1. **INTRODUCTION**
   1. **OVERVIEW OF THE PROJECT**

A sort of recommendation system called collaborative filtering suggests products to other users based on the interests and actions of numerous users. By using collaborative filtering, the algorithm first collects information on how people have interacted with things, such as their ratings or past purchases. The system then employs this information to determine user similarities based on their preferences or behavior. Lastly, based on the preferences of other users, the system offers recommendations for products that a user has not yet engaged with. In the rubber business, where several production processes take place and inputs with similar characteristics share output with other inputs, we have put this collaborative filtering notion into practice. Upload the details of the raw materials in the sub-module name ‘raw materials' and view the raw materials of natural and rubber. We find the best output and we suggest the rubber output to the input of the similar addictives in the making of rubber, so that the recommendation system works better with this concept of collaborative filtering, in this module analyze the report and then upload the raw materials for a testing report using classification algorithm.

## 1.2 OBJECTIVE OF THE SYSTEM

* It is applicable to nonlinear complex problems.
* Works with large amounts of input data.
* After learning, it offers immediate prediction.
* Even with similar data, the very same accuracy ratio can be achieved.
* Collaborative filtering is the effective way to recommend ones output to another’s input.

**1.3 EXISTING SYSTEM**

Our current system can assist scholars in becoming aware of new fields and fully actually understanding the development of their fields; on the contrary side, it can assist scholars in staying current with state-of-the-art works in their disciplines. More importantly, it facilitates the discovery of relevant articles by scholars. Based on collaborative filtering, hybrid recommendation methods have been proposed (CF). Cluster and NCN can take advantage of the entire citation network topology by leveraging the power of NRL. The NRL method is used to pick adjacent researchers in the cluster and NCN procedures. Due to the scarcity of the citation network, the comparison between scholars may be less than zero, resulting in incorrect recommendations. The more accurate its knowledge extraction, the slower and more efficient its decisions will be. We present a novel method for developing a paper rating matrix based on attributed citation network embedding that takes into account both network topology and text information. To solve the data sparsity problem in neighbor scholar selection, a novel method based on attributed collaboration network representation learning is proposed. Cluster is a novel citation recommendation algorithm that combines the benefits of collaborative filtering with network representation learning for citation recommendation. Scholars are publishing at an unprecedented rate in the age of scholarly big data, making it difficult to find related papers.

**1.4 PROPOSED SYSTEM**

In our business model (Rubber Industry), we have used the concept of collaborative filtering which is used to find the similarities between set of data and does the recommendation with the help of our strong classifier algorithm called multi-layer perception (MLP). After requesting, a process unit will predict the rubber based recommendation of the raw materials. The Prediction for the raw materials is analyzed based on the assigned positions in the algorithm. Following the prediction, the process for recommendation takes place. Once the recommendation process for one request is completed then the algorithm shows to which raw materials order the next production for materials goes. MLP is a type of supervised learning technique for categorizing data. The algorithm works by training multiple neural networks. As inputs, the estimation is generated using the amount of input required per hidden layer and its connection parameters. Collaborative Filtering (CF) is a statistical method for identifying and recommending products and services that may be relevant to a specific user. Customer feedback is scored and stored in a database by the algorithm. As long as the database is large enough to reflect the opinions of a specific user base, CF allows for continuous improvement.

**1.4.1 Advantage of Proposed System**

* It is applicable to nonlinear complex problems.
* Works with large amounts of input data.
* After learning, it offers immediate prediction.
* Even with similar data, the very same accuracy ratio can be achieved.
* Collaborative filtering is the effective way to recommend ones output to another’s input.

**2.SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 SOFTWARE REQUIREMENTS**

* **Front end :** Core Python, CSS, JS
* **Web application :** Django
* **Back end :** MySQL
* **Algorthim :** MLP
  1. **FUNCTIONAL REQUIREMENTS**
* Following is a list of functionalities of the browsing enabled system.
* An Activity with a UI that allows you to browser settings. Provide a second Activity that allows users to access the share with permission from the administrator. Handle activity lifecycle appropriately. A precondition for any points in this part of the grade is code that compiles and runs.
* Your application should allow a user to browse the shares, buy and sell the shares with specific metadata. The assignment requires you to create a UI for browsing and a UI for integrating the two.
* The Net beans provide a number of useful layout components, views, and tools that you may want to use to create your location browser. As with the final project, you should design your application to only use the buttons on the Key board and mouse as input. Your application should use the Key board, Mouse and keywords.

**2.3 NON-FUNCTIONAL REQUIREMENTS**

* The system should be supported Net beans. The member should use the System browser. Each member should have a separate system.
* The system should ask the username and password to open the application. It doesn’t permit to unregistered user to access the System.
* The system should have Role based System functions access. Approval Process has to be defined.
* The system should have Modular customization components so that they can be reused across the implementation.
* These are the mainly following:
* Secure access of confidential data. 24 X 7 availability
* Better component design to get better performance at peak time
* Flexible service based architecture will be highly desirable for future extension

**2.4 HARDWARE AND SOFTWARE REQUIREMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Developing Kit | | | |
|  | Processor | RAM | Disk Space |
| Pycharm  Anaconda | Computer with a 2.6GHz processor or higher  Computer with a 2.6GHz processor or higher | 4GB  4GB | Minimum 20 GB  Minimum 20 GB |
| Database | | | |
| MySQL | Intel Pentium processor at 2.6GHz or faster | Minimum 512 MB Physical Memory; 1 GB Recommended | Minimum 20 GB |
| WampServer | Intel Pentium processor at 2.6GHz or faster | Minimum 512 MB Physical Memory | Minimum 20 GB |

Table 2.4 Hardware and Software Requirements

**2.5 LANGUAGE SPECIFICATION**

**2.5.1 INTRODUCTION TO PYTHON**

* Python was conceived in the late 1980s by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatics](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica)  in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to [ABC programming language](https://en.wikipedia.org/wiki/ABC_(programming_language)), which was inspired by [SETL](https://en.wikipedia.org/wiki/SETL), capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)) operating system. Its implementation began in December 1989; Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's [Benevolent Dictator for Life](https://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life), a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. He now shares his leadership as a member of a five-person steering council.  In January 2019, active Python core developers elected Brett Cannon, Nick Coughlan, Barry Warsaw, Carol Willing and Van Rossum to a five-member "Steering Council" to lead the project.  Guido van Rossum has since then withdrawn his nomination for the 2020 Steering council.
* Python 2.0 was released on 16 October 2000 with many major new features, including a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode).
* Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility). Many of its major features were [back ported](https://en.wikipedia.org/wiki/Backporting) to Python 2.6.xand 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.
* Python 2.7's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)) date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)), only Python 3.6.xand later are supported.
* Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible [remote code execution](https://en.wikipedia.org/wiki/Remote_code_execution) and [web cache poisoning](https://en.wikipedia.org/wiki/Cache_poisoning).

**2.5.1.1 IMPORTANCE OF PYTHON TO THE INTERNET**

Python is a general-purpose language sometimes referred to as utilitarian which is designed to be simple to read and write. The point that it’s not a complex language is important. The designers placed less of an emphasis on conventional syntax, which makes it easier to work with, even for non-programmers or developers. Furthermore, because it’s considered truly universal and used to meet various development needs, it’s a language that [offers a lot of options to programmers](https://www.python.org/about/success/) in general. If they begin working with Python for one job or career, they can easily jump to another, even if it’s in an unrelated industry. The language is used for system operations, web development, server and administrative tools, deployment, scientific modeling and much more. But, surprisingly, many developers don’t pick up Python as their primary language. Because it’s so easy to use and learn, they choose it as a second or third language. This may be another reason why it’s so popular among developers. Plus, it just so happens that one of the biggest tech companies in the world — Google — uses the language for a number of their applications. They even have a [developer portal devoted to Python](https://developers.google.com/edu/python/), with free classes offered including exercises, lecture videos and more. In addition, the rise in the use of the Django framework for web development and a decline in popularity of PHP has also contributed to Python’s success, but, ultimately, it’s the perfect storm — just the right amount of developer and official support, as well as demand.

**2.5.2 DESIGN OF PYTHON**

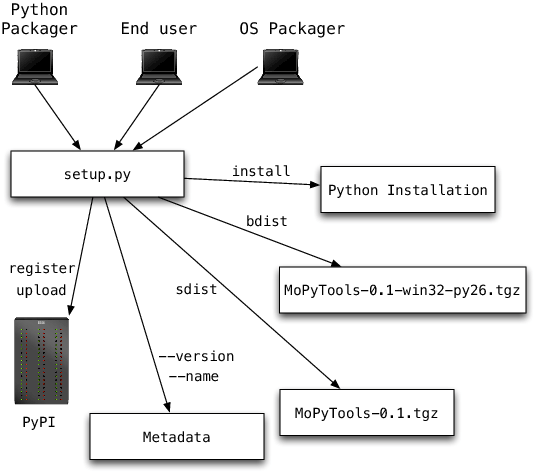
****

Fig 2.5.2 Design of Python

**2.5.3 PYTHON ARCHITECTURE**

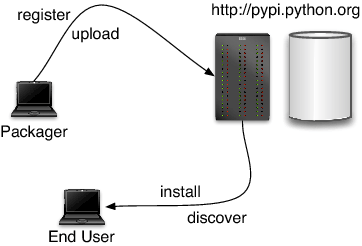
****

Fig 2.5.3 Python Architecture

PyPy’s Python Interpreter is written in Python and implements the full Python language. This interpreter very closely emulates the behaviour of Python. It contains the following key components:

* a byte code compiler responsible for producing Python code objects from the source code of a user application;
* a [byte code evaluator](https://doc.pypy.org/en/latest/interpreter.html) responsible for interpreting Python code objects;
* a [standard object space](https://doc.pypy.org/en/latest/objspace.html#standard-object-space), responsible for creating and manipulating the Python objects seen by the application.

The byte code compiler is the pre-processing phase that produces a compact byte code format via a chain of flexible passes (tokenizer, lexer, parser, abstract syntax tree builder, byte code generator). The byte code evaluator interprets this byte code. It does most of its work by delegating all actual manipulations of user objects to the object space. The latter can be thought of as the library of built-in types. It defines the implementation of the user objects, like integers and lists, as well as the operations between them, like addition or truth-value-testing. This division between byte code evaluator and object space gives a lot of flexibility. One can plug in different [object spaces](https://doc.pypy.org/en/latest/objspace.html) to get different or enriched behaviors’ of the Python objects.

**2.5.4 API TERMINOLOGY**

When using or building APIs, you will encounter these terms frequently:

* **HTTP (Hypertext Transfer Protocol)** is the primary means of communicating data on the web. HTTP implements a number of “methods,” which tell which direction data is moving and what should happen to it. The two most common are GET, which pulls data from a server, and POST, which pushes new data to a server.
* **URL (Uniform Resource Locator)** - An address for a resource on the web, such as https://programminghistorian.org/about. A URL consists of a **protocol** (http://), domain (programminghistorian.org), and optional **path** (/about). A URL describes the location of a specific resource, such as a web page. When reading about APIs, you may see the terms URL, request, URI, or endpoint used to describe adjacent ideas. This tutorial will prefer the terms URL and request to avoid complication. You can follow a URL or make a GET request in your browser, so you won’t need any special software to make requests in this tutorial.
* **JSON (JavaScript Object Notation)** is a text-based data storage format that is designed to be easy to read for both humans and machines. JSON is generally the most common format for returning data through an API, XML being the second most common.
* **REST (Representational State Transfer)** is a philosophy that describes some best practices for implementing APIs. APIs designed with some or all of these principles in mind are called REST APIs. While the API outlined in this lesson uses some REST principles, there is a great deal of disagreement around this term. For this reason, I do not describe the example APIs here as REST APIs, but instead as web or HTTP APIs.

This example of our Distant Reading Archive API pulls in data from a database, implements error handling, and can filter books by publication date. The database used is SQLite, a lightweight database engine that is supported in Python by default. SQLite files typically end with the .db. file extension.

Before we modify our code, first [download the example database from this location](https://programminghistorian.org/assets/creating-apis-with-python-and-flask/books.db) and copy the file to your api folder using your graphical user interface. The final version of our API will query this database when returning results to users.

**2.5.4.1 USER AUTHORIZATION**

One way to perform session tracking is to leverage the information that comes with User authorization. When a web server restricts access to some of its resources to only those clients that log in using a recognized username and password. After the client logs in, the username is available to a servlet through getRemoteUser ().

When use the username to track the session. Once a user has logged in, the browser remembers her user name and resends the name and password as the user views new pages on the site. A servlet can identify the user through her username and they’re by Track her session.

The biggest advantage of using user authorization to perform session tracking is that it’s easy to implement. Simply tell the protect a set of pages, and use getRemoteUser() to identify each client. Another advantage is that the technique works even when the user accesses your site form or exits her browser before coming back.

The biggest disadvantage of user authorization is that it requires each user to register for an account and then log in each time the starts visiting your site. Most users will tolerate registering and logging in as a necessary evil when they are accessing sensitive information, but it’s all overkill for simple session tracking. Another problem with user authorization is that a user cannot simultaneously maintain more than one session at the same site.

**2.5.5 DJANGO**

**Django** sometimes stylized as **django** is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language))-based [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [web framework](https://en.wikipedia.org/wiki/Web_framework) that follows the model-template-views (MTV) [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern_(computer_science)). It is maintained by the [Django Software Foundation](https://en.wikipedia.org/wiki/Django_Software_Foundation) (DSF), an American independent organization established as non-profit.Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability)  of components, less code, low coupling, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself).Python is used throughout, even for settings, files, and data models. Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Type_introspection) and configured via admin models. Despite having its own nomenclature, such as naming the callable objects generating the [HTTP](https://en.wikipedia.org/wiki/HTTP) responses "views",[[9]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-faq-mvc-9) the core Django framework can be seen as an [MVC](https://en.wikipedia.org/wiki/Model-view-controller) architecture.[[10]](https://en.wikipedia.org/wiki/Django_(web_framework)#cite_note-djangobook-mvc-10) It consists of an [object-relational mapper](https://en.wikipedia.org/wiki/Object-relational_mapping) (ORM) that mediates between [data models](https://en.wikipedia.org/wiki/Data_modeling) (defined as Python classes) and a [relational database](https://en.wikipedia.org/wiki/Relational_database) ("**M**odel"), a system for processing HTTP requests with a [web templating system](https://en.wikipedia.org/wiki/Web_template_system) ("**V**iew"), and a [regular-expression](https://en.wikipedia.org/wiki/Regular_expression)-based [URL](https://en.wikipedia.org/wiki/Uniform_Resource_Locator) dispatcher ("**C**ontroller")

**2.5.6 Werkzeug**

Werkzeug is a utility library for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)), in other words a toolkit for [Web Server Gateway Interface](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface) (WSGI) applications, and is licensed under a [BSD License](https://en.wikipedia.org/wiki/BSD_licenses). Werkzeug can realize software objects for request, response, and utility functions. It can be used to build a custom [software framework](https://en.wikipedia.org/wiki/Software_framework) on top of it and supports Python 2.7 and 3.5 and later.

**2.5.7 Jinja**

Jinja, also by Ronacher, is a [template engine](https://en.wikipedia.org/wiki/Template_engine_(web)) for the Python programming language and is licensed under a BSD License. Similar to the [Django web framework](https://en.wikipedia.org/wiki/Django_(web_framework)), it handles templates in a [sandbox](https://en.wikipedia.org/wiki/Sandbox_(computer_security)).

**2.6 HTML, JAVASCRIPT**

Hypertext Markup Language(HTML), the languages of the World Wide Web (WWW), allows users to produce web pages that included text, graphics and pointer to other web pages (Hyperlinks).

HTML is not a programming language, but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but Specialized to hypertext and adapted to the Web. The idea behind Hypertext one point to another point. We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed.Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop

HTML provides tags (special codes) to make the document look attractive.

HTML provides are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

##### **2.6.1 ADVANTAGE**

* + A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
  + HTML is platform independent
  + HTML tags are not case-sensitive.

**2.7 JAVA SCRIPT**

JavaScript is a compact, object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page. And Livewire enables you to create server-based applications similar to common gateway interface (CGI) programs.

In a client application for Navigator, JavaScript statements embedded in an HTML Page can recognize and respond to user events such as mouse clicks form input, and page navigation.

For example, you can write a JavaScript function to verify that users enter valid information into a form requesting a telephone number or zip code. Without any network transmission, an Html page with embedded Java Script can interpret the entered text and alert the user with a message dialog if the input is invalid or you can use JavaScript to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in) in response to the user opening or exiting a page.

**3.SYSTEM DESIGN**

**3.1 INTRODUCTION**

In our application we help a process the best recommendation system called collaborative filtering, with this one’s output is fed in to the others input. Upload the details of the raw materials in the sub-module name ‘raw materials' and view the raw materials of natural and rubber. Then send the rubber details to the vendor in the module view materials on the manager home page. While sending the supply he will provide the necessary supply id for raw materials and list out the supply items, not in stock. Then the supplier lets the company manager know whether he can provide the unsent quantity of raw material on a particular date. Get not-ready stock materials and get an alternative day for raw materials from the vendor module. After receiving an alternative date the company will fix the date of production to start the process. After uploading the raw materials sent to the analysis process, here we will analyze and find the appropriate rubber output. We find the best output and we suggest the rubber output to the input of the similar addictives in the making of rubber, so that the recommendation system works better with this concept of collaborative filtering, in this module analyze the report and then upload the raw materials for a testing report

**3.2 Module Description**

**Modules:**

1) Company Manager

2) Vendor

3) Process Unit

4) Admin

**3.2.1 COMPANY MANAGER**

This module gives the registration process with the company manager's details of name, email id, contact number, date of birth, and password. With this, the company manager can log in to the company page. If the manager is new, he can create a new account and go to the company's main page. Upload the details of the raw materials in the sub-module name ‘raw materials' and view the raw materials of natural and rubber. Then send the rubber details to the vendor in the module view materials on the manager home page. Get not-ready stock materials and get an alternative day for raw materials from the vendor module. After receiving an alternative date the company will fix the date of production to start the process and send it to the processing unit. Then a confirmation email is been sent to the vendor.

**3.2.2 VENDOR**

This module gives the registration process with the vendor details of name, email id, contact number, password, date of birth and address. With this, the vendor can log in to the vendor page. If the vendor is new, he can create a new account. After admin approval, the vendor can log in to the vendor page. Then receive raw materials from the company manager which are to be processed and sent on the due date. While sending the supply he will provide the necessary supply id for raw materials and list out the supply items, not in stock. Then the supplier lets the company manager know whether he can provide the unsent quantity of raw material on a particular date. After getting approval from the company manager he sent the remaining quantity with an alternative Day for raw materials and sends it to the company manager. If there is any product which could increase its price he let it know to the admin, where the admin does statistical report and get data so that it will be helpful for the company to store the stocks.

**3.2.3 PROCESS UNIT**

This module gives the registration process with the process unit details of name, email id, contact number, password, date of birth and address. With this, the Process unit can log in to the Process unit page. If the processing unit is created a new account then login to the process unit page. After the login process view report from the manager then update and upload the raw materials in synthetic and natural rubber. After uploading the raw materials sent to the analysis process, here we will analyze and find the appropriate rubber output. We find the best output and we suggest the rubber output to the input of the similar addictives in the making of rubber, so that the recommendation system works better with this concept of collaborative filtering, in this module analyze the report and then upload the raw materials for a testing report. After uploading the testing report send it to the admin.

**3.2.4 ADMIN**

This module allows the admin to login into the admin page. In the admin module, there are sub-modules named view testing report, approve vendor, testing, and view matching report. The admin views the project vendor details approved by the admin then view the testing report from the processing unit and then matches the common materials into the two tables after matching the process. As said above we get the output of rubber from our prediction, thus by this output, the admin can search for similar addictives in the making of rubber and he can suggest the output rubber type to add as addictive for the new process. So through this recommendation, we will get better product results in making of rubber.

**3.3 SYSTEM ARCHITECTURE**

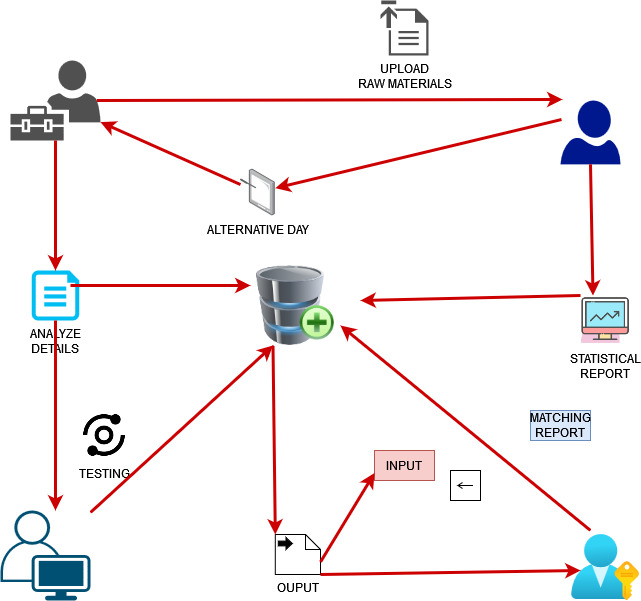
****

Fig 3.3 System Architecture

**4.SYSTEM TESTING AND IMPLEMENTATION**

**4.1. INTRODUCTION**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

**Component testing**

**Integration Testing**

**User Testing**

Fig 4.1 Software testing cycle

**4.2 Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**4.2.1. WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .We have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**4.2.2. BASIC PATH TESTING**

The established technique of flow graph with Cyclamate complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graphs.

Determine the Cyclamate complexity of the resultant flow graph, using formula:

V (G) =E-N+2 or

V (G) =P+1 or

V (G) =Number of Regions

Where V (G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**4.2.3 DATA FLOW TESTING**

This type of testing selects the path of the program, according to the location of the definition and use of variables. This kind of testing was used only when some local variable were declared. The definition-use chain method was used in this type of testing. These were particularly useful in nested statements.

**4.2.4 LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Test Scenario** | **Expected Result** | **Test Result** |
| 1 | Username is correct. Password is incorrect. | Username and Password is incorrect. | Username and Password is incorrect. |
| 2 | Username is incorrect. Password is correct. | Username and Password is incorrect. | Username and Password is incorrect. |
| 3 | Username is empty. Password is correct. | Username is required. | Username is required. |
| 4 | Username is correct. Password is empty. | Password is required. | Password is required |
| 5 | Both Username and Password is incorrect. | Username and Password is incorrect. | Username and Password is incorrect. |
| 6 | Both Username and Password is empty. | Username and Password is required. | Username and Password is required. |
| 7 | Both Username and Password is correct. | Login Successful. | Login Successful. |

Table 4.2.4 Loop Testing

**4.3 DATA FLOW**

1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3. A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data in the beginning process.
4. A Data flow to a data store means update (delete or change).
5. A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

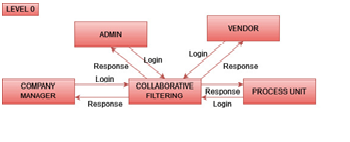
**4.3.1 DATA FLOW DIAGRAMS**

Fig 4.3.1 Data Flow Diagrams level 0

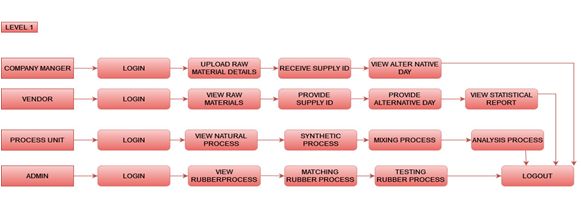
****

Fig 4.3.1 Data Flow Diagrams level 1

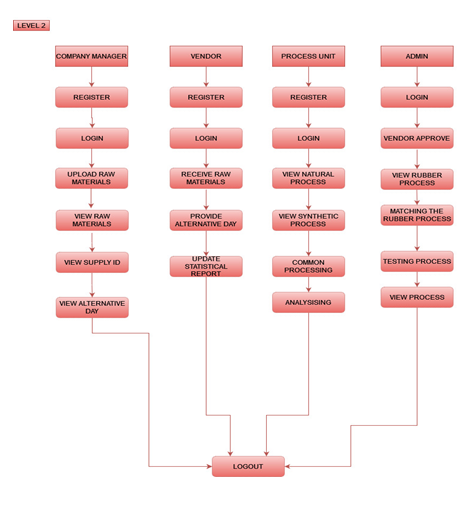
****

Fig 4.3.1 Data Flow Diagrams level 2

**4.3.2 USE CASE DIAGRAM**

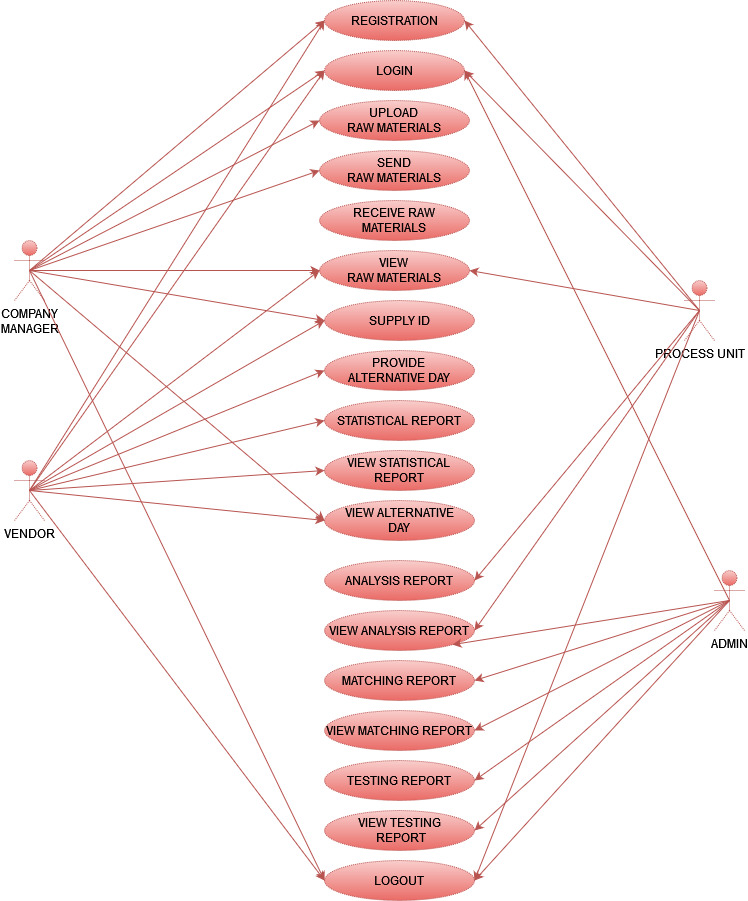
****

Fig 4.3.2 Use Case Diagram

**4.3.3 SEQUENCE DIAGRAM**

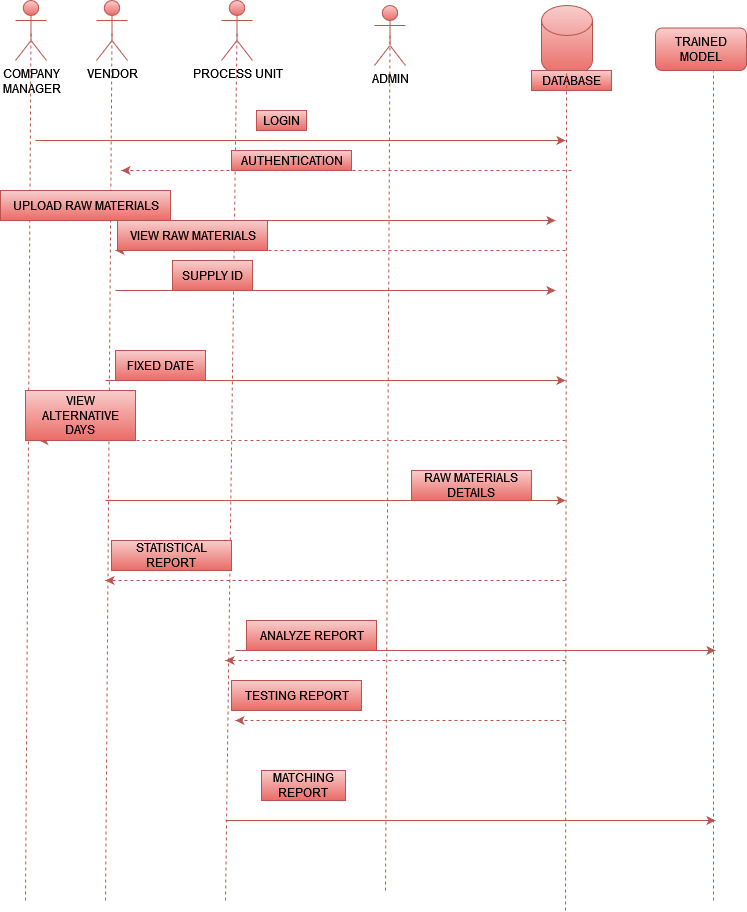
****

Fig 4.3.3 Sequence Diagram

**4.3.4 CLASS DIAGRAM**

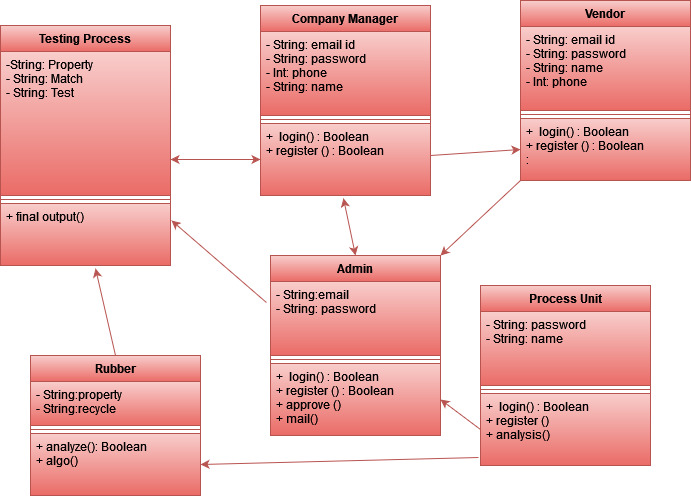
****

Fig 4.3.4 Class Diagram

**4.4 CODING**

**4.4.1 ADMIN**

from django.shortcuts import render,redirect

from django.contrib import messages

from vendor. models import \*

from process\_unit. models import \*

# Create your views here.

def admin\_home(request):

return render(request, 'admin/admin\_home.html')

def admin\_login(request):

if request.method == "POST":

email = request.POST["email"]

password = request.POST["password"]

print(email)

if email == "admin@gmail.com" and password == "admin":

print(email)

request.session['admin'] = "admin@gmail.com"

messages.info(request, "Successfully Registered ")

return render(request,'admin/admin\_home.html')

elif email != "admin@gmail.com":

messages.error(request, "Wrong Mail id")

return render (request,'admin/login.html')

elif password != "admin":

messages.error(request,"wrong password")

return render (request, 'admin/login.html')

else:

return render(request,'admin/login.html')

return render(request,'admin/login.html')

def approve\_vendor(request):

if 'admin' in request.session:

values = register.objects.filter(vendor\_approve=False)

return render(request,'admin/approve\_vendor.html',{'values': values})

def true\_vendor(request,id):

if "admin" in request.session:

values = register.objects.get(id=id)

values.vendor\_approve = True

values.save()

messages.info(request, "successfully approved for vendor")

return redirect('/approve\_vendor/')

def admin\_logout(request):

if 'admin' in request.session:

request.session.pop('admin',None)

messages.success(request,'logout already successfully')

return redirect('/')

else:

messages.success(request, 'session logged out')

return redirect('/admin\_logout/')

def testing\_table1\_table2(request):

if 'admin' in request.session:

value = analyse\_rubber.objects.all()

values = testing\_rubber.objects.all()

return render(request, 'admin/testing.html', {'value':value,'values': values})

def matching(request, id):

if 'admin' in request.session:

value = analyse\_rubber.objects.filter(send\_report=False)

values = testing\_rubber.objects.get(id=id)

r = values.id

print(values.property)

for i in value:

i.send\_report=True

i.save()

print(i.property, 1)

if i.property == values.property and i.recycle == values.recycle and\

i.compound == values.compound and i.polymer== values.polymer:

f = i.rubber\_type

print(f)

st = testing\_rubber.objects.filter(id=r).update(output=f)

messages.info(request, 'successfully added your recommend')

print(st)

else:

messages.info(request, 'No Matching for our materials')

return redirect('/testing\_table1\_table2/')

def matching\_report (request):

if 'admin' in request.session:

values = testing\_rubber.objects.filter(matching=False)

return render(request,'admin/testing.html',{'values': values})

def view\_matching\_report (request,id):

if "admin" in request.session:

values = testing\_rubber.objects.get(id=id)

values.matching = True

values.save()

messages.info(request, "successfully sent")

return redirect('/matching\_report/')

def matching\_report\_true (request):

if 'admin' in request.session:

values = testing\_rubber.objects.filter(matching=True)

return render(request,'admin/matching\_report.html',{'values': values})

def view\_testing\_matching\_report\_true(request):

if 'admin' in request.session:

values = testing\_rubber.objects.filter(send\_testing=True)

return render(request,'admin/view\_testing\_report.html',{'values': values})

**4.4.2 COMPANY**

from django.shortcuts import render, redirect

from django.shortcuts import \*

from . models import \*

from django.db import IntegrityError

from django.contrib import messages

from django.core.mail import send\_mail

from vendor.models import \*

# Create your views here.

def company\_home(request):

return render(request,'company/company\_home.html')

def company\_login(request):

if request.method == 'POST':

email = request.POST['email']

password = request.POST['password']

try:

r = co\_register.objects.get(email=email, password=password)

request.session['company'] = r.email

print(request.session['company'])

if r is not None:

messages.info(request, 'welcome')

return redirect('/company\_home/')

except co\_register.DoesNotExist as e:

messages.info(request, 'Email does not exists')

return redirect('/company\_login/')

else:

return render(request, 'company/login.html')

def company\_register(request):

if request.method == 'POST':

username = request.POST['username']

email = request.POST['email']

contact\_no = request.POST['contact\_no']

age = request.POST['age']

address = request.POST['address']

password = request.POST['password']

if username and email and contact\_no and age and address and password is not None:

try:

co\_register(username=username, email=email, contact\_no=contact\_no, age=age, address=address,

password=password).save()

messages.info(request, "successfully created")

return redirect('/company\_login/')

except IntegrityError as e:

messages.info(request, "Email already exists")

return redirect('/company\_register/')

else:

messages.info(request, "Fields Should not be empty")

return render(request, 'company/login.html')

def synthetic\_raw\_materials(request):

if request.method == "POST":

regi = co\_register.objects.get(email=request.session['company'])

syn= synthetic\_rubber()

syn.mineral\_type = request.POST['mineral\_type']

syn.email =regi.email

syn.polymer\_type = request.POST['polymer\_type']

syn.gases = request.POST['gases']

syn.industry = request.POST['industry']

syn.product = request.POST['product']

syn.storage\_type = request.POST['storage\_type']

syn.chemical\_type = request.POST['chemical\_type']

syn.property = request.POST['property']

syn.rubber\_type = request.POST['rubber\_type']

syn.duration = request.POST['duration']

syn.need\_chemical = request.POST['need\_chemical']

syn.save()

return render(request, 'company/synthetic\_raw\_material.html')

def natural\_raw\_materials(request):

if request.method == "POST":

old = co\_register.objects.get(email=request.session['company'])

new = natural\_rubber()

new.latex\_method = request.POST['latex\_method']

new.email = old.email

new.latex\_char = request.POST['latex\_char']

new.acid = request.POST['acid']

new.Time\_taken = request.POST['Time\_taken']

new.polymer = request.POST['polymer']

new.industry = request.POST['industry']

new.product = request.POST['product']

new.type\_of\_trees = request.POST['type\_of\_trees']

new.type\_of\_plants = request.POST['type\_of\_plants']

new.property = request.POST['property']

new.need\_latex =request.POST['need\_latex']

new.save()

return render(request, 'company/raw\_material.html')

def company\_logout(request):

if 'company' in request.session:

request.session.pop('company',None)

messages.success(request, 'logout already successfully')

return redirect('/')

else:

messages.success(request, 'session logged out')

return redirect('/company\_logout/')

def natural\_send\_vendor(request):

if 'company' in request.session:

natural = natural\_rubber.objects.filter(send\_natural=False)

return render(request,'company/send\_to\_vendor\_natural.html', {"natural":natural})

def natural\_company(request,id):

if 'company' in request.session:

natural = natural\_rubber.objects.get(id=id)

natural.send\_natural = True

natural.save()

messages.info(request, "send to vendor in natural\_rubber raw materials details")

return redirect('/natural\_send\_vendor/')

def synthetic\_send\_vendor(request):

if "company" in request.session:

synthetic = synthetic\_rubber.objects.filter(send\_synthetic=False)

return render(request, 'company/send\_to\_vendor\_syn.html', {"synthetic":synthetic})

def synthetic\_company(request,id):

if 'company' in request.session:

synthetic = synthetic\_rubber.objects.get(id=id)

synthetic.send\_synthetic = True

synthetic.save()

messages.info(request, "send to vendor in synthetic rubber raw material details")

return redirect("/synthetic\_send\_vendor/")

def view\_alternative\_day\_nal(request):

if "company" in request.session:

natural = natural\_rubber.objects.filter(send\_fixing\_date=True)

return render(request, 'company/view\_natural\_alternativeday.html', {"natural":natural})

def view\_alternative\_day\_syn(request):

if "company" in request.session:

synthetic = synthetic\_rubber.objects.filter(send\_fixing\_date1=True)

return render(request, 'company/view\_synthetic\_alternative\_day.html', {"synthetic":synthetic})

def natural\_send\_process\_unit(request):

if 'company' in request.session:

natural = natural\_rubber.objects.filter(send\_process\_nal=False)

return render(request,'company/nat\_send\_to\_process\_unit.html', {"natural":natural})

def natural\_company\_process\_unit(request,id):

if 'company' in request.session:

natural = natural\_rubber.objects.get(id=id)

natural.send\_process\_nal = True

natural.save()

messages.info(request, "send to process unit in natural rubber raw materials details")

return redirect('/natural\_send\_process\_unit/')

def syn\_send\_process\_unit(request):

if 'company' in request.session:

synthetic = synthetic\_rubber.objects.filter(send\_process\_syn=False)

return render(request,'company/syn\_send\_to\_process\_unit.html', {"synthetic":synthetic})

def syn\_company\_process\_unit(request,id):

if 'company' in request.session:

synthetic = synthetic\_rubber.objects.get(id=id)

synthetic.send\_process\_syn = True

synthetic.save()

messages.info(request, "send to process unit in synthetic rubber raw materials details")

return redirect('/syn\_send\_process\_unit/')

def send\_mail\_vendor\_natural(request, id):

if "company" in request.session:

natural = register.objects.get(id=id)

msg1 = '''You have allocated alternate day for remaining raw materials,

make sure to confirm the order on the particular day'''

send\_mail(

'Subject here',

msg1,

"authentication4email@gmail.com",

[natural.email],

fail\_silently=False,

)

natural.mail\_vanish = True

natural.save()

messages.info(request, "Sucessfully Sent To Vendor")

return redirect("/view\_alternative\_day\_nal/")

def send\_mail\_synthetic(request, id):

if "company" in request.session:

natural = register.objects.get(id=id)

msg1 = '''You have allocated alternate day for remaining raw materials,

make sure to confirm the order on the particular day'''

send\_mail(

'Subject here',

msg1,

"authentication4email@gmail.com",

[natural.email],

fail\_silently=False,

)

natural.mail\_vanish = True

natural.save()

messages.info(request, "Sucessfully Sent To Vendor")

return redirect("/view\_alternative\_day\_syn/")

**4.4.3 PROCESS UNIT**

from django.shortcuts import render, redirect

from django.shortcuts import \*

from . models import \*

from django.db import IntegrityError

from django.contrib import messages

from company.models import \*

# Create your views here.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt, seaborn as sns

from sklearn.preprocessing import LabelEncoder

import warnings

warnings.filterwarnings('ignore')

from sklearn.neural\_network import MLPClassifier

def process\_home(request):

return render(request,'process/process\_home.html')

def process\_login(request):

if request.method == 'POST':

email = request.POST['email']

password = request.POST['password']

try:

r = unit\_register.objects.get(email=email, password=password)

request.session['process'] = r.email

print(request.session['process'])

if r is not None:

messages.info(request, 'welcome')

return redirect('/process\_home/')

except unit\_register.DoesNotExist as e:

messages.info(request, 'Email does not exists')

return redirect('/process\_login/')

else:

return render(request, 'process/login.html')

def process\_register(request):

if request.method == 'POST':

username = request.POST['username']

email = request.POST['email']

contact\_no = request.POST['contact\_no']

age = request.POST['age']

address = request.POST['address']

password = request.POST['password']

try:

unit\_register(username=username, email=email, contact\_no=contact\_no, age=age, address=address,

password=password).save()

messages.info(request, "successfully created")

return redirect('/process\_login/')

except IntegrityError as e:

messages.info(request, "Email already exists")

return redirect('/process\_register/')

return render(request, 'process/login.html')

def syn\_view\_process\_unit(request):

if 'process' in request.session:

synthetic = synthetic\_rubber.objects.filter(send\_process\_syn=True)

return render(request,'process/view\_synthetic.html', {"synthetic":synthetic})

def natural\_view\_process\_unit(request):

if 'process' in request.session:

natural = natural\_rubber.objects.filter(send\_process\_nal=True)

return render(request,'process/view\_natural.html', {"natural":natural})

def process\_logout(request):

if 'process' in request.session:

request.session.pop('process',None)

messages.success(request,'logout already successfully')

return redirect('/')

else:

messages.success(request, 'session logged out')

return redirect('/process\_logout/')

def analyse\_form(request):

if request.method == "POST":

property = request.POST['property']

recycle = request.POST['recycle']

compound = request.POST['compound']

chemical\_formula = request.POST['chemical\_formula']

polymer = request.POST['polymer']

mixing = request.POST['mixing']

made\_from\_type = request.POST['made\_from\_type']

type=request.POST['type']

analyse\_rubber(property=property,recycle=recycle, compound=compound,chemical\_formula=chemical\_formula,

polymer=polymer,mixing=mixing, made\_from\_type=made\_from\_type,type=type).save()

return render(request, 'process/analyse\_form.html')

def view\_analyse\_form(request):

if 'process' in request.session:

natural = analyse\_rubber.objects.all()

return render(request,'process/view\_analyse\_form.html', {"natural":natural})

def algo(datas,r):

data = pd.read\_csv('final.csv')

data\_x = data.iloc[:, :-1]

data\_y = data.iloc[:, -1]

string\_datas = [i for i in data\_x.columns if data\_x.dtypes[i] == np.object\_]

LabelEncoders = []

for i in string\_datas:

newLabelEncoder = LabelEncoder()

data\_x[i] = newLabelEncoder.fit\_transform(data\_x[i])

LabelEncoders.append(newLabelEncoder)

ylabel\_encoder = None

if type(data\_y.iloc[1]) == str:

ylabel\_encoder = LabelEncoder()

data\_y = ylabel\_encoder.fit\_transform(data\_y)

model = MLPClassifier()

model.fit(data\_x, data\_y)

value = {data\_x.columns[i]: datas[i] for i in range(len(datas))}

l = 0

for i in string\_datas:

z = LabelEncoders[l]

value[i] = z.transform([value[i]])[0]

l += 1

value = [i for i in value.values()]

predicted = model.predict([value])

print(12334455)

if ylabel\_encoder:

predicted = ylabel\_encoder.inverse\_transform(predicted)

return predicted[0]

df = pd.DataFrame(pd.read\_excel("final.xlsx"))

read\_file = pd.read\_excel("final.xlsx")

read\_file.to\_csv("final.csv", header=True, index=False)

def disapper\_analyse\_form(request, id):

natural = analyse\_rubber.objects.get(id=id)

natural.send\_analyse = True

natural.save()

r=natural.id

inputvar = []

property = natural.property

recycle = natural.recycle

compound = natural.compound

chemical\_formula = natural.chemical\_formula

polymer = natural.polymer

mixing = natural.mixing

made\_from\_type = natural.made\_from\_type

type = natural.type

inputvar.append(property)

inputvar.append(recycle)

inputvar.append(compound)

inputvar.append(chemical\_formula)

inputvar.append(polymer)

inputvar.append(mixing)

inputvar.append(made\_from\_type)

inputvar.append(type)

print('input:', inputvar)

f=algo(inputvar,r)

print('OUTPUT:',f)

st = analyse\_rubber.objects.filter(id=r).update(rubber\_type=f)

# return render(request,'process/view\_analyse\_form.html', {'natural': natural})

return redirect('/view\_analyse\_form/')

def testing\_analyse\_form(request):

if request.method == "POST":

property = request.POST['property']

recycle = request.POST['recycle']

compound = request.POST['compound']

chemical\_formula = request.POST['chemical\_formula']

polymer = request.POST['polymer']

mixing = request.POST['mixing']

made\_from\_type = request.POST['made\_from\_type']

type=request.POST['type']

testing\_rubber(property=property,recycle=recycle, compound=compound,chemical\_formula=chemical\_formula,

polymer=polymer,mixing=mixing, made\_from\_type=made\_from\_type,type=type).save()

return render(request, 'process/testing\_analyse\_form.html')

def send\_to\_admin\_testing(request):

if 'process' in request.session:

synthetic = testing\_rubber.objects.filter(send\_testing=False)

return render(request,'process/send\_to\_matching.html', {"synthetic":synthetic})

def true\_admin\_testing(request,id):

if "process" in request.session:

synthetic = testing\_rubber.objects.get(id=id)

synthetic.send\_testing = True

synthetic.save()

messages.info(request, "successfully sent to admin")

**4.4.4 VENDOR**

from django.shortcuts import render, redirect

from django.shortcuts import \*

from . models import \*

from django.db import IntegrityError

from django.contrib import messages

from company.models import \*

import random

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt, seaborn as sns

import warnings

warnings.filterwarnings('ignore')

def vendor\_home(request):

return render(request,'vendor/vendor\_home.html')

def vendor\_login(request):

if request.method == 'POST':

email = request.POST['email']

password = request.POST['password']

try:

r = register.objects.get(email=email, password=password, vendor\_approve=True)

request.session['vendor'] = r.email

print(request.session['vendor'])

if r is not None:

messages.info(request, 'welcome')

return redirect('/vendor\_home/')

except register.DoesNotExist as e:

messages.info(request, 'Email does not exists')

return redirect('/vendor\_login/')

else:

messages.info(request, 'waiting for admin approve')

return redirect('/vendor\_login/')

else:

return render(request, 'vendor/login.html')

def vendor\_register(request):

if request.method == 'POST':

username = request.POST['username']

email = request.POST['email']

contact\_no = request.POST['contact\_no']

age = request.POST['age']

address = request.POST['address']

password = request.POST['password']

try:

register(username=username, email=email, contact\_no=contact\_no, age=age, address=address,

password=password).save()

messages.info(request, "successfully created")

return redirect('/vendor\_login/')

except IntegrityError as e:

messages.info(request, "Email already exists")

return redirect('/vendor\_register/')

return render(request, 'vendor/login.html')

def vendor\_logout(request):

if 'vendor' in request.session:

request.session.pop('vendor',None)

messages.success(request,'logout already successfully')

return redirect('/')

else:

messages.success(request, 'session logged out')

return redirect('/vendor\_logout/')

def view\_natural(request):

natural = natural\_rubber.objects.all()

if 'vendor' in request.session:

natural = natural\_rubber.objects.all()

return render(request, 'vendor/view\_generator.html', {'natural': natural})

def generate\_natural\_id(request,id):

st = natural\_rubber.objects.get(id=id)

r = random.randint(1000,2000)

st.natural\_id = r

st.save()

return redirect('/vendor\_home/')

def view\_synthetic(request):

synthetic = synthetic\_rubber.objects.all()

if 'vendor' in request.session:

synthetic = synthetic\_rubber.objects.all()

return render(request, 'vendor/view\_syn\_supplyid.html', {'synthetic': synthetic})

def generate\_synthetic\_id(request,id):

st = synthetic\_rubber.objects.get(id=id)

r = random.randint(1000,2000)

st.syn\_id = r

st.save()

return redirect('/vendor\_home/')

def give\_email(request, id):

sd = natural\_rubber.objects.get(id=id)

reg = register.objects.get(id=id)

# print(reg.email)

a = reg.email

print(a)

natural\_rubber(give\_vendor=a).save()

return redirect('/vendor\_home/')

def give\_email\_syn(request, id):

sd = natural\_rubber.objects.get(id=id)

reg = register.objects.get(id=id)

# print(reg.email)

a = reg.email

print(a)

natural\_rubber(give\_vendor\_email=a).save()

return redirect('/vendor\_home/')

def view\_natural\_rubber(request):

natural = natural\_rubber.objects.all()

if 'vendor' in request.session:

natural = natural\_rubber.objects.all()

return render(request, 'vendor/view\_natural\_materials.html', {'natural': natural})

def view\_synthetic\_rubber(request):

synthetic = synthetic\_rubber.objects.filter(send\_synthetic=True)

if 'vendor' in request.session:

synthetic = synthetic\_rubber.objects.filter(send\_synthetic=True)

return render(request, 'vendor/view\_synthetic\_materials.html', {'synthetic': synthetic})

def check\_availability(request, id):

if request.method == "POST":

availability = request.POST['availability']

print(availability, "1")

print(id)

data1 = natural\_rubber.objects.get(id=id)

data1.availability = availability

data1.save()

a = data1.need\_latex

b = data1.availability

print(a)

print(b)

remaining = int(a) - int(b)

data1.remaining = remaining

print(remaining)

data1.check\_remaining = True

data1.save()

return render(request, 'vendor/view\_natural\_materials.html')

def check\_availability\_syn(request, id):

if request.method == "POST":

availability = request.POST['availability']

print(availability, "1")

print(id)

data1 = synthetic\_rubber.objects.get(id=id)

data1.availability = availability

data1.save()

a = data1.need\_chemical

b = data1.availability

print(a)

print(b)

remaining = int(a) - int(b)

data1.remaining = remaining

data1.check\_remaining = True

data1.save()

return render(request, 'vendor/view\_synthetic\_materials.html')

def view(request):

natural = natural\_rubber.objects.all()

return render(request, 'vendor/alternative\_day.html', {'natural': natural})

def view1(request):

synthetic = synthetic\_rubber.objects.all()

return render(request, 'vendor/set\_syn\_alterday.html', {'synthetic': synthetic})

def fixing\_date\_natural(request,id):

if request.method == "POST":

fixing\_date = request.POST['fixing\_date']

print(fixing\_date, "1")

data = natural\_rubber.objects.get(id=id)

data.fixing\_date = fixing\_date

data.save()

return render(request,'vendor/alternative\_day.html')

def fixing\_date\_synthetic(request,id):

if request.method == "POST":

fixing\_date1 = request.POST['fixing\_date1']

print(fixing\_date1, "1")

data = synthetic\_rubber.objects.get(id=id)

data.fixing\_date1 = fixing\_date1

data.save()

return render(request,'vendor/set\_syn\_alterday.html')

def send\_fixing\_date\_natural(request):

values = natural\_rubber.objects.filter(send\_fixing\_date=False)

if 'vendor' in request.session:

values = natural\_rubber.objects.filter(send\_fixing\_date=False)

return render(request,'vendor/send\_alternative\_day.html',{'values': values})

return render(request, 'vendor/send\_alternative\_day.html', {'values': values})

def disappear\_fixing\_date\_natural(request, id):

if "vendor" in request.session:

values = natural\_rubber.objects.get(id=id)

values.send\_fixing\_date = True

values.save()

messages.info(request, "sent to alternative day for raw materials")

return redirect('/send\_fixing\_date\_natural/')

def send\_fixing\_date\_synthetic(request):

values = synthetic\_rubber.objects.filter(send\_fixing\_date1=False)

if 'vendor' in request.session:

values = synthetic\_rubber.objects.filter(send\_fixing\_date1=False)

return render(request,'vendor/send\_alter\_syn.html',{'values': values})

return render(request, 'vendor/send\_alter\_syn.html', {'values': values})

def disappear\_fixing\_date\_syn(request, id):

if "vendor" in request.session:

values = synthetic\_rubber.objects.get(id=id)

values.send\_fixing\_date1 = True

values.save()

messages.info(request, "sent to alternative day for raw materials")

return redirect('/send\_fixing\_date\_synthetic/')

def graph\_view(request):

return render(request, 'vendor/graph.html')

def view\_graph(request):

# df = pd.DataFrame(pd.read\_excel("Month Natural Rubber.xlsx"))

# read\_file = pd.read\_excel("Month Natural Rubber.xlsx")

# read\_file.to\_csv("Month Natural Rubber.csv", header=True, index=False)

# df1 = pd.DataFrame(pd.read\_csv("Month Natural Rubber.csv"))

df1 = pd.read\_csv("Month Natural Rubber.csv")

sns.scatterplot(x='year', y='total', data=df1)

plt.show()

# df3 = pd.read\_csv("Natural Rubber Production.csv")

# sns.scatterplot(x='Date', y='Metric Tons', data=df3)

# plt.show()

# df5 = pd.read\_csv("Synthetic Rubber.csv")

# sns.scatterplot(x='year', y='Total', data=df5)

# plt.show()

return redirect('/graph\_view/')

def vendor\_natural\_graph(request):

# if 'vendor' in request.session:

values = register.objects.filter(graph1=False)

return render(request,'vendor/graph.html',{'values': values})

def send\_natural\_graph(request):

# # if "vendor" in request.session:

# values = register.objects.get(id=id)

# values.graph1 = True

# values.save()

messages.info(request, "successfully sent to admin")

return redirect('/vendor\_home/')

def view\_graph1(request):

# df2 = pd.DataFrame(pd.read\_excel("Natural Rubber Production.xlsx"))

# read\_file = pd.read\_excel("Natural Rubber Production.xlsx")

# read\_file.to\_csv("Natural Rubber Production.csv", header=True, index=False)

# df3 = pd.DataFrame(pd.read\_csv("Natural Rubber Production.csv"))

df3 = pd.read\_csv("Natural Rubber Production.csv")

sns.scatterplot(x='Date', y='Metric Tons', data=df3)

plt.show()

return redirect('/graph\_view/')

def vendor\_syn\_graph(request):

if 'vendor' in request.session:

values = register.objects.filter(graph2=False)

return render(request,'vendor/graph.html',{'values': values})

def send\_syn\_graph(request,id):

if "vendor" in request.session:

values = register.objects.get(id=id)

values.graph2 = True

values.save()

messages.info(request, "successfully sent to admin")

return redirect('/vendor\_syn\_graph/')

def view\_graph2(request):

# df4 = pd.DataFrame(pd.read\_excel("Synthetic Rubber.xlsx"))

# read\_file = pd.read\_excel("Synthetic Rubber.xlsx")

# read\_file.to\_csv("Synthetic Rubber.csv", header=True, index=False)

# df5 = pd.DataFrame(pd.read\_csv("Synthetic Rubber.csv"))

df5 = pd.read\_csv("Synthetic Rubber.csv")

sns.scatterplot(x='year', y='Total', data=df5)

plt.show()

return redirect('/graph\_view/')

def vendor\_syn\_nan\_graph(request):

if 'vendor' in request.session:

values = register.objects.filter(graph3=False)

return render(request,'vendor/graph.html',{'values': values})

def send\_syn\_nan\_graph(request,id):

if "vendor" in request.session:

values = register.objects.get(id=id)

values.graph3 = True

values.save()

messages.info(request, "successfully sent to admin")

return redirect('/vendor\_syn\_nan\_graph/')

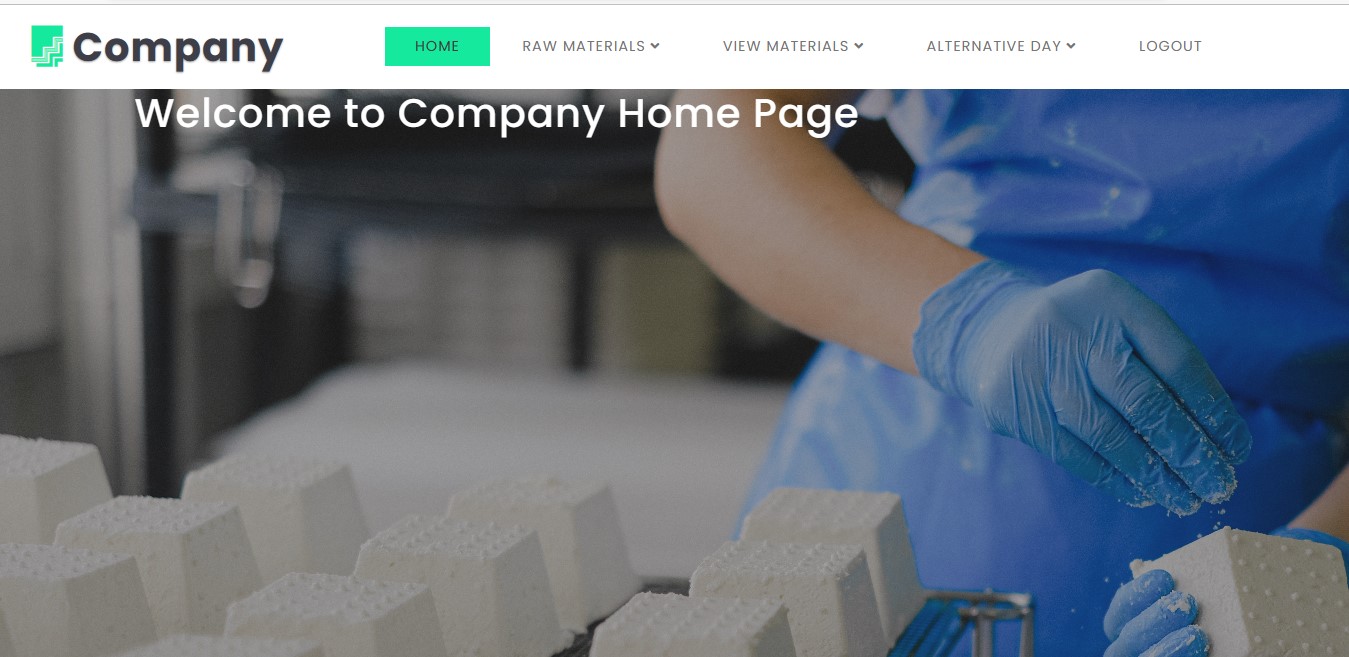
**5.CONCLUSION & FUTURE ENHANCEMENT**

**5.1 CONCLUSION**

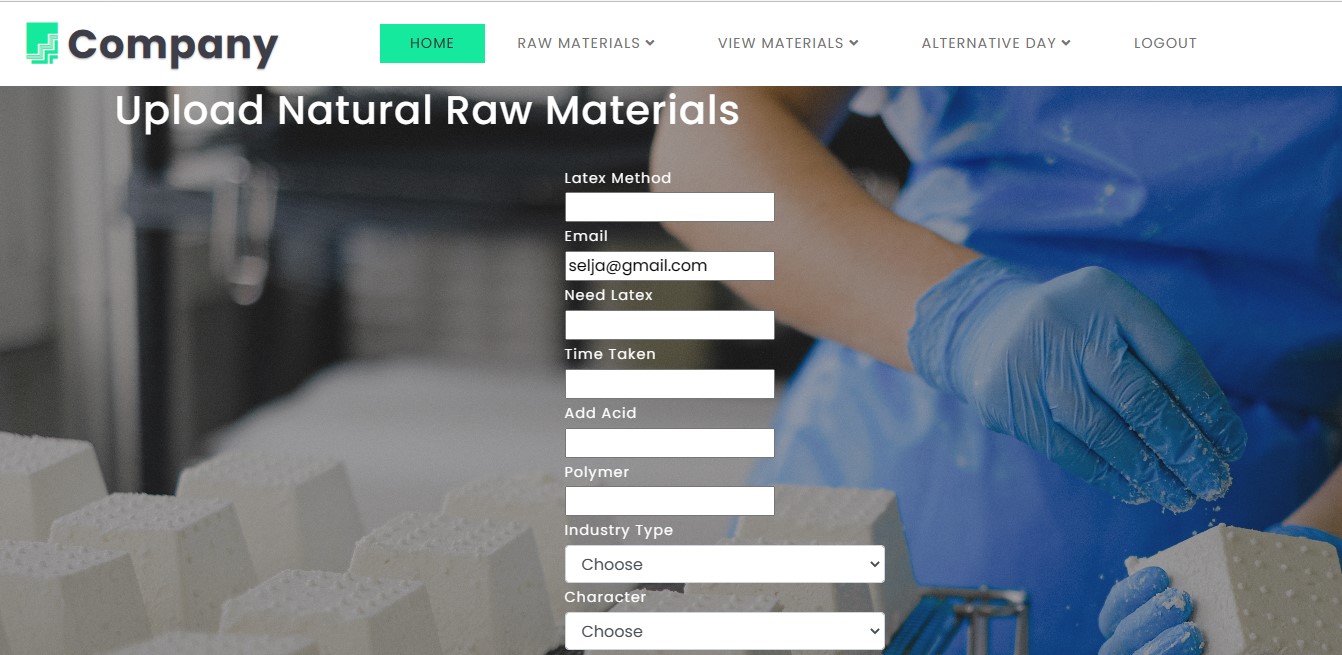
We have proposed a recommendation system called collaborative filtering, This doesn’t alone check for the recommendation but also the similarities so that in the similar process one’s output can be fed into the input of the other as a recommendation. Which will become an apt one because of its similar nature? We use this in the rubber industry, where our rubber from one process is fed into the other as an input. So the final output will be a quality one. We achieve it through our analysis process.

**5.2 FUTURE WORK**

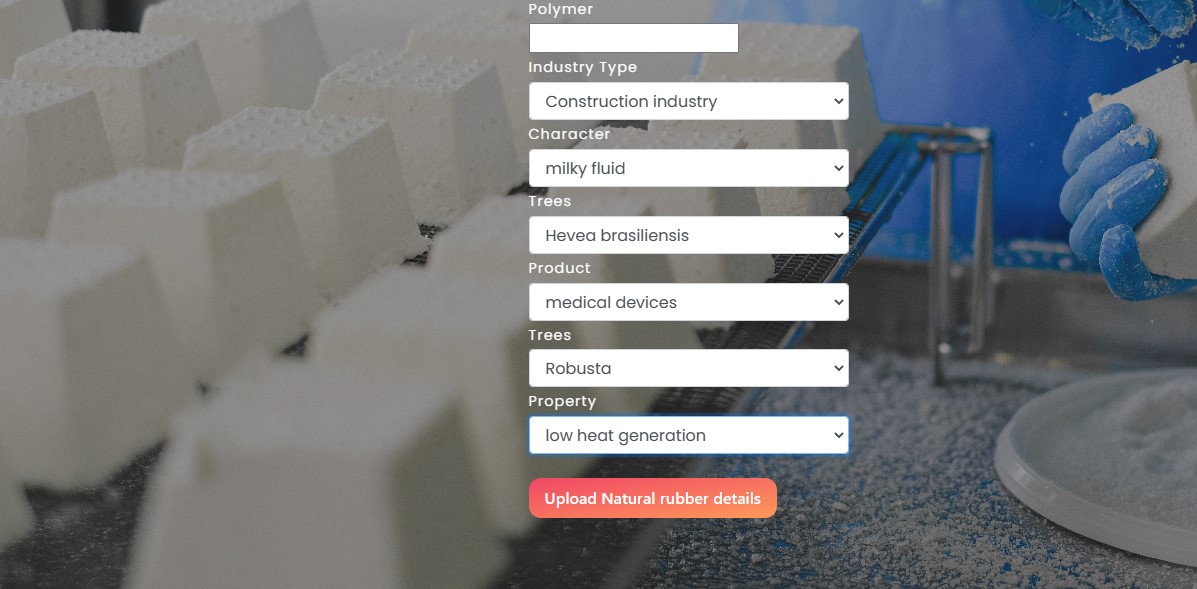
The multi-layer classifier of neural network learning helps to categorize the similarities and the recommendation for the production of the material or product according to the client need. **Collaborative Filtering** is the most famous application suggestion engine and is based on calculated guesses; the people who liked the product will enjoy the same product in the future. In future we will switch to the content-based filter, because in this the recommendations are unique to the user, the model does not require any information about other users. This makes scaling to a large number of users easier.

****

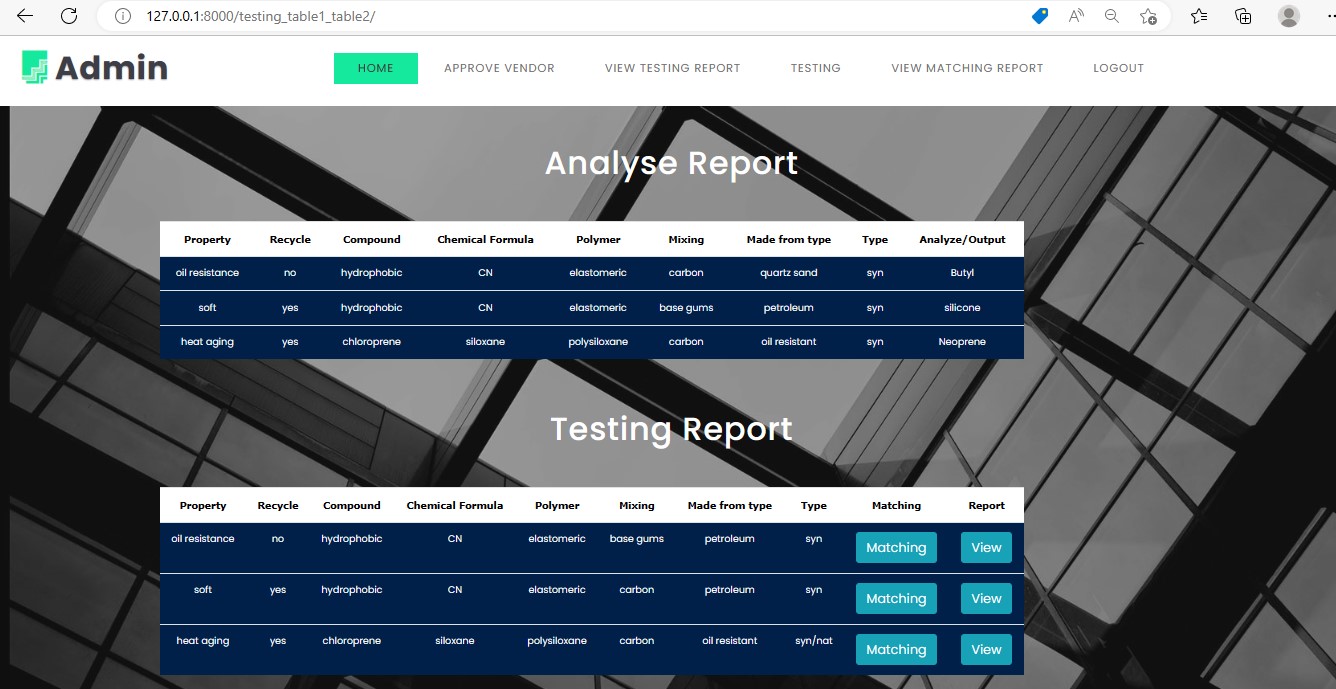
**A1.HOME PAGE**

****

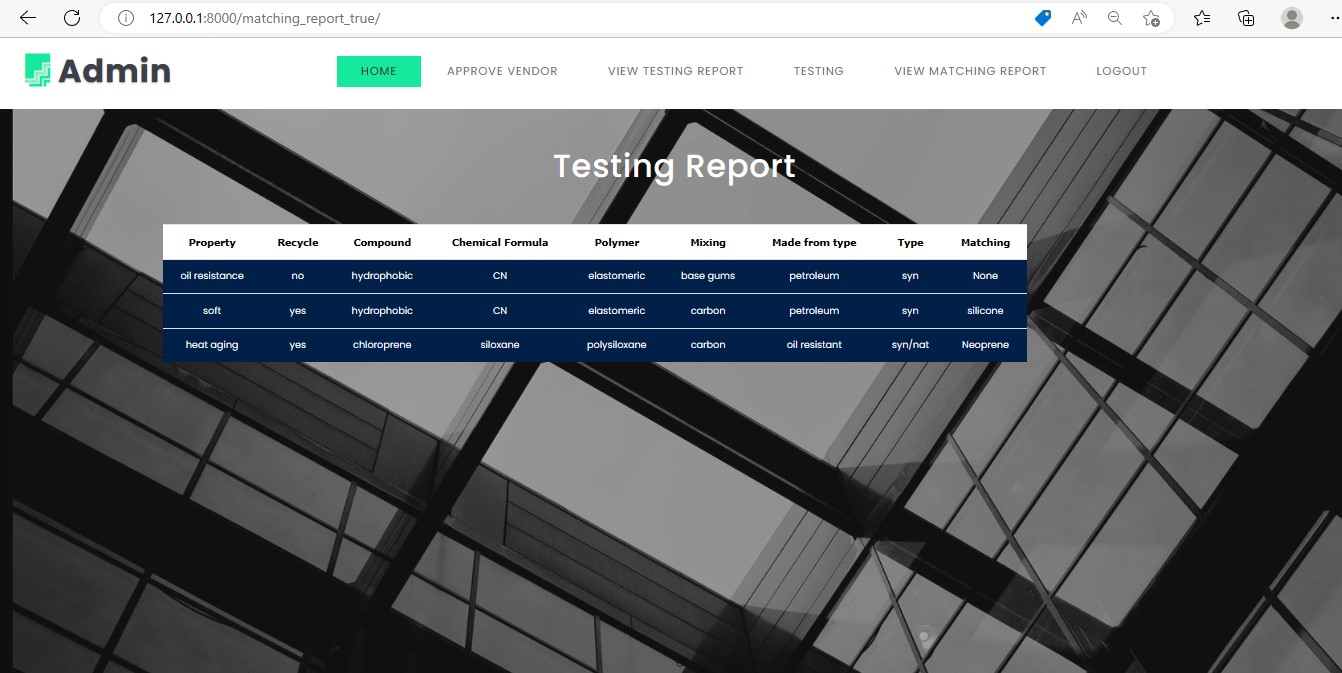
**A2.UPLOAD NATURAL RAW MATERIALS**

****

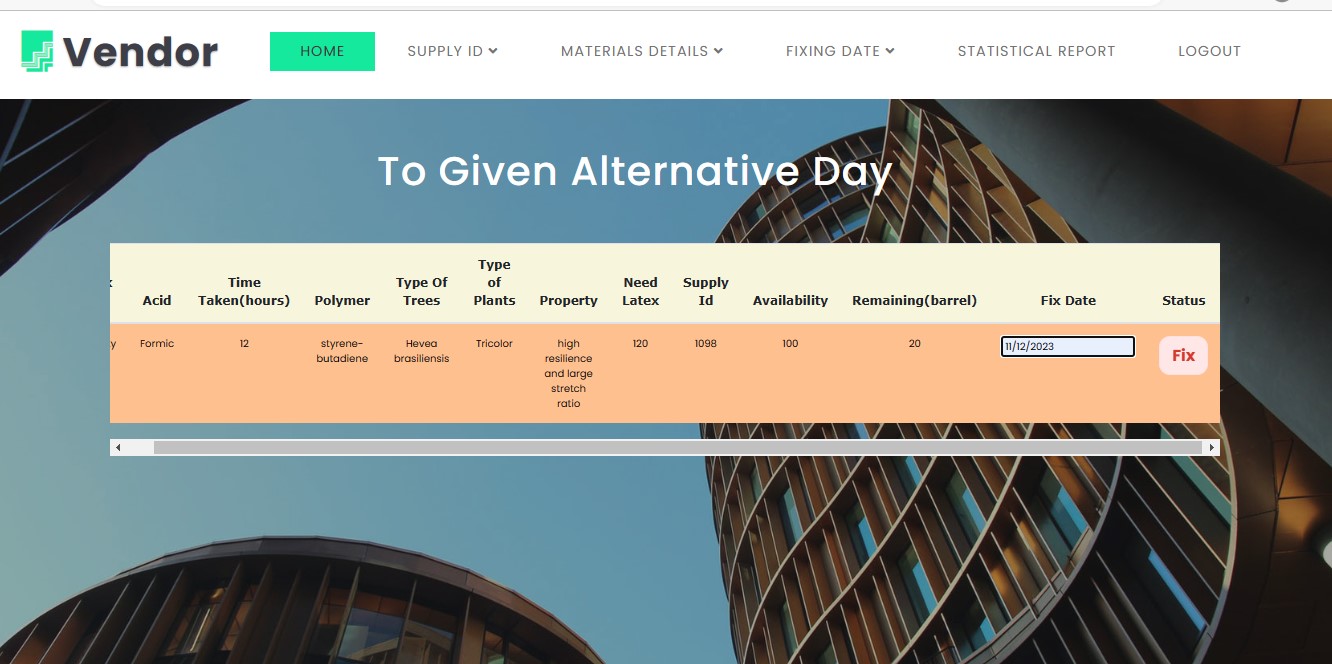
**A3. RAW MATERIALS**

****

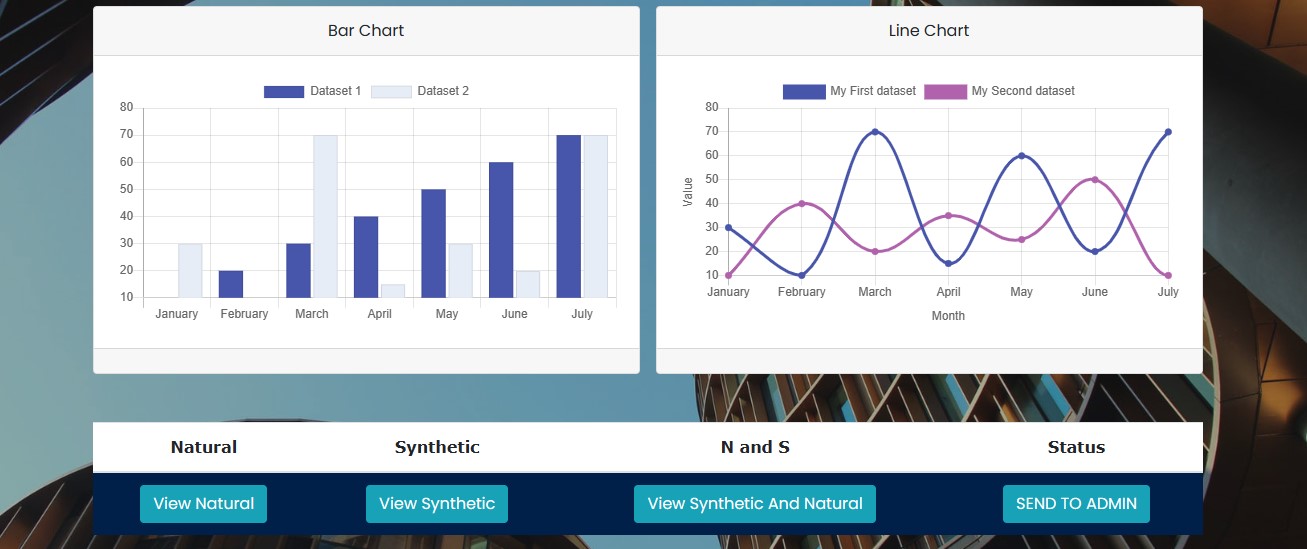
**A4. ADMIN REPORT**

****

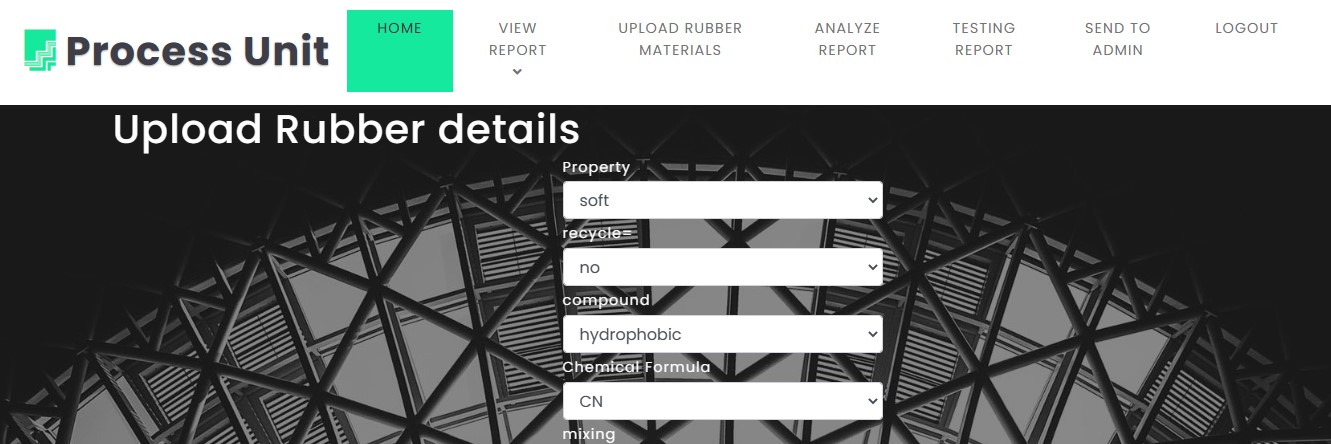
**A5. TESTING REPORT**

****

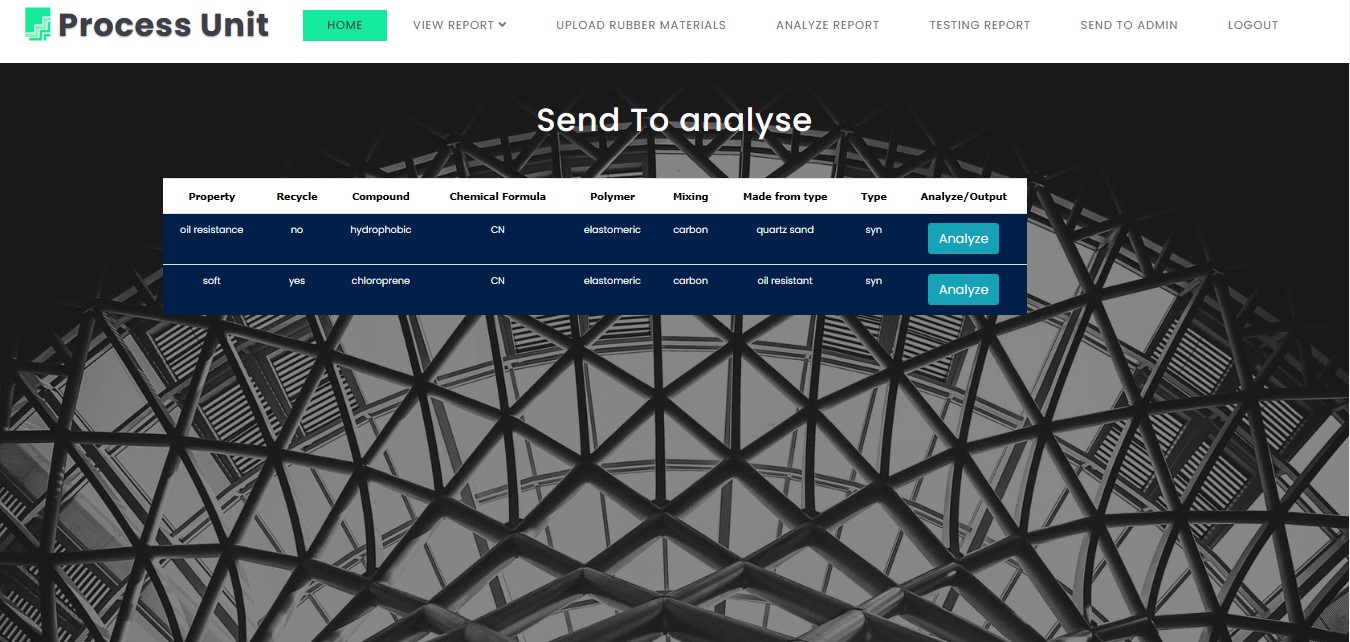
**A6. VENDOR**

****

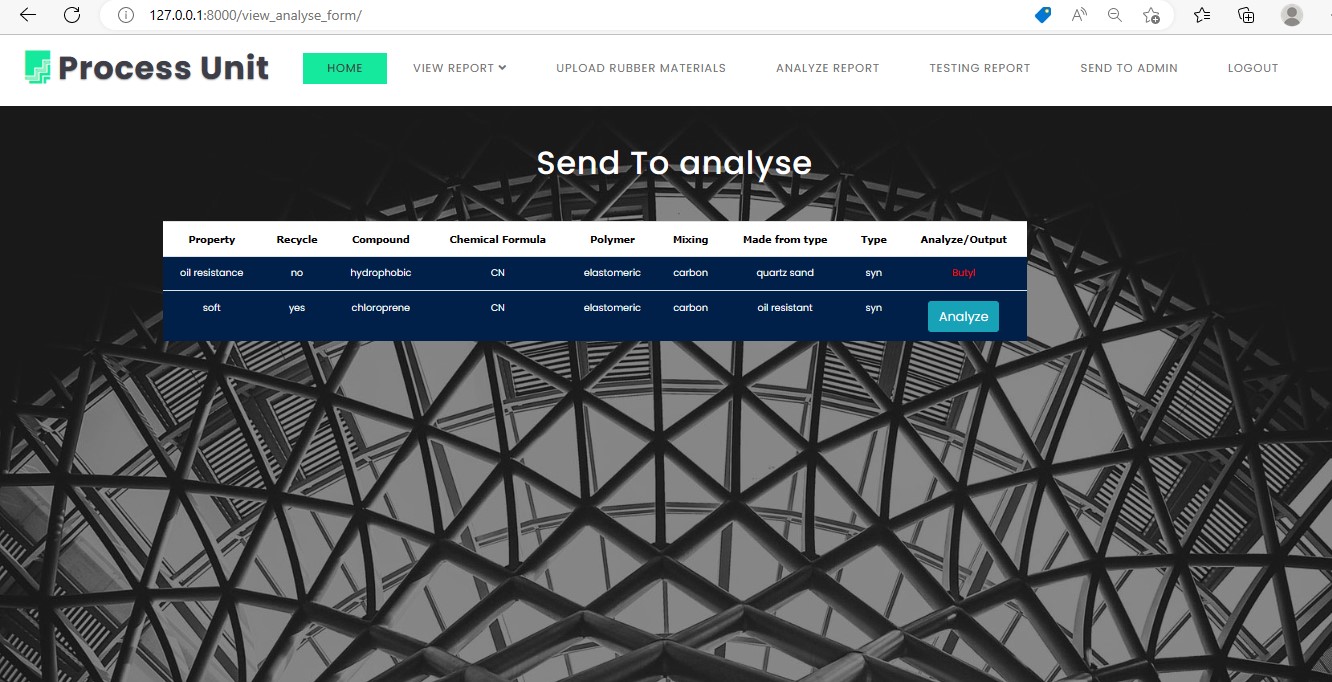
**A7. GRAPH**

****

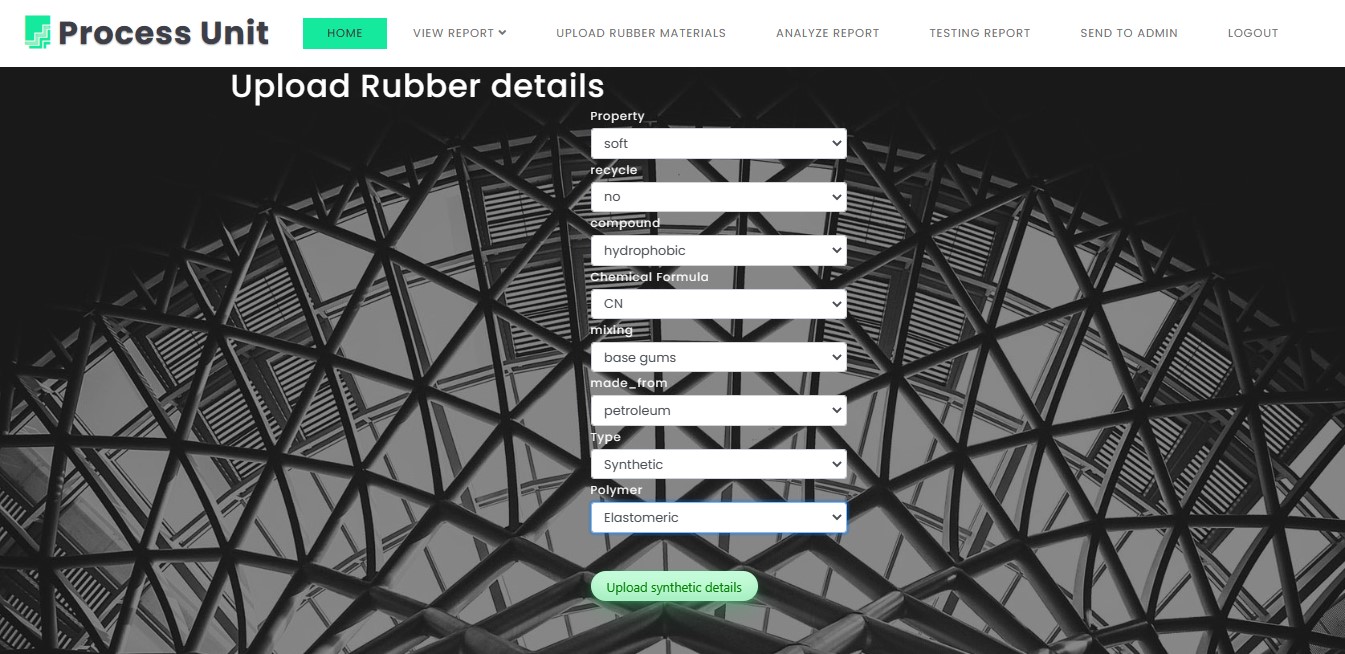
**A8. PROCESS UNIT**

****

**A9. PROCESS UNIT- ANALYSE**

****

**A10. PROCESS UNIT- ANALYSE REPORT**

****

**A11. PROCESS UNIT- RUBBER DETAILS**

[2] J. Beel, B. Gipp, S. Langer, and C. Breitinger, “paper recommender systems: a literature survey,” International Journal on Digital Libraries, vol. 17, no. 4, pp. 305–338, 2016

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” Journal of machine Learning research, vol. 3, no. 1, pp. 993–1022, 2003

[4] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles, “Context-aware citation recommendation,” in Proceedings of the 19th international conference on world wide web, 2010, pp. 421–430

[5] W. Huang, S. Kataria, C. Caragea, P. Mitra, C. L. Giles, and L. Rokach, “Recommending citations: translating papers into references,” in Proceedings of the 21st ACM international conference on Information and knowledge management, 2012, pp. 1910–1914.

[6] H. Liu, X. Kong, X. Bai, W. Wang, T. M. Bekele, and F. Xia, “Context-based collaborative filtering for citation recommendation,” IEEE Access, vol. 3, pp. 1695–1703, 2015.

[7] C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, 2011, pp. 448–456.

[8] K. Sugiyama and M.-Y. Kan, “A comprehensive evaluation of scholarly paper recommendation using potential citation papers,” International Journal on Digital Libraries, vol. 16, no. 2, pp. 91–109, 2015

[9] C. Hsu, M. Yeh, and S. Lin, “A general framework for implicit and explicit social recommendation,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 12, pp. 2228–2241, Dec 2018.

[10] L. Guo, X. Cai, F. Hao, D. Mu, C. Fang, and L. Yang, “Exploiting fine-grained co-authorship for personalized citation recommendation,” IEEE Access, vol. 5, pp. 12 714–12 725, 2017.

[11] W. Wang, B. Xu, J. Liu, Z. Cui, S. Yu, X. Kong, and F. Xia, “Csteller: forecasting scientific collaboration sustainability based on extreme gradient boosting,” World Wide Web, pp. 1–22, 2019

[12] D. Zhang, J. Yin, X. Zhu, and C. Zhang, “Network representation learning: A survey,” arXiv preprint arXiv:1801.05852, 2017.

[13] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, “Heterogeneous information network embedding for recommendation,” IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 2, pp. 357–370, 2019.

[14] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 855–864.

[15] S. Ganguly and V. Pudi, “Paper2vec: combining graph and text information for scientific paper representation,” in European Conference on Information Retrieval, 2017, pp. 383–395.

[16] T. Dai, T. Gao, L. Zhu, X. Cai, and S. Pan, “Low-rank and sparse matrix factorization for scientific paper recommendation in heterogeneous network,” IEEE Access, vol. 6, pp. 59 015–59 030, 2018.

[17] L. Yang, Y. Zheng, X. Cai, H. Dai, D. Mu, L. Guo, and T. Dai, “A lstm based model for personalized context-aware citation recommendation,” IEEE Access, vol. 6, pp. 59 618–59 627, 2018.

[18] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “Arnetminer: extraction and mining of academic social networks,” in Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, 2008, pp. 990–998.

[19] K. Sugiyama and M.-Y. Kan, “Exploiting potential citation papers in scholarly paper recommendation,” in Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries, 2013, pp. 153–162.

[20] X. Ren, J. Liu, X. Yu, U. Khandelwal, Q. Gu, L. Wang, and J. Han, “Cluscite: Effective citation recommendation by information network-based clustering,” in Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, 2014, pp. 821–830.

[21] M. Gori and A. Pucci, “Research paper recommender systems: A random-walk based approach,” in IEEE/WIC/ACM International Conference on Web Intelligence, 2006, pp. 778–781.

[22] F. Meng, D. Gao, W. Li, X. Sun, and Y. Hou, “A unified graph model for personalized query-oriented reference paper recommendation,” in Proceedings of the 22nd ACM international Conference on Information and Knowledge Management, 2013, pp. 1509–1512.

[23] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl, “On the recommending of citations for research papers,” in Proceedings of the 2002 ACM conference on Computer supported cooperative work, 2002, pp. 116– 125.

[24] P. Goyal and E. Ferrara, “Graph embedding techniques, applications, and performance: A survey,” Knowledge-Based Systems, vol. 151, pp. 78–94, 2018.

[25] H. Cai, V. W. Zheng, and K. C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” IEEE Transactions on Knowledge and Data Engineering, vol. 30, no. 9, pp. 1616–1637, Sept 2018.