveIntegerField(null=True)
48     wide_length = models.PositiveIntegerField(null=True)
49     teeth = models.PositiveIntegerField(null=True)
50     cutting_code = models.PositiveIntegerField(null=True)
51     surface_code = models.PositiveIntegerField(null=True)
52     predicted_tooltype = models.CharField(max_length=100, blank=True)
53 
54     def save(self, *args, **kwargs):
55         ml_model = joblib.load('tool_model_clf/tool_model_clf.joblib')
56         self.predicted_tooltype = ml_model.predict([[self.diameter,
57                                                      self.wide_length,
58                                                      self.teeth,
59                                                      self.cutting_code,
60                                                      self.surface_code,
61                                                      ]])
62         return super().save(*args, **kwargs)
63 
64     class Meta:
65         verbose_name_plural = 'Tools'
66 #    0,    9,    2, 300, 400
67     def __str__(self):
68         return self.name
```

Figure 30. Models.py

Also in this model, Data Matrix code will be generated after executing the creation of Tool object. It will contain URL for the respective tool ID. It's unique identifier for the Tool.

There is also predictive algorithm on play in this file, at line 55, the field is being used by importing 'tool_model_clf.joblib' object . This enables whatever Tool parameters user insert into the system, it will tirgger a calculation to predict the category the Tool belongs

When dealing with object creation or any CRUD (create, read/retrieve, update, delete) functions, there is necessary to have Form application for the user to key-in the object and saved in the database. The form is called in the views.py in the "Dashboard" model, later it will be rendered to the respective URLs.

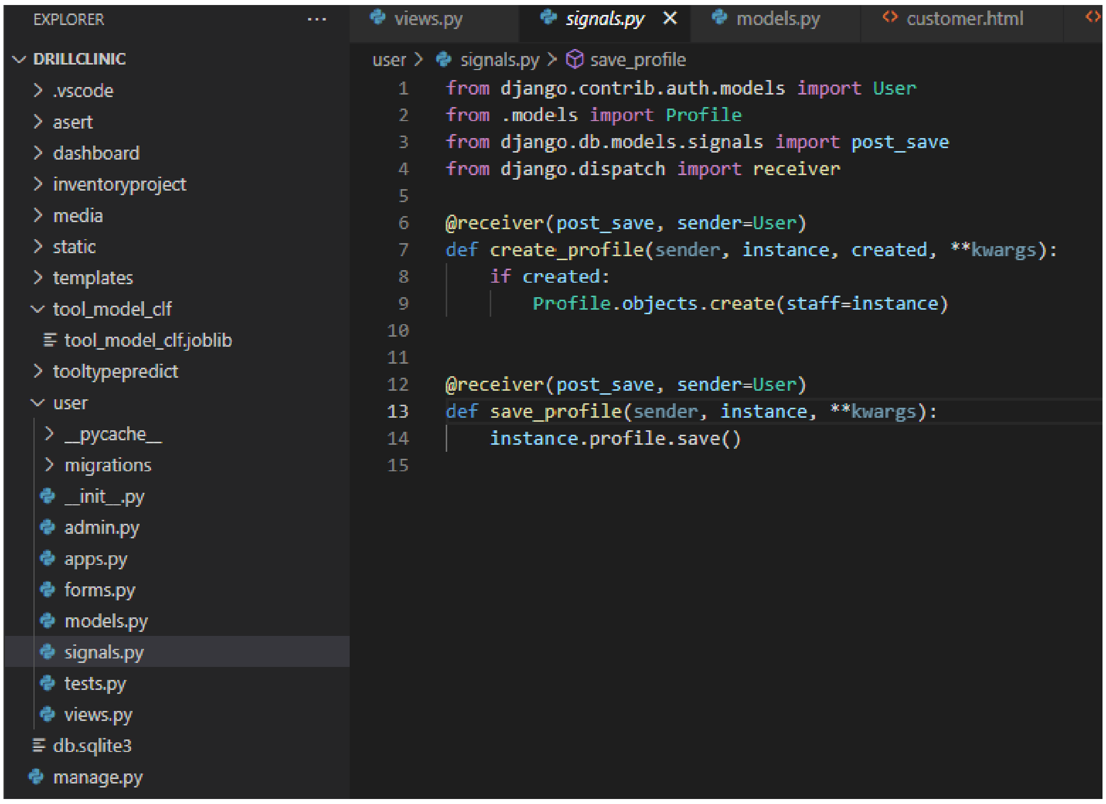


```python
@login_required
def customer(request):
    customer = Customer.objects.all()
    # for cards count
    customer_count = customer.count()
    staff_count = User.objects.all().count()
    tool_count = Tool.objects.all().count()
    tooltype_count = Tooltype.objects.all().count()
    # for form
    if request.method=='POST':
        form = CustomerForm(request.POST)
        if form.is_valid():
            form.save()
            customer_name = form.cleaned_data.get('name')
            messages.success(request, f'{customer_name} has been added')
            return redirect('dashboard-customer')
    else:
        form = CustomerForm()
    context = {
        'customer': customer,
        'customer_count': customer_count,
        'form': form,
        'staff_count': staff_count,
        'tool_count': tool_count,
        'tooltype_count': tooltype_count
    }
    return render(request, 'dashboard/customer.html', context)
```

31 view.py

3.8.2 User model

For this model, there will be Django signals feature to be utilized. It is there is one-to-one relationship between different classes. In this case, whenever the user is registered into the database, concurrently that User has its own Profile created. The following code shows the signals.py to create profile after user is created.



```python
from django.contrib.auth.models import User
from .models import Profile
from django.db.models.signals import post_save
from django.dispatch import receiver

@receiver(post_save, sender=User)
def create_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(staff=instance)


@receiver(post_save, sender=User)
def save_profile(sender, instance, **kwargs):
    instance.profile.save()

```

32. signals.py

3.9 Framework and Environment used

3.9.1 Django

Django is a high-level Python web framework that promotes quick development and simple, practical design. It's built by professional developers to take care of a lot of the headaches of web development so you can concentrate on developing your app instead of reinventing the wheel. It's open source and free. It is mostly used for developing the backend of this application. Several benefits of using Django:

- Extremely quick.
  Django was created to make it as easy as possible for developers to get from idea to completion.
- It's reassuringly safe.
  Django is concerned about security and assists developers in avoiding numerous frequent security blunders.
- Extremely adaptable.
  Django's ability to grow swiftly and flexibly is used by some of the busiest websites on the internet.

3.9.2 Bootstrap

Bootstrap is a massive library of HTML, CSS, and JavaScript code that may be reused. It's also a frontend development framework that allows developers and designers to create completely responsive websites in a short amount of time.

Essentially, Bootstrap saves you time by reducing the amount of CSS code you have to write, allowing you to spend more time building websites. It is mostly used for developing the frontend of this application

3.9.3 Google Colab

Google Research's Colaboratory, or "Colab" for short, is a product. Colab is a web-based Python editor that enables anybody to create and run arbitrary Python code. It's notably useful for machine learning and data analysis for this project. Colab, in more technical terms, is a hosted Jupyter notebook service that needs no installation and provides free access to computer resources, including GPUs.

3.9.4 Microsoft's Visual Studio Code

Microsoft's Visual Studio Code (often known as VS Code) is a free open-source text editor. For Windows, Linux, and macOS, VS Code is available. VS Code has numerous significant features that have made it one of the most popular development environment tools in recent years, despite its modest weight. For this project, it is the main environment tools to develop the application.

3.9.5 Selenium

Selenium is an open-source web browser automation tool. It offers a single interface for writing test scripts in a variety of programming languages, including Ruby, Java, NodeJS, PHP, Perl, Python, and C#.

The scripts are then executed by a browser-driver on a browser-instance on your device. Selenium is the main tool to perform integration test of the core functionalities of this application.

3.10 System Integration Test

To create test cases for the system, it is best to relate with use case diagram to arrange the test cases according to user-perspective. All the test cases are grouped as classes for example (Tools, User, Profile, Customer). Due to the orientation of the test table to best be in landscape mode, the test case table is attached at appendices section together with the result of the test.

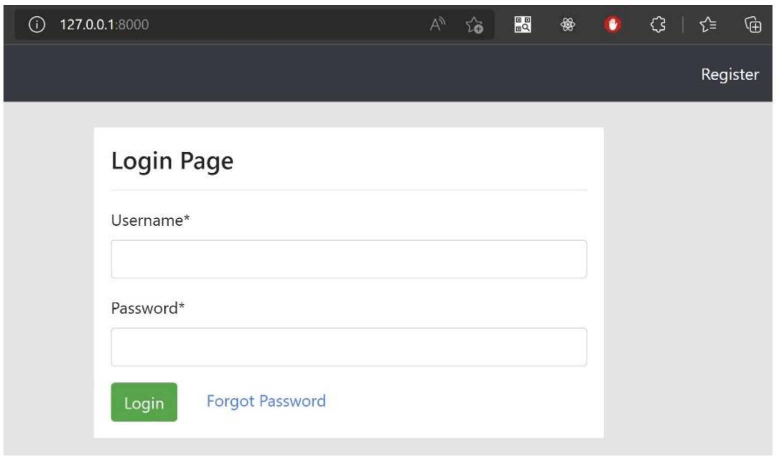
4. Results and Discussion

We'll go through all of the Drillclinic Inventory Management Application's ultimate outputs in this section of the talk. Aside from that, we'll talk about and investigate if the system has met the goals that were previously agreed and defined. As a result, we'll go further into those parts.

4.1 User interface

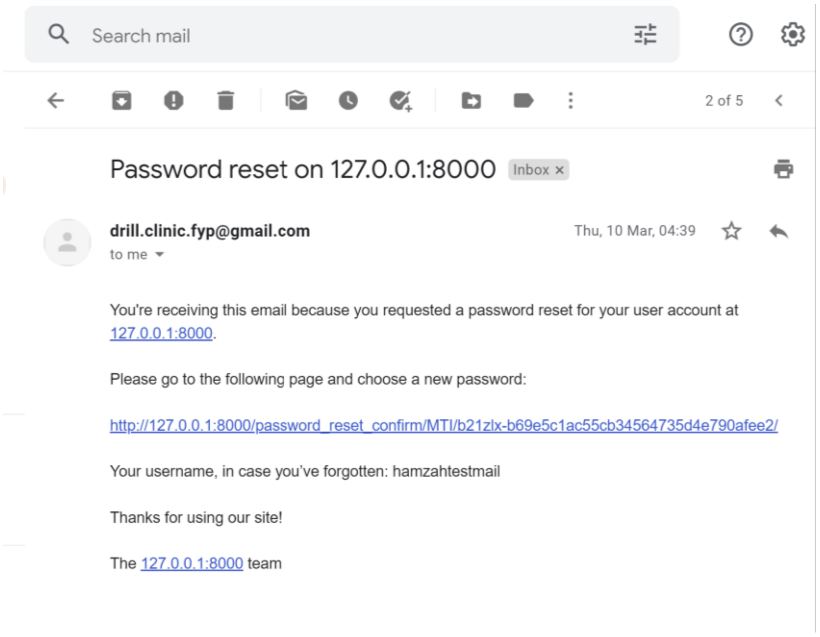
4.1.1 Drillclinic Login page

This page enable user to enter credentials or register for the first time.



33. Login page

There is also “forgot password” feature where user will receive confirmation link sent to their email as follows (if using gmail)



Password reset on 127.0.0.1:8000

drill.clinic.fyp@gmail.com
to me
Thu, 10 Mar, 04:39

You're receiving this email because you requested a password reset for your user account at 127.0.0.1:8000.

Please go to the following page and choose a new password:

http://127.0.0.1:8000/password_reset_confirm/MTI/b21zlx-b69e5c1ac55cb34564735d4e790afee2/

Your username, in case you've forgotten: hamzahtestmail

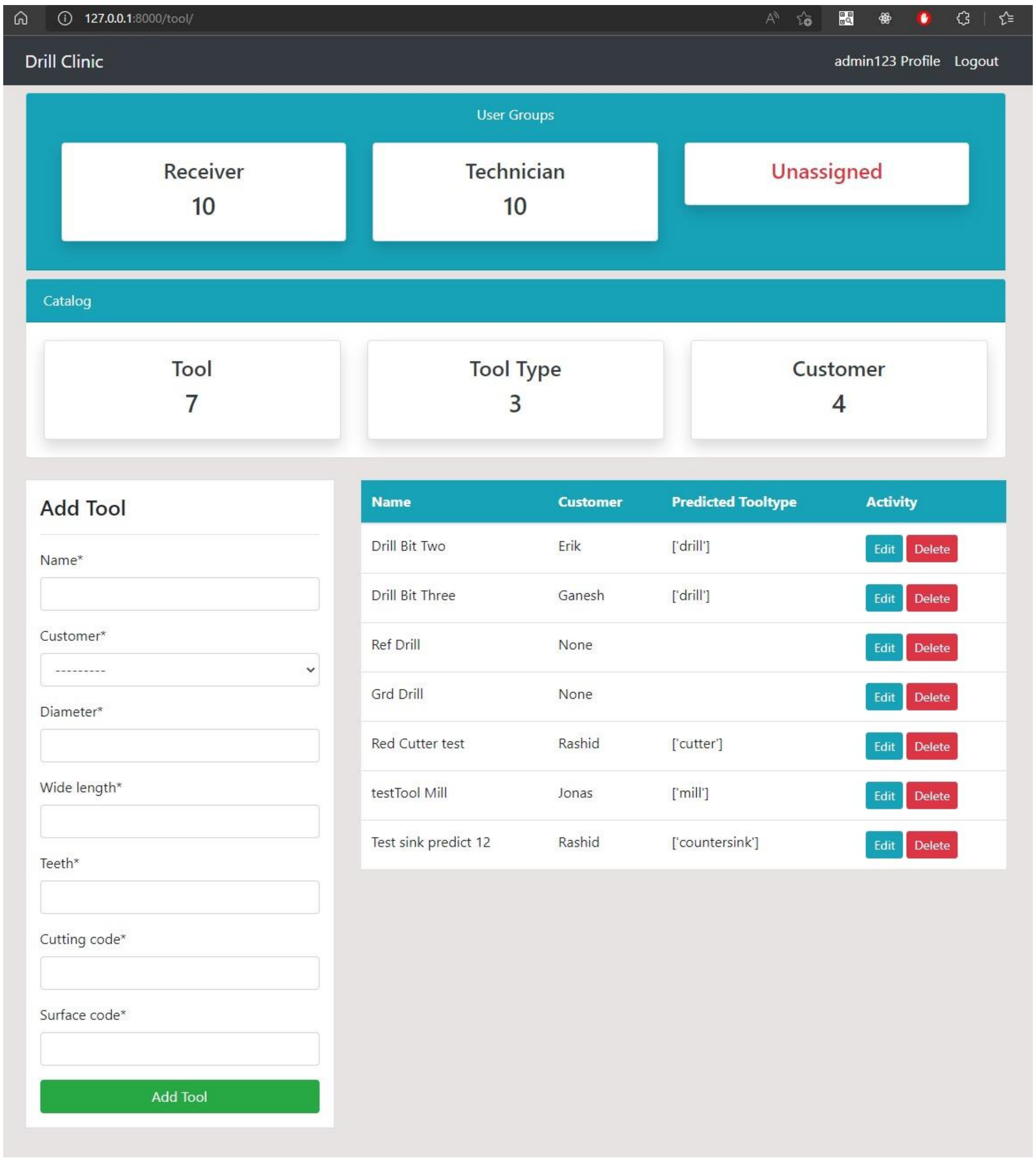
Thanks for using our site!

The 127.0.0.1:8000 team

34. Forgot password email link

4.1.2 Drillclinic Dashboard

In this page, Receiver users, have the permission to create a Tool Object as well viewing Tool list. They can also navigate to Customer page by clicking on the ‘Customer’ card, where they can create and view all Customers in the database. The user also can navigate to his/her profile page at top right corner and able to edit the user information. In this page also, the user can see the predictive classification for Tooltype once they entered the parameters in the form on the left screen corner.



127.0.0.1:8000/tool/

Drill Clinic — admin123 Profile — Logout

User Groups

Receiver	Technician	Unassigned
10	10	

Catalog

Tool	Tool Type	Customer
7	3	4

Add Tool

Name*

Customer* ---------

Diameter*

Wide length*

Teeth*

Cutting code*

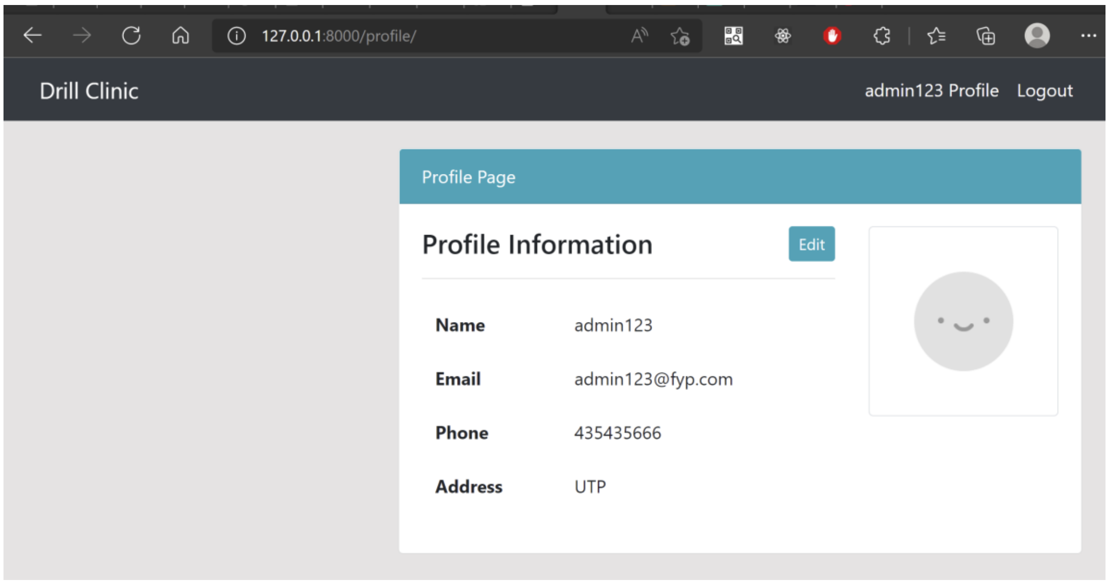
Surface code*

Add Tool

Name	Customer	Predicted Tooltype	Activity
Drill Bit Two	Erik	['drill']	Edit Delete
Drill Bit Three	Ganesh	['drill']	Edit Delete
Ref Drill	None		Edit Delete
Grd Drill	None		Edit Delete
Red Cutter test	Rashid	['cutter']	Edit Delete
testTool Mill	Jonas	['mill']	Edit Delete
Test sink predict 12	Rashid	['countersink']	Edit Delete

35. Drillclinic dashboard

4.1.3 Drillclinic Profile page


127.0.0.1:8000/profile/

Drill Clinic — admin123 Profile Logout

Profile Page

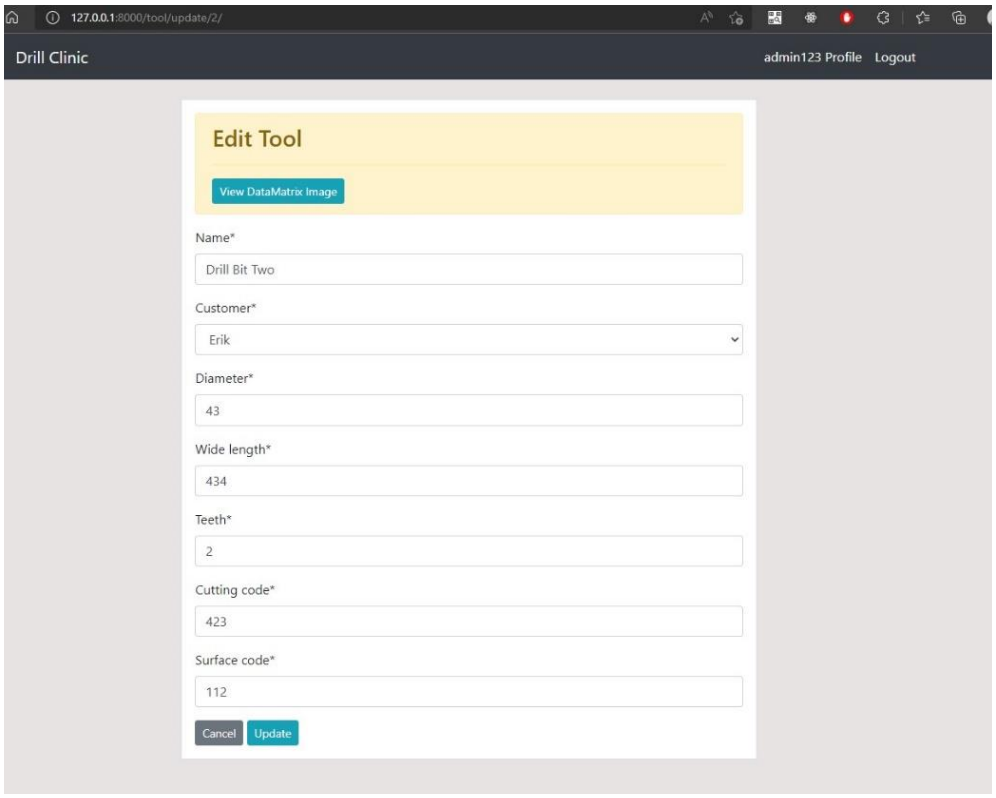
Profile Information Edit

Name	admin123
Email	admin123@fyp.com
Phone	435435666
Address	UTP

36. Profile page

4.1.4 Drillclinic Object Detail page

The detail page of object (Tool/Customer/Profile) enable user to edit/update the values of the object parameter. In the case of Tool detail page, there is extend for viewing Data Matrix code image for this specific tool


127.0.0.1:8000/tool/update/2/

Drill Clinic — admin123 Profile Logout

Edit Tool

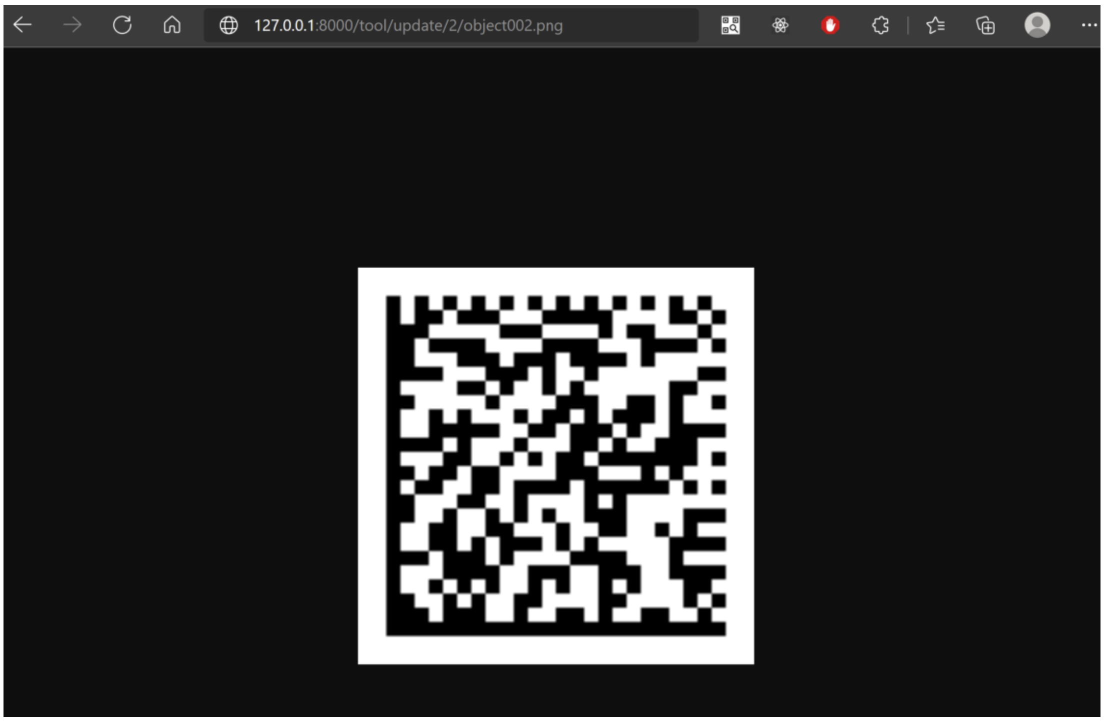
View DataMatrix Image

Name*	Drill Bit Two
Customer*	Erik
Diameter*	43
Wide length*	434
Teeth*	2
Cutting code*	423
Surface code*	112

Cancel Update

37. Tool Detail page

The image below are an extend of Tool detail page after user click 'View Datamatrix image' button from previous page. You can try to scan the code, it should show the exact URL of the tool.



38. Data matrix code image

4.2 Integration Testing Result

All core functionalities are tested multiple times to debug any error, the result are tabulated together with test case table. Attached in appendices

5. Conclusion and Recommendation

Onto that note, all the core functionalities have satisfied the project objectives and the functional requirement have been tested and verified. We hope this application would help to increase the efficiency of the staff to record Tools and reduce the time taken for daily business operation. The insight that we get from data mining process would help the company to automate their process even further without having high dependencies on manual labour.

In future perspective, there can be additional feature to be added in the application for more informative system. The feature that can be implemented are Tool Lifetime Prediction to measure the durability of each tool after going thru several maintenances. This can be done, only when there are adequate datasets provided with information of the number of occurrences the Tool is sent for maintenance within a period of time. The higher the occurrence, the lower the durability of the Tool.

Besides, in terms of business perspective, this can help business to prepare the tool for maintenance even before they are heavily worn out. This would help business not to have downtime and always prepare a standby Tool. In conclusion, there can be more feature to increase the use of IOT in the industry and improve the system.

References

Himanshu Gore, R. K. (2021). Django: Web Development Simple &amp; Fast. Annals of the Romanian Society for Cell Biology. *Annals of the Romanian Society for Cell Biology*, 4576–4585. Retrieved from https://www.annalsofrscb.ro/index.php/journal/article/view/6301

Karrach, L. &. (2018). Options to Use Data Matrix Codes in Production Engineering. *Management Systems in Production Engineering, 4*(26), 231-236. doi:https://doi.org/10.1515/mspe-2018-0038

Karrach, L. a. (2018). *The analyse of the various methods for location of Data Matrix codes in images.* ELEKTRO. doi:10.1109/ELEKTRO.2018.8398250.

Karrach, L., & Pivarčiová, E. (2021). Comparative Study of Data Matrix Codes Localization and Recognition Methods. *Journal of Imaging*(7(9)), 163. doi:https://doi.org/10.3390/jimaging7090163

Ravindran, A. (2018). *Django Design Patterns and Best Practices: Industry-standard web development techniques and solutions using Python, 2nd Edition.* Packt Publishing Ltd.

Unmesh Gundecha, S. A. (2018). *Selenium WebDriver 3 Practical Guide: End-to-end automation testing for web and mobile browsers with Selenium WebDriver, 2nd Edition* (2 ed.). Packt Publishing Ltd.

Vincent, W. S. (2021). *Django for Beginners: Build websites with Python and Django.* WelcomeToCode.

Appendices

Data Matrix code

Test Case ID	**Test Case Name**	**Test Case Category**	User Group	**Test Data / Inputs example**	**Preconditions**	**Test Steps**	**Expected Outcome**	**Selenium Result**
dtmx_uc1_01	Generate datamatrix	Critical	• Admin • Receiver	• drill bit two	Login using admin/receiver credentials	1. Page: Tool list page 2. Create tool name: drill bit two 3. Click button 'View datamatrix image'	A Datamatrix image encoded with 'drill bit two' URL is seen	PASS
dtmx_uc1_02	Download data matrix	Critical	• Admin • Receiver	• drill bit two	Tool object 'drill bit two' is created	1. Page: Tool list page 2. Click to 'drill bit two' 3. Click button 'View datamatrix image'	A Datamatrix image encoded with 'drill bit two' URL is seen and can be downloaded using browser context	PASS