**A PROJECT REPORTED ON**

**DYNAMIC DISPERSAL MODEL OF TRAFFICKING MACHINE FAILURES LIFETIME IN COPPER MANUFACTURING INDUSTRY**

**ABSTRACT:**

The copper manufacturing industry faces common failure or error in n number of machines simultaneously. However, to maintain production, it is important to monitor the real-time errors or failure of machines simultaneously. In this project we propose some method to get the failure data with some regression, then we analytically check or estimate the failure data’s with efficient computing. We developed a visualization platform to display and understand the statistical data of real-time failures of the machine in the copper production industry. Our result indicates the failure occurred by which machine and to know the type of failure in that machine. After analyzing the failure of the machine and its type, then it helps to change the fails occurred machine to a proper output giving machine. Once the changing of the machine is done then the analytics for that machine are also monitored for plotting the statistical chart to display the performance of the changed machine. It is used to consume the time for error finding, that if the failure occurs we can’t identify the failure within a short that where it occurs. But using this technology we can reduce the failure finding time to increase the efficiency of the manufacturing of the copper materials.

**CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION TO PROJECT
   2. PURPOSE OF THE PROJECT
2. **SYSTEM ANALYSIS**
   1. INTRODUCTION
   2. ANALYSIS MODEL
   3. SDLC PHASES
   4. HARDWARE & SOFTWARE REQUIRMENT
   5. INPUT AND OUTPUT
   6. LIMITATIONS
   7. EXISTING STSTEM
   8. SOLUTION OF THESE PROBLEMS IN PROPOSED SYSTEM
3. **FEASIBILITY REPORT**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMIC FEASIBILITY
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
   1. FUNCTIONAL REQUIREMENTS
   2. NON-FUNCTIONAL REQUIREMENTS
   3. PERFORMANCE REQUIREMENTS
5. **SYSTEM DOVELOPEMENT ENVIRONMENT**
   1. INTRODUCTION TO PYTHON
   2. API TERMINOLOGY
   3. HTML,JAVASCRIPT
   4. FRAMEWORKS
6. **SYSTEM DESIGN**
   1. INTRODUCTION
   2. NORMALIZATION
   3. SYSTEM ARCHITECTURE
   4. E-R DIAGRAM
   5. DFD SYMBOLS
   6. USE CASE DIAGRAM
   7. SEQUENCE DIAGRAM

6.8. CLASS DIAGRAM

1. **OUTPUT SCREENS**
2. **CODING**
3. **SYSTEM TESTING AND IMPLEMENTATION** 
   1. INTRODUCTION
   2. STRATEGIC APPROACH OF SOFTWARE TESTING
   3. UNIT TESTING
4. **SYSTEM SECURITY**

10.1. INTRODUCTION

* 1. SECURITY IN SOFTWARE

1. **CONCLUSION& FUTURE ENHANCEMENT**

**CHAPTER-1**

**1.1 INTRODUCTION TO PROJECT**

## An industry plays an important role in economic development of the country. The favorable factors present in the country are the reasons for the development of many industries. Since these industries provide a larger contribution to gross nation product, productive capacity of people, increasing employment opportunities and as well as exporting products to other countries, they play a crucial role in support and development of economy. For any industry, the continuous flow production is the best process because it eliminates unnecessary steps, reduces processing time and generally manufactures products faster and easier to maintain quality as the product will be the same throughout as it running continuously. Since these industries making their machines to run continuously it is hard in maintaining and cleaning. And this will affect the product quality when not monitored. Find the fault one particular machine is impossible as it will lead to the entire shutdown of the industry. So to resolve this, identifying the fault machine on the running time is the good solution. And hence in this project we are indentifying the faulted machine without causing any interruptions to the other sectors.

## 1.2 PURPOSE OF THE SYSTEM

* Finding the fault with in an efficient time.
* Swift action in reallocation of respective machines.
* Increased productivity.
* Graphical view of monitoring system which is interactive to find malfunction.
* To retain customer.

**CHAPTER-2**

## 

**SYSTEM ANALYSIS**

**2.1 INTRODUCTION**

A regression is made for the usefulness of data collection, data analysis and data integration. In the larger areas such as industries which are generating tons of data will get the help of regression to every element of data operations like data labeling, segmenting and analyzing. The fusion of regression with massive data is a never-ending loop knowing the interest of the customer is the most critical elements for a profitable business. Regression analyze the market and helps business organizations to understand their target customers. The massive data allows industries to calculate the probability of different outcomes and decisions. Predictive analysis helps then by providing suggestion for manufacturing. The input of regression is the information extracted for massive data. Here by making a regression of finding the defective machine in the enormous amount of machines, makes the industry advances in their business by providing a good quality product.

**2.2 ANALYSIS**

**SOFTWARE DEVELOPMENT LIFE CYCLE**

**INTRODUCTION:**

The System Development Lifecycle framework is designed to outline a complete development and implementation process suitable for developing complex applications. SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

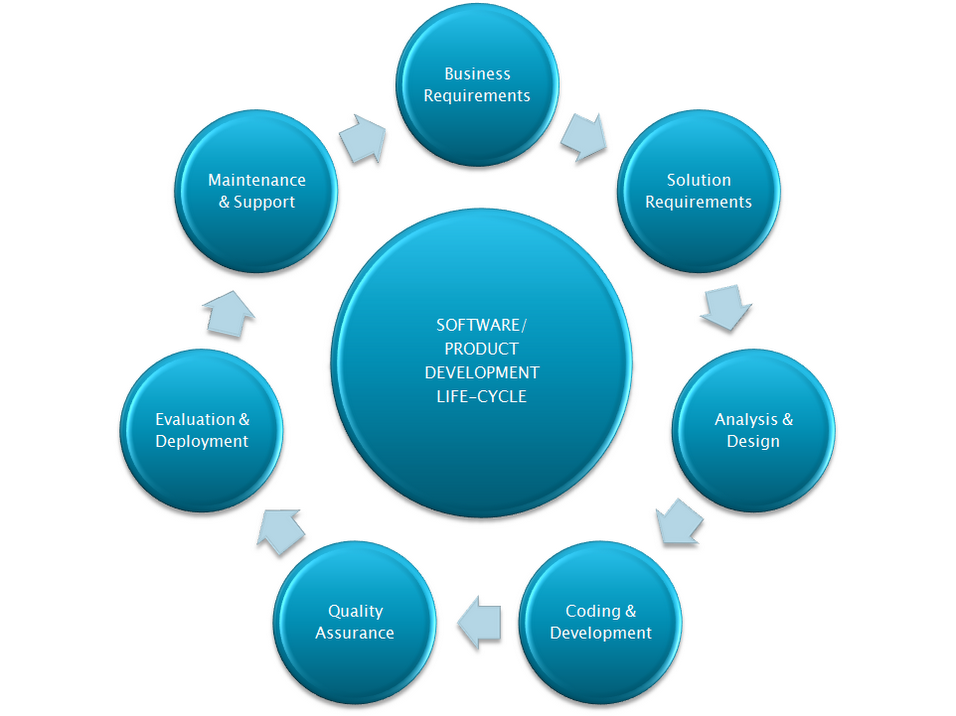
* Business – legislation regulatory requirements, policy, SOP’s, guidelines etc.
* Process – how the business is implemented
* Data – the core business data elements collected for the business
* Application – the gate to the business collecting
* Infrastructure- the servers, network, workstations, etc.

**2.3 SDLC Phases:**

**Stage 1: Scheduling and Requisite investigation:**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational, and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

****

## Stage 2: Significant necessities:

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through .SRS. . Software Requirement Specification document which consists of all the product requirements to be designed and developed during the project life cycle.

## Stage 3: Scheming the product design:

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in the SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

## Stage 4: Structure or Mounting the Product:

## In this stage of SDLC the actual development starts and the product are built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers have to follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers etc.are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java, and PHP are used for coding.

## Stage 5: Testing the Product:

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product, where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

## Stage 6: Consumption in the Market and Safeguarding:

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometime product deployment happens in stages as per the organizations. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

The product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**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; 1 GB Recommended | Minimum 20 GB |

**Software Requirements:**

* **Front end :** Core Python, CSS, JS
* **Web application :** Django, Flask
* **Back end :** MySQL

**OVERVIEW OF SOFTWARE ENGINEERING:**

**Software**is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product**.** Engineering on the other hand, is all about developing products, using well-defined, scientific principles and methods.

**Software engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

## Definitions

IEEE defines software engineering as:

(1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.

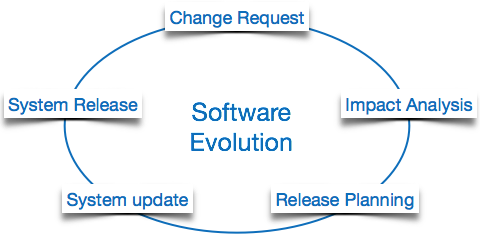
(2) The study of approaches as in the above statement.

Fritz Bauer, a German computer scientist, defines software engineering as:

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and work efficiently on real machines.

## Software Evolution

The process of developing a software product using software engineering principles and methods is referred to as software evolution. This includes the initial development of software and its maintenance and updates, till desired software product is developed, which satisfies the expected requirements.



Evolution starts from the requirement gathering process. After which developers create a prototype of the intended software and show it to the users to get their feedback at the early stage of software product development. The users suggest changes, on which several consecutive updates and maintenance keep on changing too. This process changes to the original software, till the desired software is accomplished.

Even after the user has desired software in hand, the advancing technology and the changing requirements force the software product to change accordingly. Re-creating software from scratch and to go one-on-one with requirement is not feasible. The only feasible and economical solution is to update the existing software so that it matches the latest requirements.

## Software Evolution Laws

Lehman has given laws for software evolution. He divided the software into three different categories:

* **S-type (static-type) -**This is a software, which works strictly according to defined specifications and solutions. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all. For example, calculator program for mathematical computation.
* **P-type (practical-type) -**This is a software with a collection of procedures.This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly. For example, gaming software.
* **E-type (embedded-type) -**This software works closely as the requirement of real-world environment. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations. For example, Online trading software.

## E-Type software evolution

Lehman has given eight laws for E-Type software evolution -

* **Continuing change -**An E-type software system must continue to adapt to the real world changes, else it becomes progressively less useful.
* **Increasing complexity -**As an E-type software system evolves, its complexity tends to increase unless work is done to maintain or reduce it.
* **Conservation of familiarity -**The familiarity with the software or the knowledge about how it was developed, why was it developed in that particular manner etc. must be retained at any cost, to implement the changes in the system.
* **Continuing growth-**In order for an E-type system intended to resolve some business problem, its size of implementing the changes grows according to the lifestyle changes of the business.
* **Reducing quality -**An E-type software system declines in quality unless rigorously maintained and adapted to a changing operational environment.
* **Feedback systems-**The E-type software systems constitute multi-loop, multi-level feedback systems and must be treated as such to be successfully modified or improved.
* **Self-regulation -**E-type system evolution processes are self-regulating with the distribution of product and process measures close to normal.
* **Organizational stability -**The average effective global activity rate in an evolving E-type system is invariant over the lifetime of the product.

## Software Paradigms

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in the software engineering these paradigms stand. These can be combined into various categories, though each of them is contained in one another:

Programming paradigm is a subset of Software design paradigm which is further a subset of Software development paradigm.

### Software Development Paradigm

This Paradigm is known as software engineering paradigms where all the engineering concepts pertaining to the development of software are applied. It includes various researches and requirement gathering which helps the software product to build. It consists of –

* Requirement gathering
* Software design
* Programming

### Software Design Paradigm

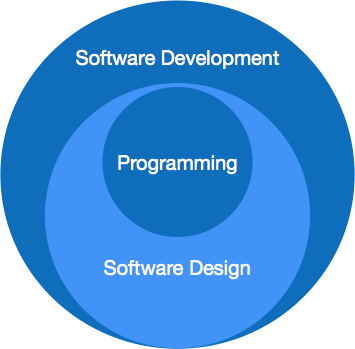
This paradigm is a part of Software Development and includes –

* Design
* Maintenance
* Programming

### Programming Paradigm

This paradigm is related closely to programming aspect of software development. This includes

* Coding
* Testing
* Integration



## Need of Software Engineering

The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.

* **Large software -**It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
* **Scalability-**If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one.
* **Cost-**As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
* **Dynamic Nature-**The always growing and adapting nature of software hugely depends upon the environment in which user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
* **Quality Management-**Better process of software development provides better and quality software product.

## Characteristics of good software

A software product can be judged by what it offers and how well it can be used. This software must satisfy on the following grounds:

* Operational
* Transitional
* Maintenance

Well-engineered and crafted software is expected to have the following characteristics:

### Operational

This tells us how well software works in operations. It can be measured on:

* Budget
* Usability
* Efficiency
* Correctness
* Functionality
* Dependability
* Security
* Safety

### Transitional

This aspect is important when the software is moved from one platform to another:

* Portability
* Interoperability
* Reusability
* Adaptability

### Maintenance

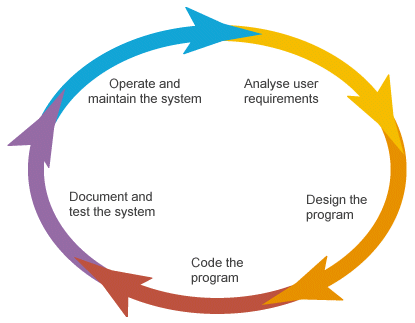
This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:

* Modularity
* Maintainability
* Flexibility
* Scalability

In short, Software engineering is a branch of computer science, which uses well-defined engineering concepts required to produce efficient, durable, scalable, in-budget and on-time software products.

**SOFTWARE DEVELOPMENT LIFE CYCLE**

The Software Development Life Cycle is a process that ensures good software is built.  Each phase in the life cycle has its own process and deliverables that feed into the next phase.  There are typically 5 phases starting with the analysis and requirements gathering and ending with the implementation.  Let’s look in greater detail at each phase:



**Stage 1: Scheduling and Requisite Analysis**

During the discovery phase our team conducts a detailed requirement analysis and creates a work-breakdown structure.

**Stage 2: Scheming the product design**

We identify the design and architecture of the project. SRS is the reference for product architects to come out with the best architecture for the product to be developed.

## Stage 3: Structure or Mounting the Product

## In this stage of SDLC the actual development starts and the product is built. Different high level programming languages such as C, C++, Pascal, Java, C# and PHP are used for coding.

## Stage 4: Testing the Product

Testing is the last phase of the Software Development Life Cycle before the software is delivered to customers. During testing, experienced testers start to test the system against the requirements.

**Stage 5: Consumption in the Market and Safeguarding**

Once the product has been fully tested and no high priority issuesremain in the software, it is time to deploy to production where customers can use the system.

**2.5 INPUT AND OUTPUT**

The major inputs and outputs and major functions of the system are follows:

**Input:**

* The employee must create the account for login. All the employee details have been stored the data in our database.
* The employee gets their corresponding machines with prior training by the data trainer.
* The production team updates the error to admin to resolve the issue on production side.
* Customer purchasing the product approach the industry and provide the requirements.

**Output:**

* The production team which is monitoring the machine performance will note the error raised by the machines in the graphical view.
* Reallocating the machine has the same functionality of the faulty machine results in speedy recovery of production efficiency.

**2.5 INPUT DESIGN**

* Input design is a part of overall system design. The main objective during the input design as given below.
* Input States: User can maintain a database in MySQL server or sql server for his/her business requirement.
* Input Media:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to:

* In this section user can give the input for storage location and get the output from admin side.

**2.6 LIMITATIONS**

* Existing system is not cost effective while compared to the proposed system.
* Output is not visualizing the error to mitigate the future error.
* The huge distributed computing system
* More Regression is used which is not a good idea by industrial standards.
* The changed machine can again give failure data

**2.7 PROBLEMS IN EXISTING SYSTEM:**

In the manufacturing industry of the previous system, Big data concept is used for identifying the machines which have failure occurred frequently. In industries, simultaneously more machines can have failure or error. For detecting the failures that occurred in machines simultaneously previous system is used. In this, they estimate the density of the failure analytically through an efficient computing process. And also developed a visualization platform to extract useful statistical information that is a real-time failure. In that system, there is no error name found but we can find the name of the machine where the error occurs. From the statistical view, we can understand the machine which had an error. And to find error the complex Gaussian hypergeometric distribution and classical KDE approach can perform best if the overﬁtting problem can be avoided and the complexity burden is overtaken.

**2.8 PROPOSED SYSTEM**

The copper manufacturing project moves with the big data analysis which is a new technology that deals with a large amount of data and also the speed of the data transferred at a time and with this that the data can be structured, semi-structured or unstructured that in other terminology they use to say volume, velocity or variety. In this volume means if a failure occurs in multiple machines then the failure details are of large data. If machine failure occurs simultaneously the at the same time large data will flow to the console at a certain speed which indicates the velocity concept in big data and also if there is an error then there can be transferring data in the form of structured, semi-structured, or unstructured data depending upon the requirement of the manufacturing industry which these indicates the variety in big data.

**Advantage of Proposed System:**

* Our proposed model Reduced Human Error exponentially
* Trustworthy Distributed Systems comparing with other systems
* Increases Accuracy when compared to existing model
* Improved Time Efficiency and Improved Understandable Data

**CHAPTER-3**

**FEASIBILITY REPORT**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**CHAPTER-4**

**SOFTWARE REQUIREMENT SPECIFICATION**

**4.1 INTRODUCTION**

The purpose of this document is to present a detailed description of the Web application system. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the Regional Historical Society for its approval.

**PURPOSE**

The purpose of this Software Requirement Specification (SRS) is to help the project. It is provided with some requirements which are used in the Transaction Mercator System. All parts; design, coding and testing will be prepared with helping of SRS. The purpose of this document is to detail the requirements placed on the Transaction Mercator System and serves as a contract between the customer and the developers as to what is to be expected of the stock exchange, and how the components of the system are working with each other with external systems.

This document will be checked by the group member’s supervisor and it will corrected by members if supervisor orders.

**DEVELOPERS RESPONSIBILITIES OVERVIEW:**

The developer is responsible for:

* Developing the system, which meets the SRS and solving all the requirements of the system?
* Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
* Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
* Conducting any user training that might be needed for using the system.
* Maintaining the system for a period of one year after installation.
  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.

**4.2 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

**4.3 PERFORMANCE REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the required specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**CHAPTER-5**

**SYSTEM DEVELOPEMENT ENVIRONMENT**

**5.1 INTRODUCTION TO PYTHON**

**About 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 [backported](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 expeditedas 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).

**Key points:**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-oriented way or a functional way.

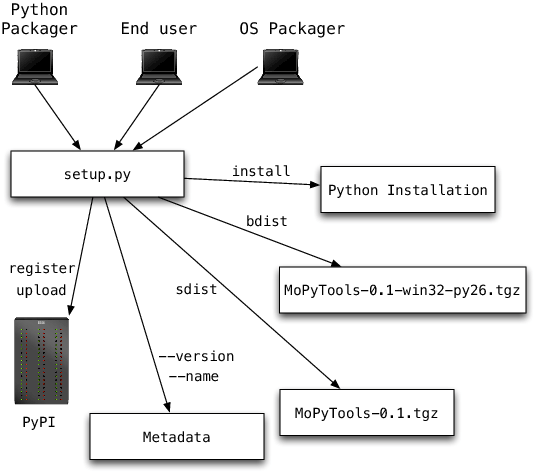
**Python Syntax compared to other programming languages**

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

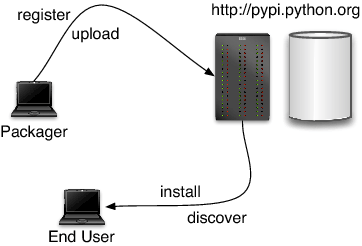
**Importance of python to the Internet**

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 modelling 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.

**Design of python:**

****

**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 bytecode compiler responsible for producing Python code objects from the source code of a user application;
* a [bytecode 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 *bytecode compiler* is the pre-processing phase that produces a compact bytecode format via a chain of flexible passes (tokenizer, lexer, parser, abstract syntax tree builder, bytecode generator). The *bytecode evaluator* interprets this bytecode. 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 bytecode 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 behaviours of the Python objects.

**Simple:**

Python was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer. Learning Python will oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

**Object oriented**

Python was not designed to be source-code compatible with any other language. This allowed the Python team the freedom to design with a blank state. One outcome of this was a clean, usable, pragmatic approach to objects. The object model in Python is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

**Robust**

The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs.Was given a high priority in the design of Python. Python is strictly typed language; it checks your code at compile time and runtime.

Python virtually eliminates the problems of memory management and deal location, which is completely automatic. In a well-written Python program, all run-time errors can and should be managed by your program.

## 5.2 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.

Copy the below code into your text editor. As before, we’ll examine the code more closely once you have it running.

**import**flask

**from**flask**import** request, jsonify

**import**sqlite3

app**=** flask.Flask(\_\_name\_\_)

app.config["DEBUG"] **=**True

**defdict\_factory**(cursor, row):

d **=** {}

**for** idx, col **in**enumerate(cursor.description):

d[col[0]] **=** row[idx]

**return** d

**@**app.route('/', methods**=**['GET'])

**defhome**():

**return**'''<h1>Distant Reading Archive</h1>

<p>A prototype API for distant reading of science fiction novels.</p>'''

**@**app.route('/api/v1/resources/books/all', methods**=**['GET'])

**defapi\_all**():

conn**=** sqlite3.connect('books.db')

conn.row\_factory **=** dict\_factory

cur**=** conn.cursor()

all\_books **=**cur.execute('SELECT \* FROM books;').fetchall()

**return** jsonify(all\_books)

**@**app.errorhandler(404)

**defpage\_not\_found**(e):

**return**"<h1>404</h1><p>The resource could not be found.</p>", 404

**@**app.route('/api/v1/resources/books', methods**=**['GET'])

**defapi\_filter**():

query\_parameters **=** request.args

id**=** query\_parameters.get('id')

published**=** query\_parameters.get('published')

author**=** query\_parameters.get('author')

query**=**"SELECT \* FROM books WHERE"

to\_filter **=** []

**if**id:

query**+=**' id=? AND'

to\_filter.append(id)

**if** published:

query**+=**' published=? AND'

to\_filter.append(published)

**if** author:

query**+=**' author=? AND'

to\_filter.append(author)

**ifnot** (id**or** published **or** author):

**return** page\_not\_found(404)

query**=** query[:**-**4] **+**';'

conn**=** sqlite3.connect('books.db')

conn.row\_factory **=** dict\_factory

cur**=** conn.cursor()

results**=** cur.execute(query, to\_filter).fetchall()

**return** jsonify(results)

app.run()

Save the code as api\_final.py in your api folder and run it by navigating to your project folder in the terminal and entering the command:

python api\_final.py

Note that if a previous version of the code is still running, you will first need to end that process by pressing Control-C before executing the new code. Once this example is running, try out the filtering functionality with these HTTP requests:

**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.

**Hidden Form Fields:**

One way to support anonymous session tracking is to use hidden from the fields. As the name implies, these are fields added to an HTML, from that are not displayed in the client’s browser, they are sent back to the server when the form that contains them is submitted.

In a sense, hidden form fields define constant variables for a form. For a servlet receiving a submitted form, there is no difference between a hidden field and a visible filed.

As more information is associated with a client’s session. It can become burdensome to pass it all using hidden form fields. In these situations, it’s possible to pass on just a unique session ID that identifies as a particular client session.

That session ID can be associated with complete information about its session that is stored on the server.

The advantage of hidden form fields is their ubiquity and support for anonymity. Hidden fields are supported in all the popular browsers, they demand on special server requirements, and they can be used with clients that haven’t registered or logged in.

The major disadvantage with this technique, however, is that works only for a sequence of dynamically generated forms, the technique breaks down immediately with static documents, emailed documents bookmarked documents and browser shutdowns.

**URL Rewriting:**

URL rewriting is another way to support anonymous session tracking, With URL rewriting every local URL the user might click on is dynamically modified. Or rewritten, to include extra, information. The extra information can be in the form of extra path information, added parameters, or some custom, server-specific.URL change. Due to the limited space available in rewriting a URL, the extra information is usually limited to a unique session.

Each rewriting technique has its own advantage and disadvantage

Using extra path information works on all servers, and it works as a target for forms that use both the Get and Post methods. It does not work well if the servlet has to use the extra path information as true path information

The advantages and disadvantages of URL. Rewriting closely match those of hidden form fields, The major difference is that URL rewriting works for all dynamically created documents, such as the Help servlet, not just forms. With the right server support, custom URL rewriting can even work for static documents.

**Persistent Cookies:**

A fourth technique to perform session tracking involves persistent cookies. A cookie is a bit of information. Sent by a web server to a browser that can later be read back form that browser. When a browser receives a cookie, it saves the cookie and there after sending the cookie back to the server each time it accesses a page on that server, subject to certain rules. Because a cookie’s value can uniquely identify a client, cookies are often used for session tracking.

Persistent cookies offer an elegant, efficient, easy way to implement session tracking. Cookies provide as automatic an introduction for each request, as we could hope for. For each request, a cookie can automatically provide a client’s session ID or perhaps a list of clients' performance. The ability to customize cookies gives them extra power and versatility.

The biggest problem with cookies is that browsers don’t always accept cookies sometimes this is because the browser doesn’t support cookies. More often it’s because the browser doesn’t support cookies. More often it’s because the user has specifically configured the browser to refuse cookies.

## Languages influenced by Python

## Python's design and philosophy have influenced many other programming languages:

* [Boo](https://en.wikipedia.org/wiki/Boo_(programming_language)) uses indentation, a similar syntax, and a similar object model.
* [Cobra](https://en.wikipedia.org/wiki/Cobra_(programming_language)) uses indentation and a similar syntax, and its *Acknowledgements* document lists Python first among languages that influenced it.
* [Coffee Script](https://en.wikipedia.org/wiki/CoffeeScript), a [programming language](https://en.wikipedia.org/wiki/Programming_language) that cross-compiles to JavaScript, has Python-inspired syntax.
* [ECMAScript](https://en.wikipedia.org/wiki/ECMAScript)/[JavaScript](https://en.wikipedia.org/wiki/JavaScript) borrowed [iterators](https://en.wikipedia.org/wiki/Iterator) and [generators](https://en.wikipedia.org/wiki/Generator_(computer_science)) from Python.
* [GD Script](https://en.wikipedia.org/wiki/GDScript), a scripting language very similar to Python, built-in to the [Godot](https://en.wikipedia.org/wiki/Godot_(game_engine)) game engine.
* [Go](https://en.wikipedia.org/wiki/Go_(programming_language)) is designed for the "speed of working in a dynamic language like Python"and shares the same syntax for slicing arrays.
* [Groovy](https://en.wikipedia.org/wiki/Groovy_(programming_language)) was motivated by the desire to bring the Python design philosophy to [Java](https://en.wikipedia.org/wiki/Java_(programming_language)).
* [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) was designed to be "as usable for general programming as Python".
* [Nim](https://en.wikipedia.org/wiki/Nim_(programming_language)) uses indentation and similar syntax.
* [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language))'s creator, [Yukihiro Matsumoto](https://en.wikipedia.org/wiki/Yukihiro_Matsumoto), has said: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language.
* [Swift](https://en.wikipedia.org/wiki/Swift_(programming_language)), a programming language developed by Apple, has some Python-inspired syntax.

Python's development practices have also been emulated by other languages. For example, the practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python, a PEP) is also used in [Tcl](https://en.wikipedia.org/wiki/Tcl), [Erlang](https://en.wikipedia.org/wiki/Erlang_(programming_language)),,  and Swift.

**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")

**Flask:**

**Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework) because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

The microframework Flask is based on the Pocoo projects, Werkzeug and Jinja2.

**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.

**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)).

**5.3 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.

**Basic Html Tags**:

<!-- --> Specific Comments.

<A>………</A> Creates Hypertext links.

<B>………</B> Creates hypertext links.

<Big>……..</Big> Formats text in large-font

<Body>……. </Body> contains all tags and text in the Html-document

<Center>……</Center> Creates Text

<DD>………..</DD> Definition of a term.

<TABLE>…… </TABLE> creates a table

<Td>………..</Td> indicates table data in a table.

<Tr>………..</Tr> designates a table row

<Th>………. </Th> creates a heading in a table.

##### **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.

**5.4 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.

**CHAPTER-6**

**6.1 INTRODUCTION**

In our application when a demand is arised from the customer side for the product. The production team views the request of the customer and starts the production process by instructing the employee. The employee once login in to respective pages, the admin allot the corresponding machines to the employee. Before this allotment process, the data trainer will train all machines. When the machine is in the running state, it may intimate some error and the error will be displayed in the form of graphical representation. Alongside, the production team which is already keep tracking of employee details and their machines will also notice the same error. To resolve this error quickly, either employee or the production can update the error details to the admin. The admin forwards the details of error immediately to the data trainer. The data trainer with check for the machine related to the previous machine used by the employee and allocate the same kind of machine to the employee and solve the issue in the production side. Now the main goal of the product team is to manufacture the quantity of products demand by their customer. Once the manufacturing process gets completed the payment process is checked to deliver the products for customer on time. And finally the admin checks the production efficiency to analyze the work and quality of their team.

**Module Description:**

Modules:

1) Customer

2) Employee

3) Data trainer

4) Production

5) Admin

* Customer:

This module gives out the registration process with the customer details of name, email id, dob, password, contact number and address. With this, the customer can login to the customer page. Within customer module there is sub module called update order, purchased material list and Purchase list. After login process in update module customer can start the order process which they give type of material, quantity, need for the period of warranty claim, time to need and quality rate want by the customer. In purchase list, visualization of ordered details by the specific customer takes place. The ordered details are shown up in the form of table. The table gives the type of the material ordered, quantity of the material, need for the period of warranty claim, time to need and quantity. These are the details fetched while ordering. But in addition to Production process and the payment details also displayed. The production process displays whether the production is complete or incomplete with undergoing process details. And the payment column shows the status payment whether paid or not. If its not paid then it shows to pay the amount for the order. In purchased material list sub module displays the table consists of the delivered materials which is ordered by the customer.

* Employee:

This module gives the registration process along with the login process. If the employee has registered after the registration is confirmed by admin they can login. Within the employee module there is another modules of upload, update, requirement and run machine. After a successful login of the employee they can update the additional tools and also additional machineries in upload module which the employee has and along with that employee needs some equipment or tools that they can upload in requirement field. In update module employee can see details of uploaded update machinery or tools of that particular employee. In addition to this in requirement module employee can visualize the requirement update which employee has updated in requirement field. In run module employee name and allocated machine name are displayed if the allocated machine has changed then this page also updated. And also in this page machine running process takes place. It redirect to the graph page which shows the output of the machine. If any error occurs then it shows the error and also option to send the error to admin.

* Data trainer:

This module also has the registration process along with the login access. If registration done then after the approval of admin data trainer can login. Within this module there is train and change machine modules are there. After a login process data trainer job is to train the given machine or the machines which is in the industry. Once the data trainer has trained the machines the all machines are ready to run. If any error has occurred in employee machine then it send to admin and that is sent to the data trainer and that is displayed in this module. If any error detected in this module then the data trainer can change that particular employee’s machine to the machine which is in the production side. If the machine has changed then there will not show any error with that machine.

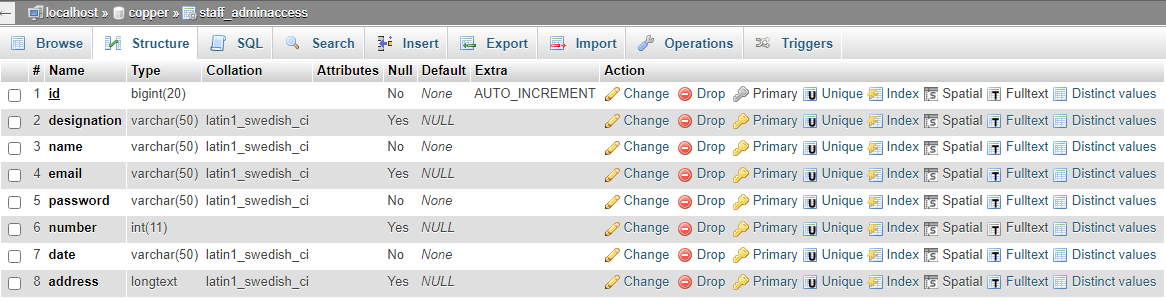
* Production:

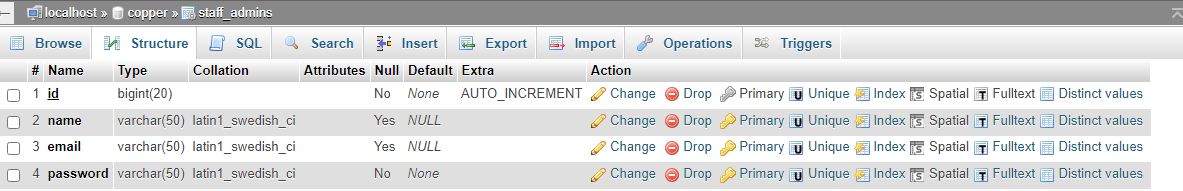
This module also gives the registration process with login access. In this module contains sub module of production, delivery and monitor. The delivery sub module shows the ordered details of all customer where it can have the material type, quantity, quality, time to need. Also in addition with the production details also shown up i.e if customer ordered for some quantity but if its not manufactured then it gives the status of manufacturing. And also in another column it displays the payment details of that particular customer. If customer has paid the amount then it displays paid and also if manufacturing has finished then it allows to deliver that specific product with the specific customer. In production sub module, the displays the produced material name along with the quantity manufactured. The monitor sub module shows the employee name, allocated machine, and status of the allocation. In this production team can monitor the each machine, where that machine runs properly or not.

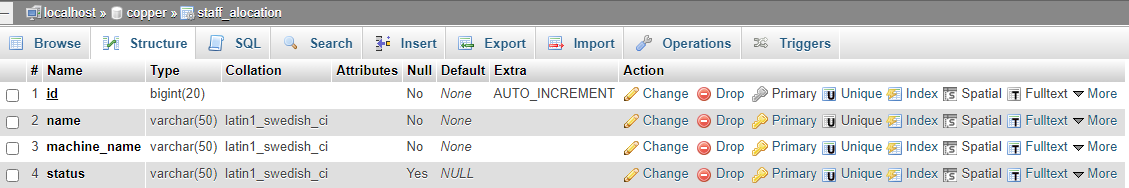
* Admin:

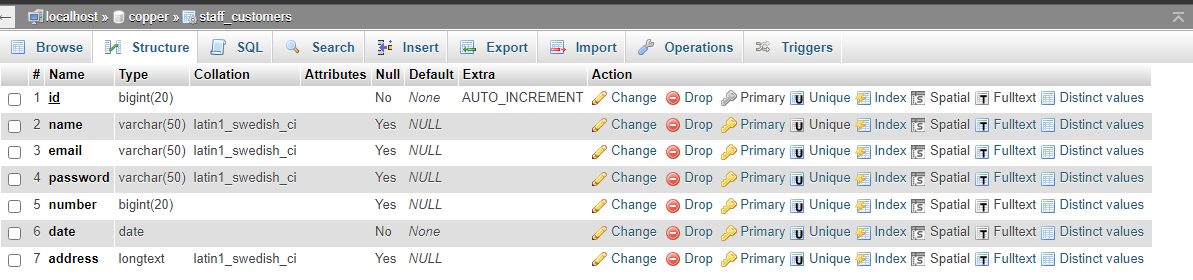
This module gives the login access for the person in admin. Within this module number of modules placed namely register queue, update, requirement, error and allocation. In register queue module if employee and data trainers are registered then that details are sent to this table. Admin work is to verify the registration details and make the particular employee or data trainer to move for the further process. And also in update module it displays the update details of machineries and tools uploaded by the employee with the particular employee name. As previous in requirement module the updated requirement details uploaded by the particular employee are displayed. In error module if the error occurred while running machine by the employee then they can send that particular error to this table which displays the name of the employee had error and also the name of the machine that error occurs and also the type of the error. In allocation module , after accessing the employee registration admin have to allocate the particular machine to specific employee.

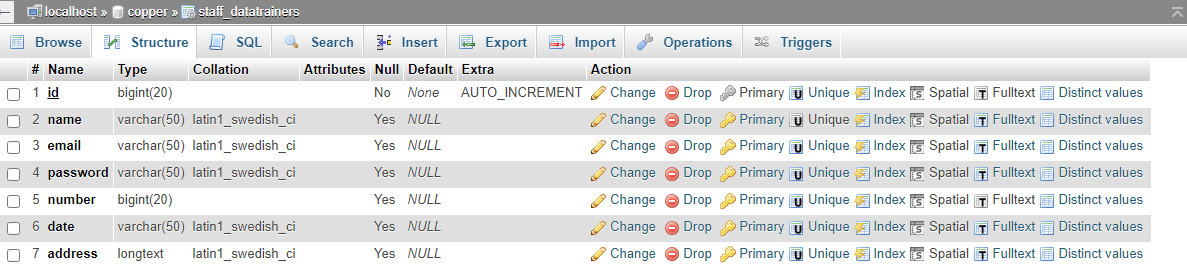
**Database Screen Shot:**

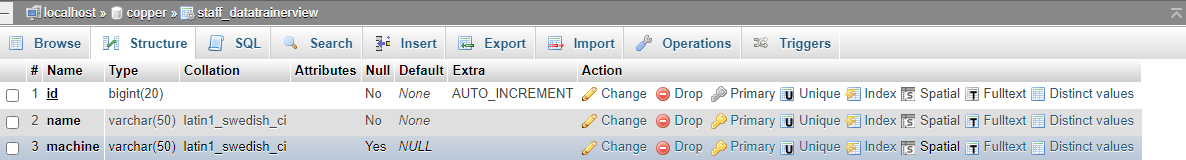


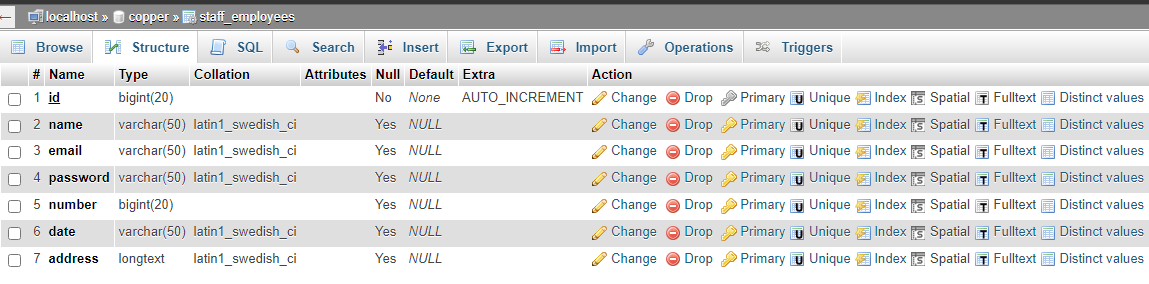


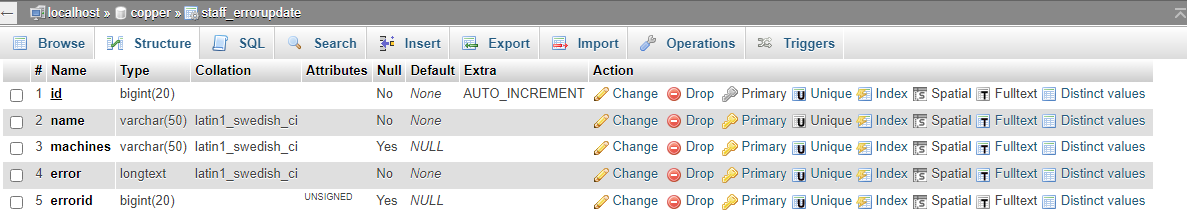


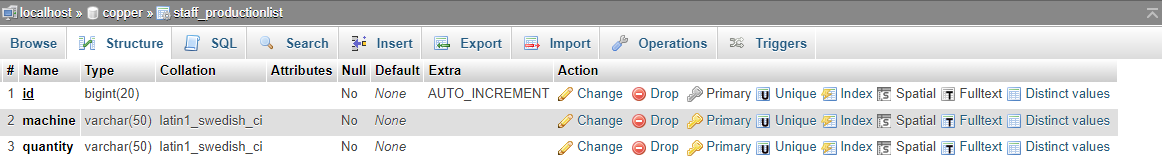


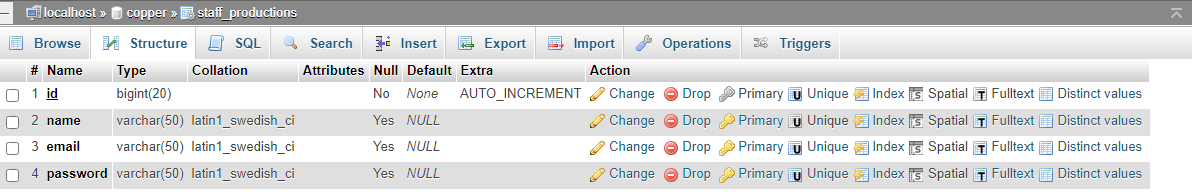


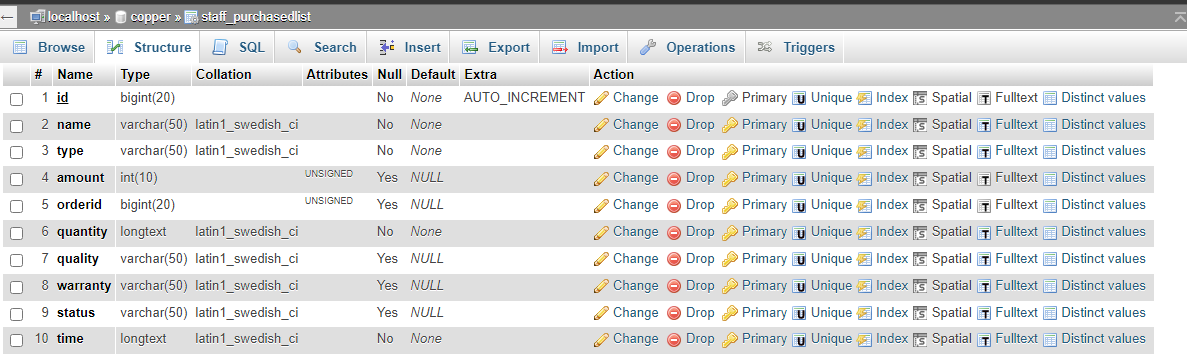


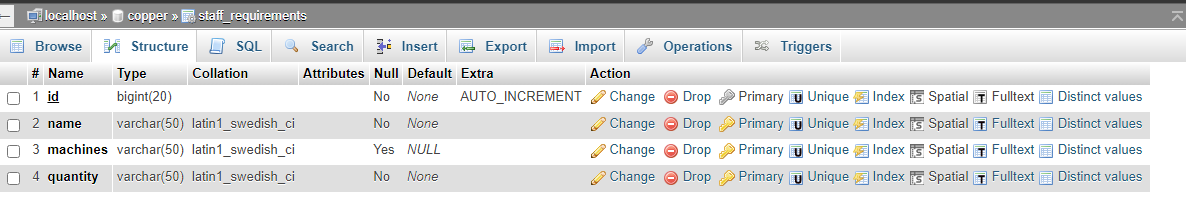


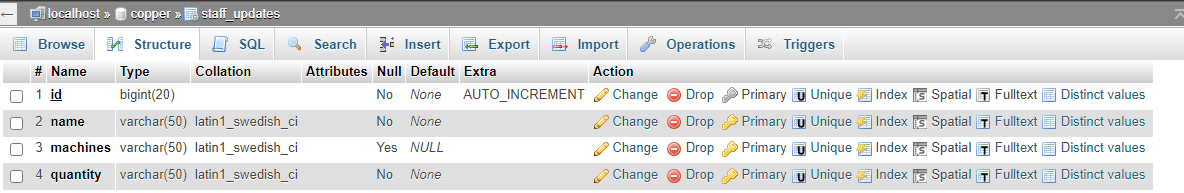




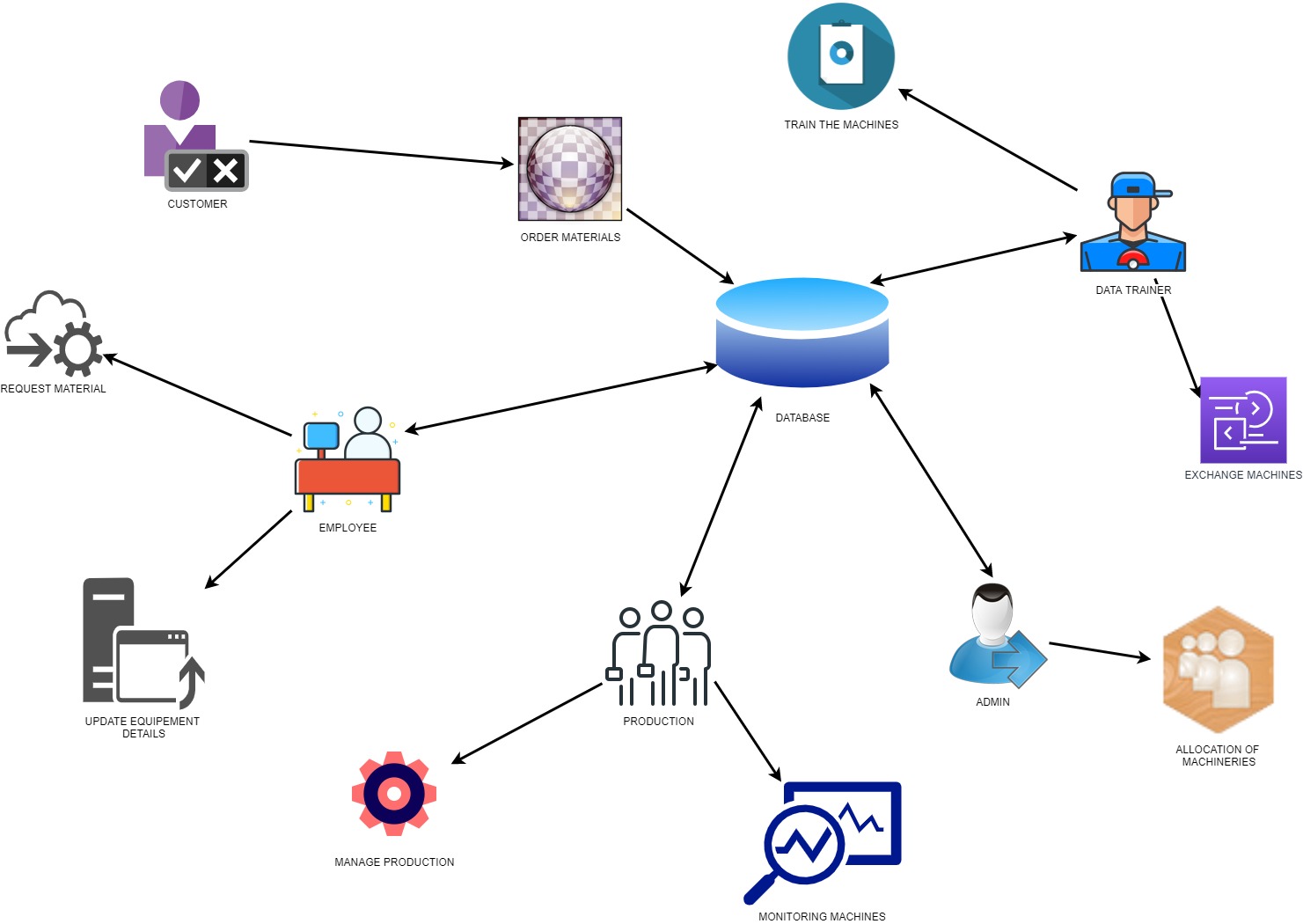








**6.3 System Architecture:**

****

**6.4 E – R DIAGRAMS**

* + The relation upon the system is structured through a conceptual ER-Diagram, which not only specifics the existing entities, but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
  + The Entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct, the date modeling activity the attributes of each data object noted, is the ERD can be described resign a data object description.
  + The set of primary components that are identified by the ERD are
  + Data object
  + Relationships
  + Attributes
  + Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.

**6.5 DFD SYMBOLS**

In the DFD, there are four symbols

1. A square defines a source (originating) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms the incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

A process that transforms the data flow

Source or Destination of data

Data flow

**CONSTRUCTING A DFD:**

Several rules of thumb are used in drawing DFD’S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw the long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

**SAILENT FEATURES OF DFD’S**

1. The DFD shows the flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process, whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

**TYPES OF DATA FLOW DIAGRAMS**

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

**CURRENT PHYSICAL**

In Current Physical DFD process label includes the name of the people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly, data flows and data stores are often labelled with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

**CURRENT LOGICAL:**

The physical aspects in the system are removed as much as possible so that the current system is reduced to its essence to the data and the processes that transform them regardless of actual physical form.

**NEW LOGICAL:**

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system, but had problems with how it was implemented typically through the new logical model will differ from the current logical model while having additional functions, absolute function removal and inefficient flows recognized.

**NEW PHYSICAL:**

The new physical represents only the physical implementation of the new system.

**RULES GOVERNING THE DFD’S**

**PROCESS**

1. No process can have only outputs.
2. No process can have only inputs. If an object has only inputs than it must be a sink.
3. A process has a verb phrase label.

**DATA STORE**

1. Data cannot move directly from one data store to another data store, a process must move data.
2. Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into the data store
3. A data store has a noun phrase label.

**SOURCE OR SINK**

The origin and /or destination of data

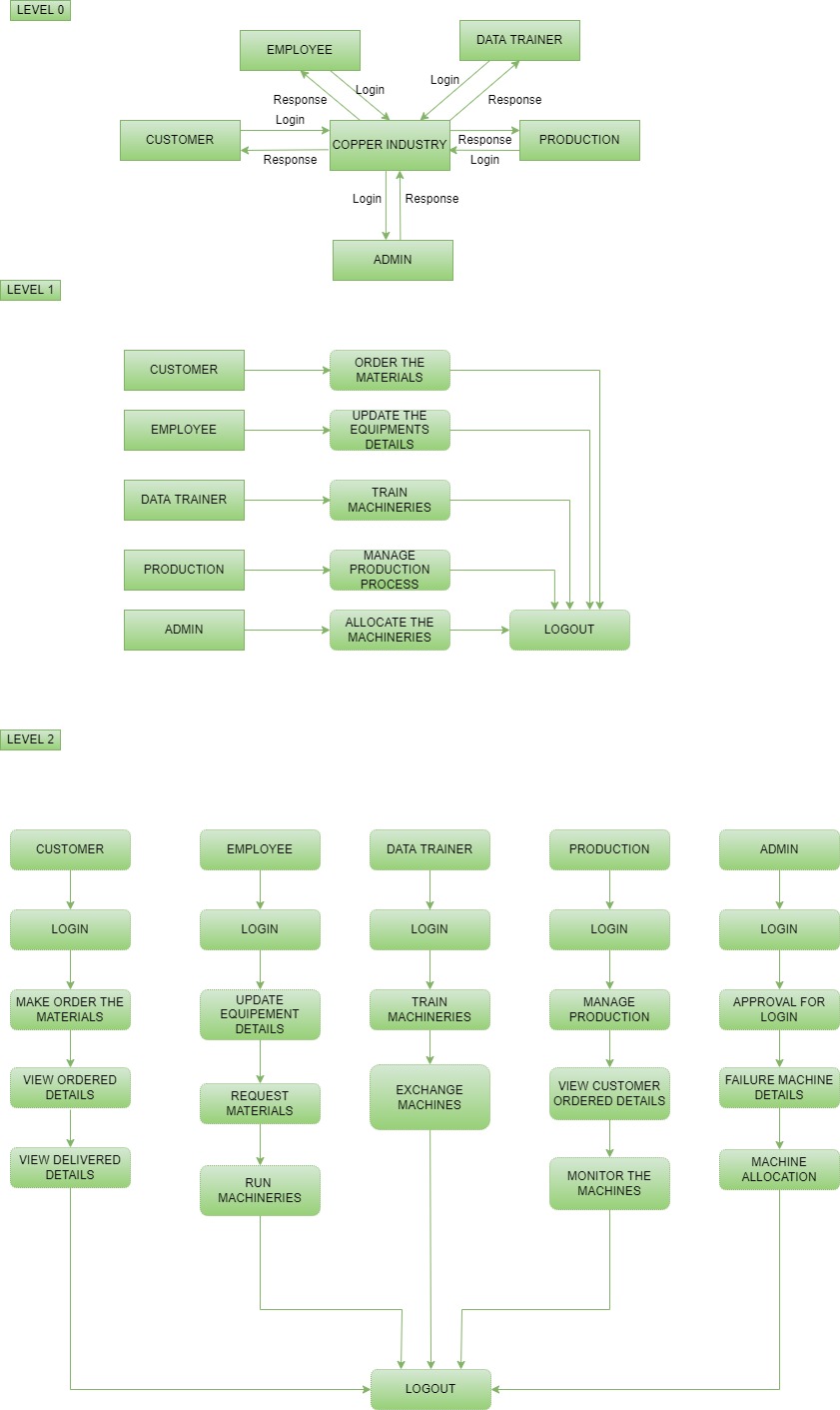
1. Data cannot move direly from a source to sink it must be moved from a process
2. A source and /or sink have a noun phrase land

**DATA FLOW**

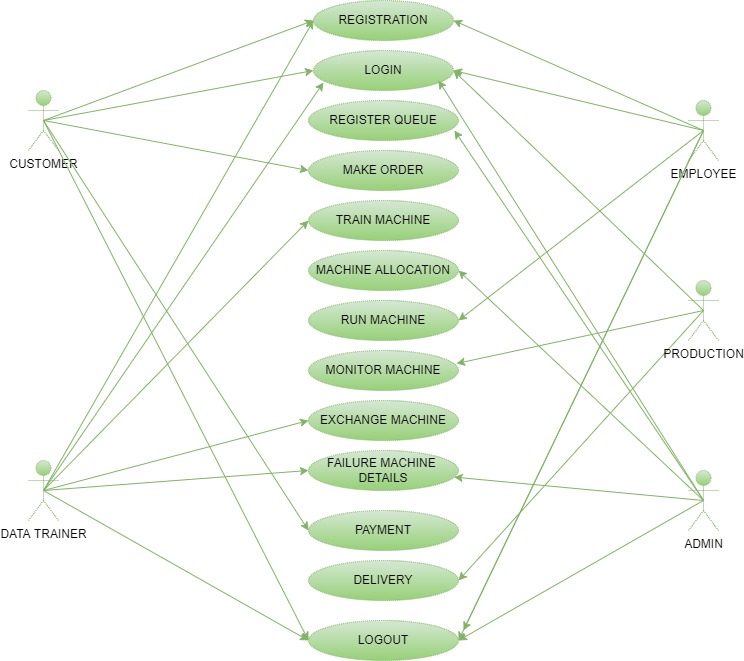
1. A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however by two separate arrows since these happen at different type.
2. A join in DFD means that exactly the 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.

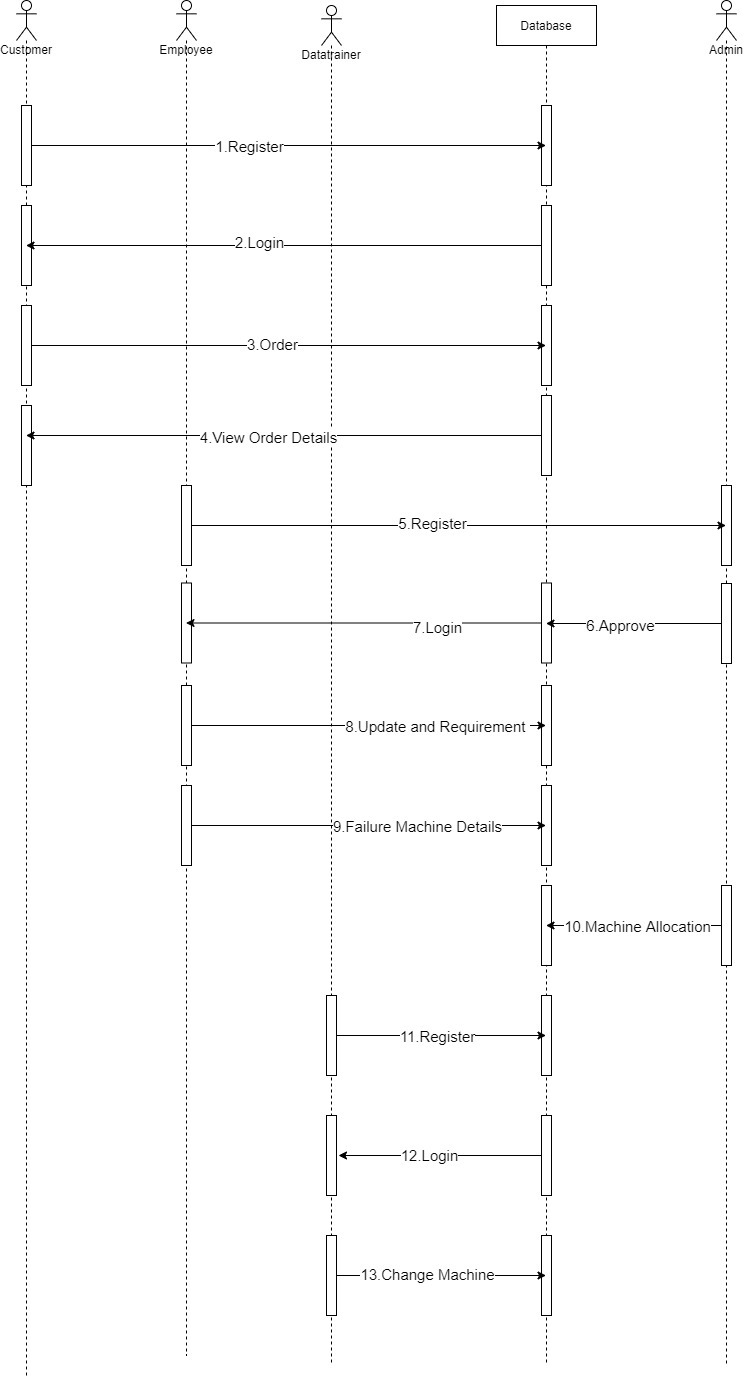
**DATA FLOW DIAGRAMS:**

****

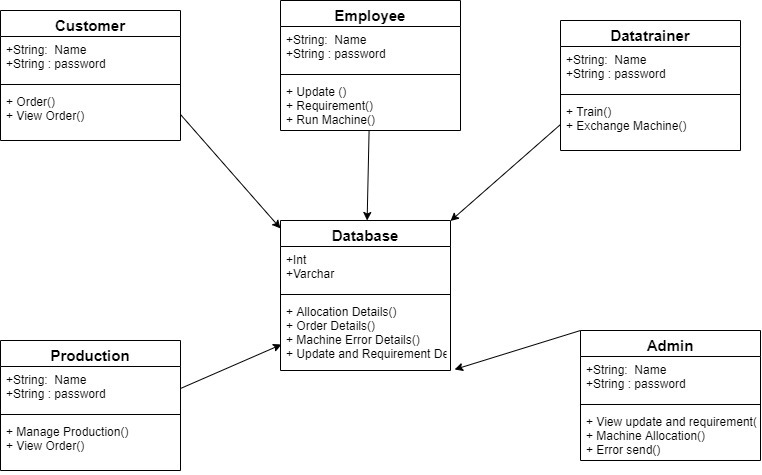
**6.6 USE CASE DIAGRAM**

****

**6.7 SEQUENCE DIAGRAM**

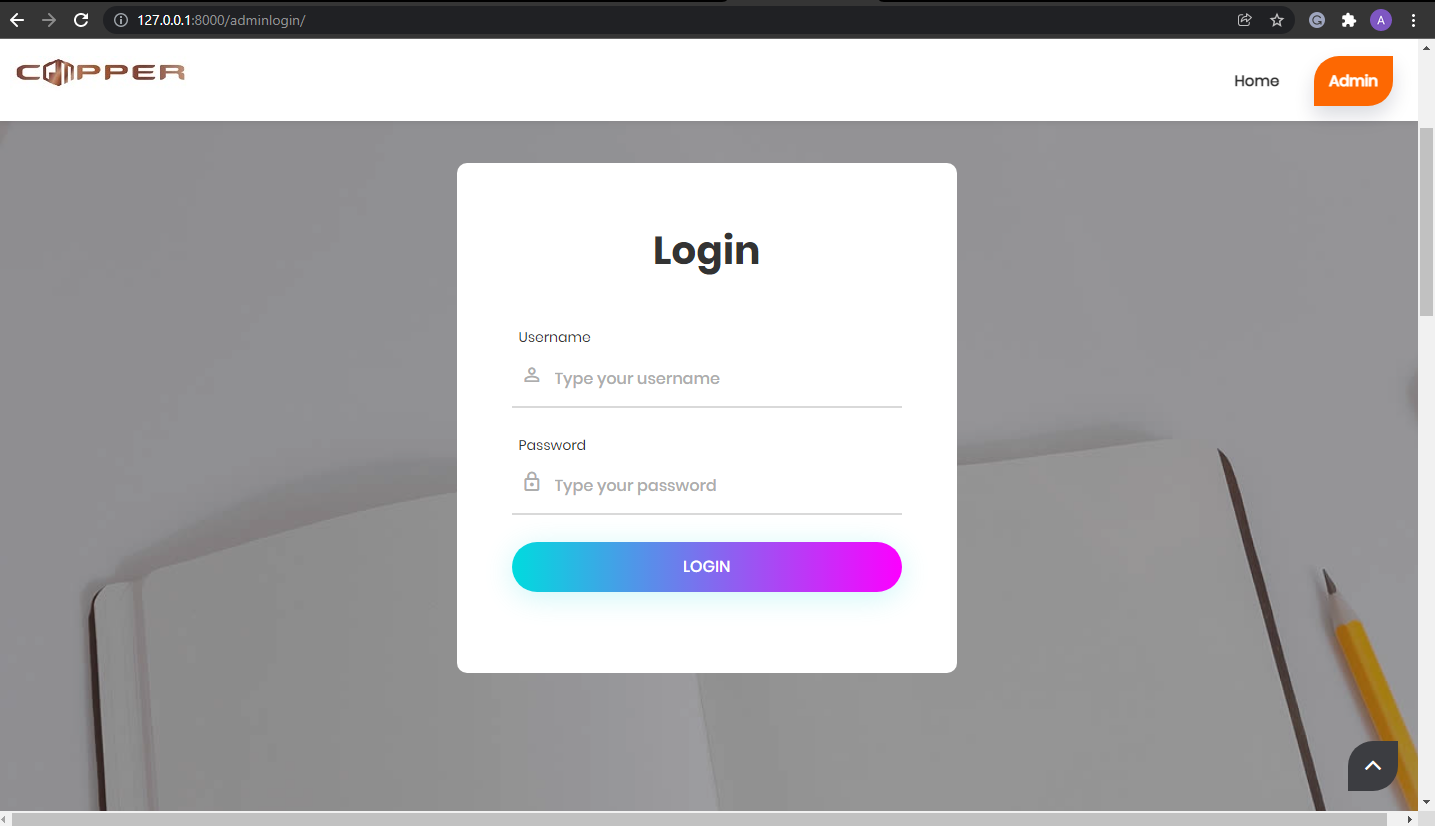
****

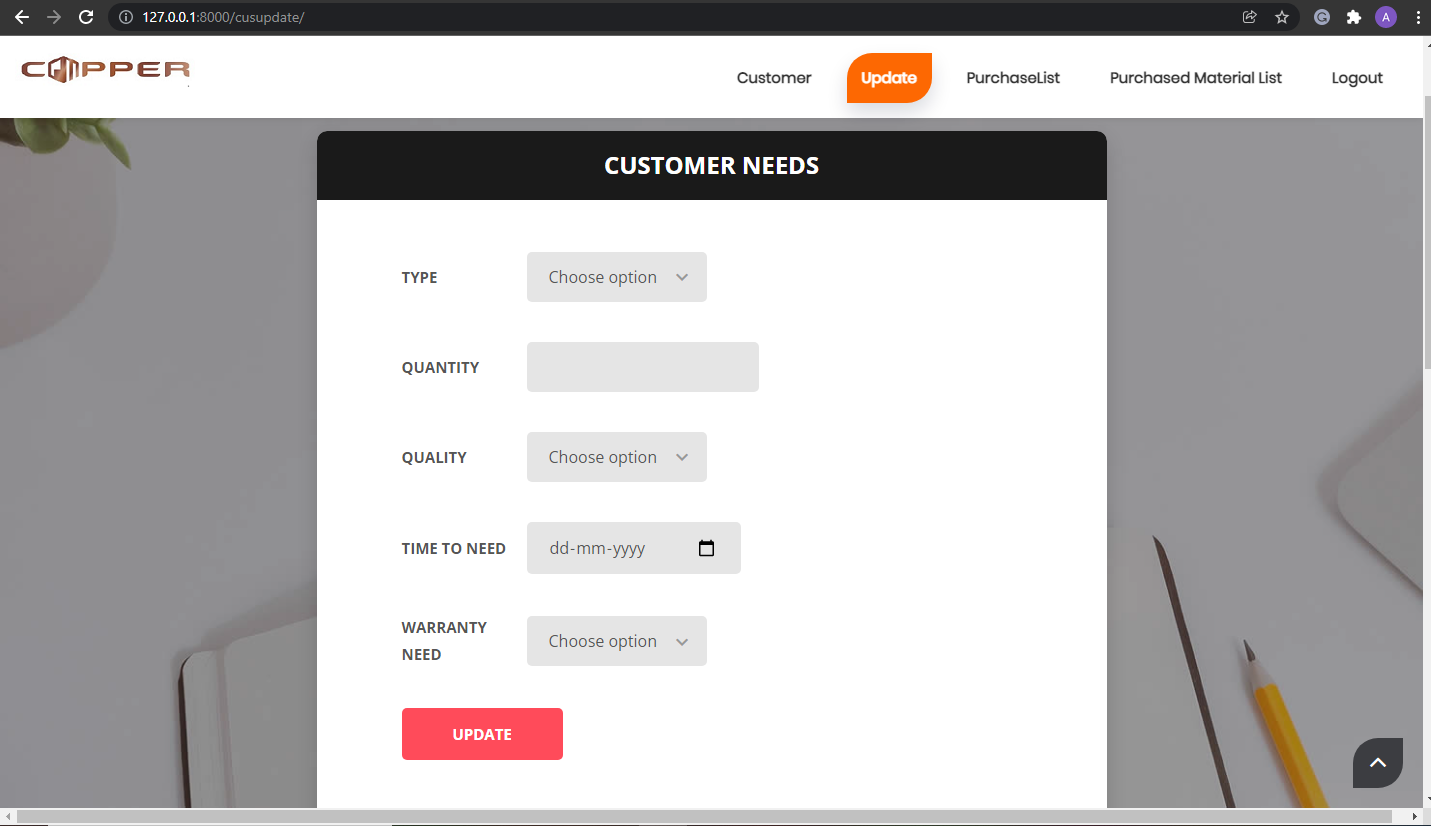
**6.8 CLASS DIAGRAM**

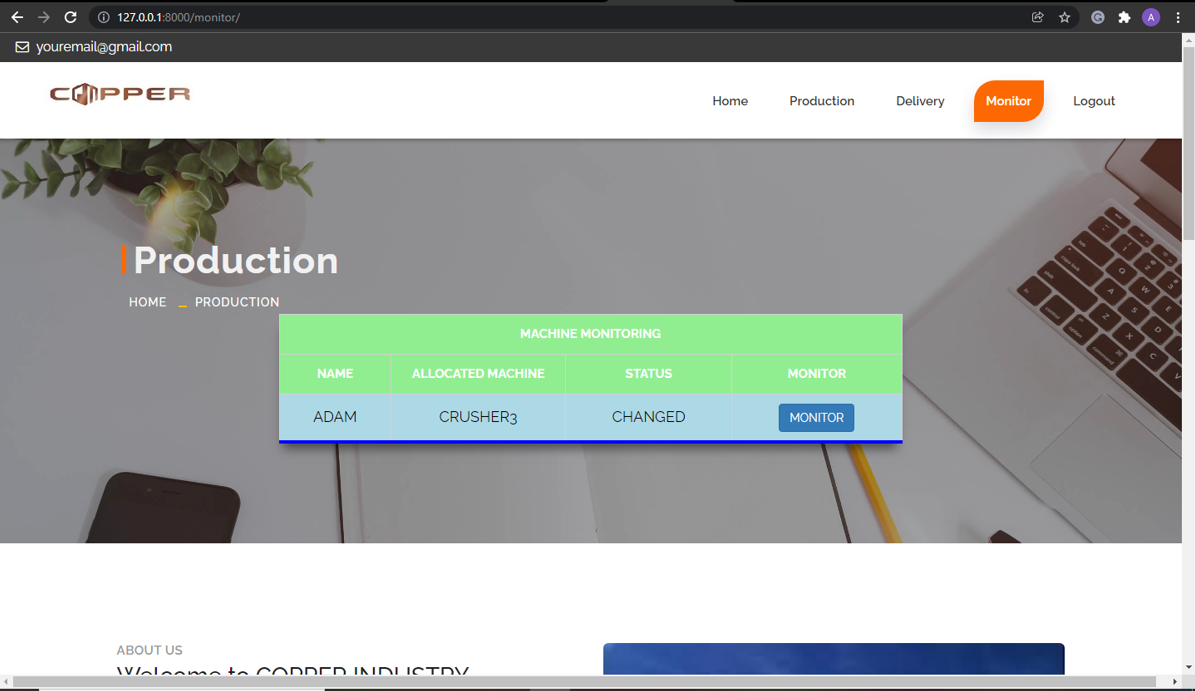
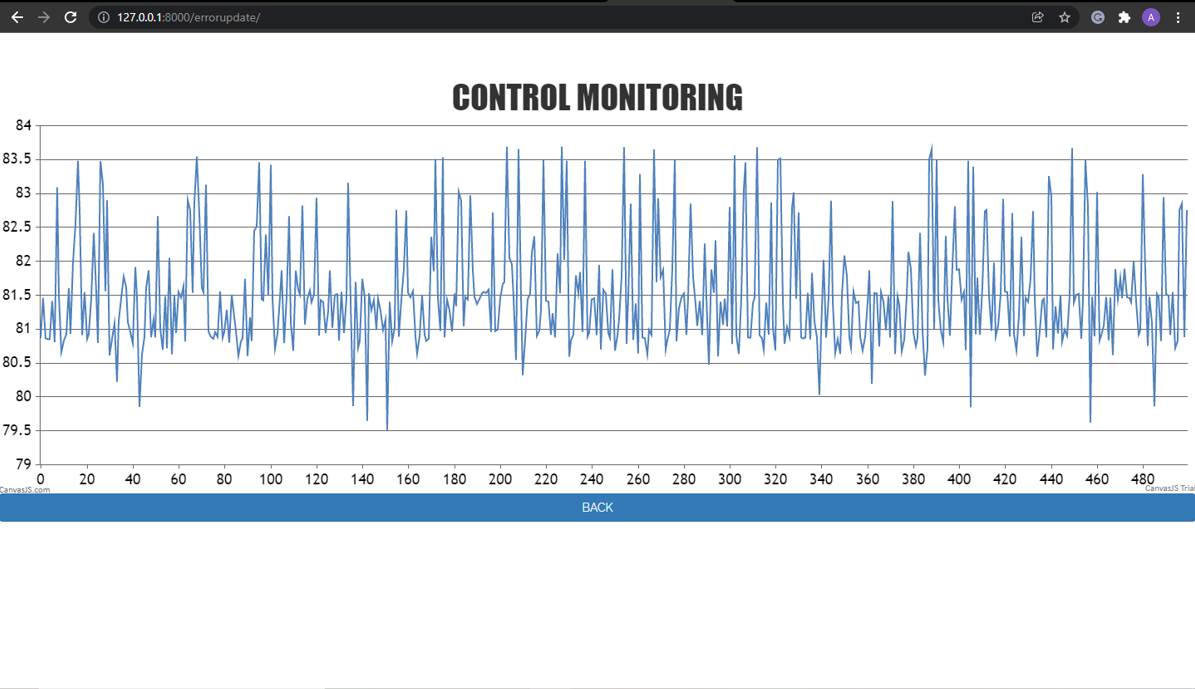
****

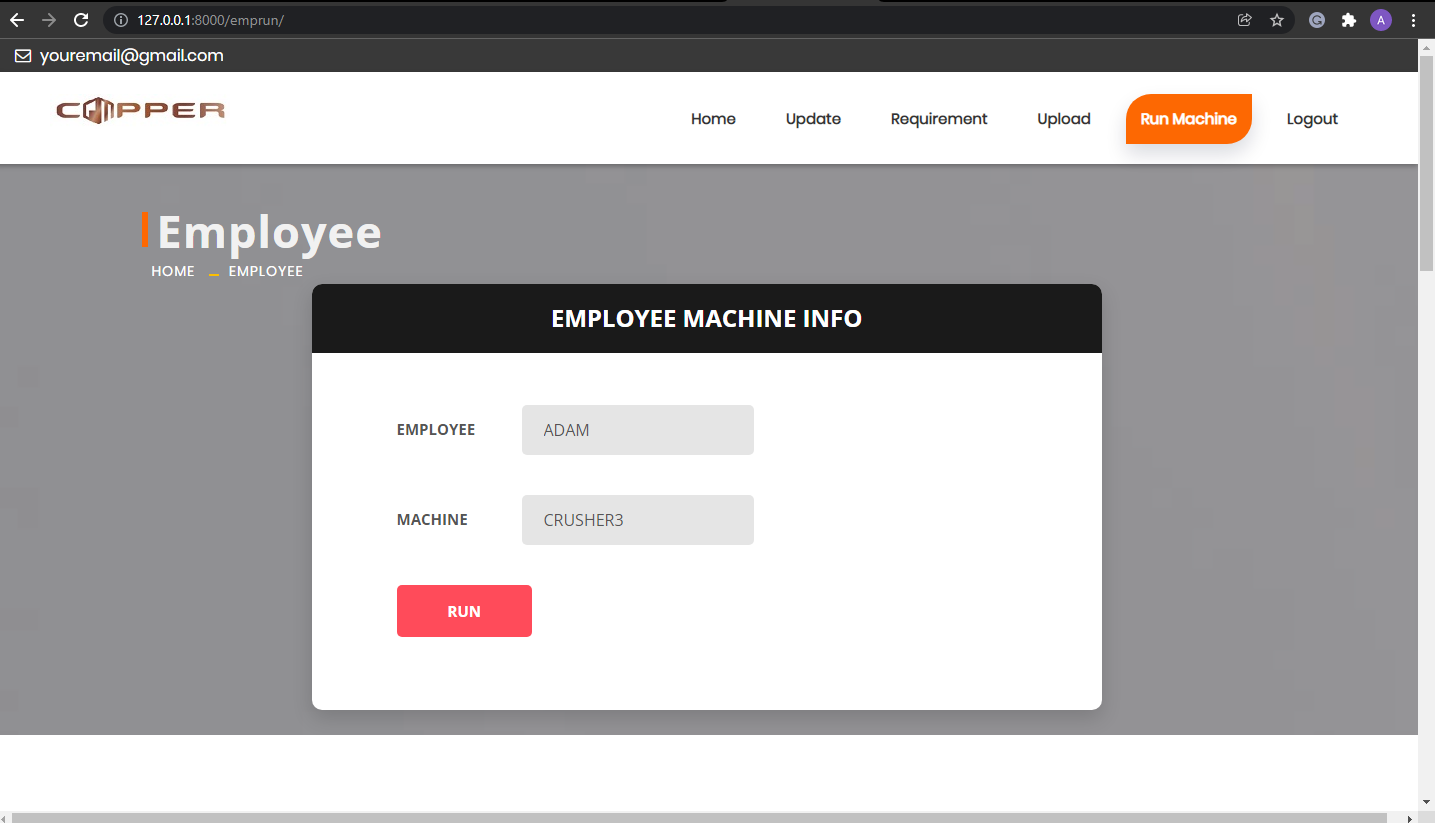
**CHAPTER-7**

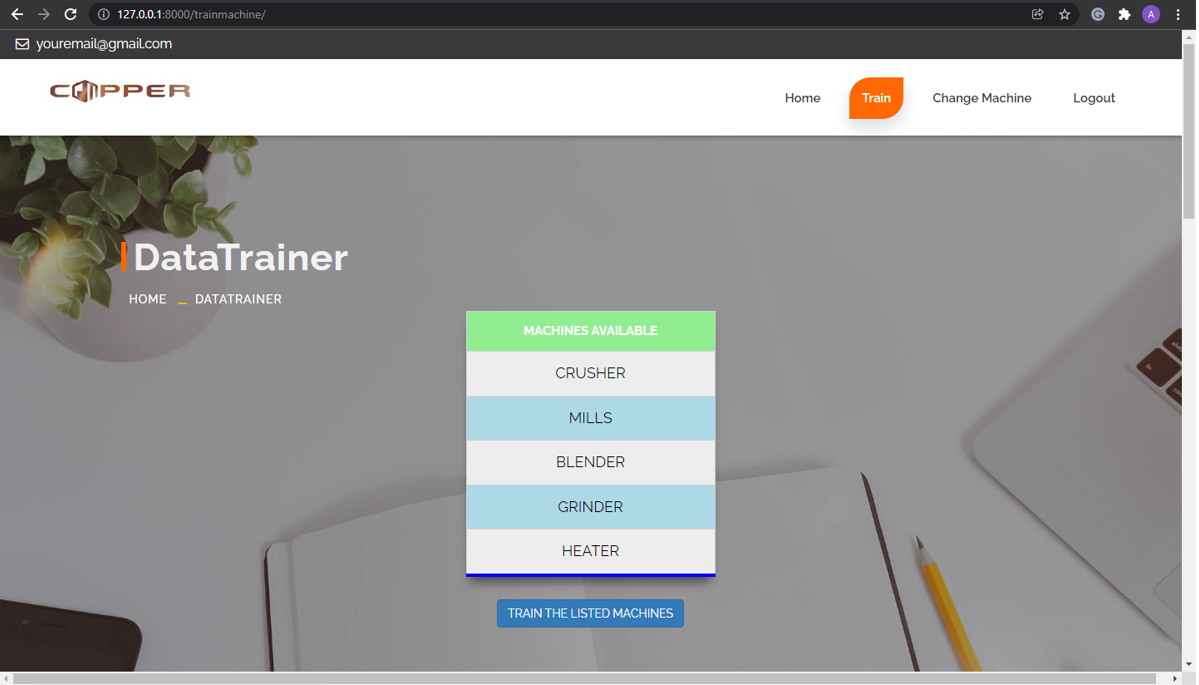
**OUTPUT SCREENS**

****

****

****

****

****

**CODINGS**

**Frontend Codings:**

<title>ADMIN</title>  
 <meta name="keywords" content="">  
 <meta name="description" content="">  
 <meta name="author" content="">  
  
 <!-- Site Icons -->  
 <link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />  
 <link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
 <!-- Site CSS -->  
 <link rel="stylesheet" href="{% static 'style.css'%}">  
 <link rel="stylesheet" href="{% static 'table.css'%}">  
 <!-- Responsive CSS -->  
 <link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
  
</head>  
<body>  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a class="active" href="">Home</a></li>  
 <li><a href="/adminqueue">Register Queue</a></li>  
 <li><a href="/adminupdate">Updates</a></li>  
 <li><a href="/adminreq">Requirements</a></li>  
 <li><a href="/adminerror">Errors</a></li>  
 <li><a href="/alocation">Allocation</a></li>  
 <li><a href="/calculation">Efficiency</a></li>  
 <li><a href="/admin/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h1 align="center" style="color:white">WELCOME TO ADMIN PAGE</h1>  
 <h2>Admin</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Admin</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
<br>  
  
 <div id="about" class="section wb">  
 <div class="container">  
 <div class="row">  
 <div class="col-md-6">  
 <div class="message-box">  
 <h4>About Us</h4>  
 <h2>Welcome to COPPER INDUSTRY</h2>  
  
 <p> The copper industry is a major contributor to the global economy. Copper is more than just a mined metal; it provides jobs and promotes of a higher standard of living. ... The copper industry provides benefits to more than just the companies mining copper and fabricating products from it.</p>  
  
 <a href="#services" class="btn btn-light btn-radius btn-brd grd1">Learn More</a>  
 </div><!-- end messagebox -->  
 </div><!-- end col -->  
  
 <div class="col-md-6">  
 <div class="post-media wow fadeIn">  
 <img src="{% static 'images/copper1.jpg'%}" alt="" class="img-responsive img-rounded">  
 </div><!-- end media -->  
 </div><!-- end col -->  
 </div><!-- end row -->  
  
 <hr class="hr1">  
  
 <div class="row">  
 <div class="col-md-6">  
 <div class="post-media wow fadeIn">  
 <img src="{% static 'images/copper2.jpg'%}" alt="" class="img-responsive img-rounded">  
 </div><!-- end media -->  
 </div><!-- end col -->  
  
 <div class="col-md-6">  
 <div class="message-box">  
 <h4>Who We are</h4>  
 <h2>We Are COPPER MATERIAL PRODUCER</h2>  
  
 <p> The copper industry is a major contributor to the global economy. Copper is more than just a mined metal; it provides jobs and promotes of a higher standard of living. ... The copper industry provides benefits to more than just the companies mining copper and fabricating products from it.</p>  
  
 <a href="#services" class="btn btn-light btn-radius btn-brd grd1">Learn More</a>  
 </div><!-- end messagebox -->  
 </div><!-- end col -->  
 </div><!-- end row -->  
 </div><!-- end container -->  
 </div><!-- end section -->

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon"/>  
<link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
<!-- Bootstrap CSS -->  
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
<!-- Site CSS -->  
<link rel="stylesheet" href="{% static 'style.css'%}">  
<!-- Responsive CSS -->  
<link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
<!-- Custom CSS -->  
<link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
<!-- Modernizer for Portfolio -->  
<script src="{% static 'js/modernizer.js'%}"></script>  
  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'fonts/font-awesome-4.7.0/css/font-awesome.min.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'fonts/iconic/css/material-design-iconic-font.min.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'vendor/animate/animate.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'vendor/css-hamburgers/hamburgers.min.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'vendor/animsition/css/animsition.min.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'vendor/select2/select2.min.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'vendor/daterangepicker/daterangepicker.css'%}">  
<!--===============================================================================================-->  
<link rel="stylesheet" type="text/css" href="{% static 'css/util.css'%}">  
<link rel="stylesheet" type="text/css" href="{% static 'css/main.css'%}">  
  
<!--[if lt IE 9]>  
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
<![endif]-->  
  
</head>  
<body>  
  
<!-- LOADER -->  
<div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
</div><!-- end loader -->  
<!-- END LOADER -->  
  
<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
</div>  
<header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"  
 aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}"  
 alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/">Home</a></li>  
 <li><a class="active" href="/adminlogin">Admin</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
  
<div class="banner-area banner-bg-1">  
 <div class="container">  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Admin</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Admin</a></li>  
 </ul>  
 <div class="limiter">  
 <div class="container-login100" style="background-image: url('images/bg-01.jpg');">  
 <div class="wrap-login100 p-l-55 p-r-55 p-t-65 p-b-54">  
 <form method='post' class="login100-form validate-form">  
 {% csrf\_token %}  
 {% if messages %}  
 {% for message in messages %}  
 <h4 align="center" style="color:red">{{message}}</h4>  
 {% endfor %}  
 {% endif %}  
 <span class="login100-form-title p-b-49">  
 Login  
 </span>  
  
 <div class="wrap-input100 validate-input m-b-23"  
 data-validate="Username is reauired">  
 <span class="label-input100">Username</span>  
 <input class="input100" type="text" name="aname"  
 placeholder="Type your username">  
 <span class="focus-input100" data-symbol="&#xf206;"></span>  
 </div>  
  
 <div class="wrap-input100 validate-input" data-validate="Password is required">  
 <span class="label-input100">Password</span>  
 <input class="input100" type="password" name="apass"  
 placeholder="Type your password">  
 <span class="focus-input100" data-symbol="&#xf190;"></span>  
 </div>  
 <br>  
 <div class="container-login100-form-btn">  
 <div class="wrap-login100-form-btn">  
 <div class="login100-form-bgbtn"></div>  
 <button class="login100-form-btn">  
 Login  
 </button>  
 </div>  
 </div>  
 <br>  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />  
 <link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
 <!-- Site CSS -->  
 <link rel="stylesheet" href="{% static 'style.css'%}">  
 <link rel="stylesheet" href="{% static 'table.css'%}">  
 <!-- Responsive CSS -->  
 <link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
  
</head>  
<body>  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
  
 <div class="phone-box">  
 <a href="tel:1234567890"><i class="fa fa-phone" aria-hidden="true"></i> +1 234 567 890</a>  
 </div>  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/admin1">Home</a></li>  
 <li><a href="/adminqueue">Register Queue</a></li>  
 <li><a class="active" href="">Updates</a></li>  
 <li><a href="/adminreq">Requirements</a></li>  
 <li><a href="/adminerror">Errors</a></li>  
 <li><a href="/alocation">Allocation</a></li>  
 <li><a href="/calculation">Efficiency</a></li>  
 <li><a href="/admin/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Admin</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Admin</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div style="background-image: url('demopic.jpeg');">  
 {% if update %}  
 <div style="overflow-x:auto;">  
 <table align="center">  
 <tr>  
 <th colspan="3">UPDATE</th>  
 </tr>  
 <tr>  
 <th>NAME</th>  
 <th>MACHINE</th>  
 <th>QUANTITY</th>  
 </tr>  
 <tr>  
 {% for i in update%}  
 <td>{{i.name}}</td>  
 <td>{{i.machines}}</td>  
 <td>{{i.quantity}}</td>  
  
 </tr>  
{% endfor %}  
 </table>  
 {% else %}  
 <h1 style="color:orange" align="center">NO UPDATES FOUND FROM THE EMPLOYEE</h1>  
 {% endif %}  
 </div>  
</div>  
 </div>

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon"/>  
<link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
<!-- Bootstrap CSS -->  
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
<!-- Site CSS -->  
<link rel="stylesheet" href="{% static 'style.css'%}">  
<link rel="stylesheet" href="{% static 'table.css'%}">  
<!-- Responsive CSS -->  
<link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
<!-- Custom CSS -->  
<link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
<!-- Modernizer for Portfolio -->  
<script src="{% static 'js/modernizer.js'%}"></script>  
  
<!--[if lt IE 9]>  
<script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
<script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
<![endif]-->  
  
<link href="{% static 'vendor/mdi-font/css/material-design-iconic-font.min.css'%}" rel="stylesheet" media="all">  
<link href="{% static 'vendor/font-awesome-4.7/css/font-awesome.min.css'%}" rel="stylesheet" media="all">  
<!-- Font special for pages-->  
<link href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i,800,800i"  
 rel="stylesheet">  
  
<!-- Vendor CSS-->  
<link href="{% static 'vendor/select2/select2.min.css'%}" rel="stylesheet" media="all">  
<link href="{% static 'vendor/datepicker/daterangepicker.css'%}" rel="stylesheet" media="all">  
  
<!-- Main CSS-->  
<link href="{% static 'css/main.css'%}" rel="stylesheet" media="all">  
<link href="{% static 'css/main.min.css'%}" rel="stylesheet" media="all">  
  
<![endif]-->  
  
</head>  
<body>  
  
<!-- LOADER -->  
<div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
</div><!-- end loader -->  
<!-- END LOADER -->  
  
<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
</div>  
<header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"  
 aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}"  
 alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/admin1">Home</a></li>  
 <li><a href="/adminqueue">Register Queue</a></li>  
 <li><a href="/adminupdate">Updates</a></li>  
 <li><a href="/adminreq">Requirements</a></li>  
 <li><a href="/adminerror">Errors</a></li>  
 <li><a class="active" href="">Allocation</a></li>  
 <li><a href="/calculation">Efficiency</a></li>  
 <li><a href="/admin/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
  
<div class="banner-area banner-bg-1" style="padding-top:40px">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Admin</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Admin</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 {% if employee %}  
  
 <div class="wrapper wrapper--w790">  
 <div class="card card-5">  
 <div class="card-heading">  
 <h2 class="title">MACHINE ALLOCATION</h2>  
 </div>  
 <div class="card-body">  
 <form action="/alocations" method="POST">  
 {% csrf\_token %}  
 <div class="form-row">  
 <div class="name">EMPLOYEE</div>  
 <div class="value">  
 <div class="input-group">  
 <div class="rs-select2 js-select-simple select--no-search">  
 <select name="empname">  
 <option disabled="disabled" selected="selected">Choose option</option>  
 {% for emp in employee %}  
 <option value="{{emp.name}}">{{emp.name}}</option>  
 {% endfor %}  
 </select>  
 <div class="select-dropdown"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="name">MACHINE</div>  
 <div class="value">  
 <div class="input-group">  
 <div class="rs-select2 js-select-simple select--no-search">  
 <select name="machname">  
 <option disabled="disabled" selected="selected">Choose option</option>  
 <option value="CRUSHER">CRUSHER</option>  
 <option value="MILLS">MILLS</option>  
 <option value="GRINDER">GRINDER</option>  
 </select>  
 <div class="select-dropdown"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div>  
 <button class="btn btn--radius-2 btn--red" type="submit">ALLOCATE</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
</div>  
{% endif %}  
</div>

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />  
 <link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
 <!-- Site CSS -->  
 <link rel="stylesheet" href="{% static 'style.css'%}">  
 <link rel="stylesheet" href="{% static 'table.css'%}">  
 <!-- Responsive CSS -->  
 <link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
  
</head>  
<body>  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/admin1">Home</a></li>  
 <li><a href="/adminqueue">Register Queue</a></li>  
 <li><a href="/adminupdate">Updates</a></li>  
 <li><a href="/adminreq">Requirements</a></li>  
 <li><a href="/adminerror">Errors</a></li>  
 <li><a href="/alocation">Allocation</a></li>  
 <li><a class="active" href="/efficiency">Efficiency</a></li>  
 <li><a href="/admin/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Admin</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Admin</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div><br>  
 <table align="center">  
 <tr>  
 <th colspan="7">OUTPUT DETAILS</th>  
 </tr>  
 <tr>  
 <th>TOTAL PRODUCTS</th>  
 <th>PRODUCTS DONE</th>  
 <th>PRODUCT TO DONE</th>  
 <th>AVG PRODUCTION RATE PER SHIFT</th>  
 <th>EFFICIENCY</th>  
 <th>IF EFFICIENCY TO</th>  
 <th>RATE PER SHIFT FOR GIVEN EFFICIENCY</th>  
 </tr>  
 <tr>  
  
 <td>{{totproduct}}</td>  
 <td>{{proddone}}</td>  
 <td>{{prodtodone}}</td>  
 <td>{{avgprod}}</td>  
 <td>{{eff}}</td>  
 <td><form action="/efficiency" method="post">  
 {% csrf\_token %}  
 <input type="number" name="eff" placeholder="Enter efficiency to..." required/><br><input type="submit" class="btn btn-primary" style="background-color:blue" value="CALCULATE"></form></td>  
 {% if reqout %}  
 <td>{{reqout}}</td>  
 {% else %}  
 <td>--</td>  
 {% endif %}  
 </tr>  
 </table >  
 </div>  
<br>

<!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
</div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a class="active" href="">Customer</a></li>  
 <li><a href="/cusupdate">Update</a></li>  
 <li><a href="/purchaselist">PurchaseList</a></li>  
 <li><a href="/purchasedmaterial">Purchased Material List</a></li>  
 <li><a href="/customer/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h1 align="center" style="color:white">WELCOME TO CUSTOMER PAGE</h1>  
 <h2>Customer</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Customer</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>

<div class="main">  
  
 <section class="signup">  
 <!-- <img src="images/signup-bg.jpg" alt=""> -->  
 <div class="container">  
 <div class="signup-content">  
 <form method="post" id="signup-form" class="signup-form">  
 {% csrf\_token %}  
 <h2 class="form-title">Create account</h2>  
 <div class="form-group">  
 <input type="text" class="form-input" name="cname" id="name" placeholder="Your Name" required/>  
 </div>  
 <div class="form-group">  
 <input type="email" class="form-input" name="cemail" id="email" placeholder="Your Email" required/>  
 </div>  
 <div class="form-group">  
 <input type="date" class="form-input" name="cdate" id="date" placeholder="Your Birth Date" required/>  
 </div>  
 <div class="form-group">  
 <input type="number" class="form-input" max="10000000000" name="cno" id="no" placeholder="Your Contact No" required/>  
 </div>  
 <div class="form-group">  
 <input type="text" class="form-input" name="caddress" id="address" placeholder="Your Address" required/>  
 </div>  
 <div class="form-group">  
 <input type="password" class="form-input" name="cpassword" id="password" placeholder="Password" required/>  
 <span toggle="#password" class="zmdi zmdi-eye field-icon toggle-password"></span>  
 </div>  
 <div class="form-group">  
 <input type="checkbox" name="agree-term" id="agree-term" class="agree-term" required/>  
 <label for="agree-term" class="label-agree-term"><span><span></span></span>I agree all statements in <a href="#" class="term-service">Terms of service</a></label>  
 </div>  
 <div class="form-group">  
 <input type="submit" name="submit" id="submit" class="form-submit" value="Sign up"/>  
 </div>  
 </form>  
 <p class="loginhere">  
 Have already an account ? <a href="/customerlogin" class="loginhere-link">Login here</a>  
 </p>  
 </div>  
 </div>  
 </section>  
  
</div>

<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
 <div class="phone-box">  
 <a href="tel:1234567890"><i class="fa fa-phone" aria-hidden="true"></i> +1 234 567 890</a>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-6 col-sm-6">  
 <div class="right-top">  
 <div class="social-box">  
 <ul>  
 <li><a href="#"><i class="fa fa-facebook-square" aria-hidden="true"></i></a></li>  
 <li><a href="#"><i class="fa fa-instagram" aria-hidden="true"></i></a></li>  
 <li><a href="#"><i class="fa fa-linkedin-square" aria-hidden="true"></i></a></li>  
 <li><a href="#"><i class="fa fa-twitter-square" aria-hidden="true"></i></a></li>  
 <li><a href="#"><i class="fa fa-rss-square" aria-hidden="true"></i></a></li>  
 <ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
</div>  
<header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"  
 aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}"  
 alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/datatrainer">Home</a></li>  
 <li><a href="/trainmachine">Train</a></li>  
 <li><a class="active" href="/changemach">Change Machine</a></li>  
 <li><a href="/datatrainer/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
</header>  
  
<div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>DataTrainer</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">DataTrainer</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div><BR>  
  
{% if data %}  
 <div class="wrapper wrapper--w790" id="a1" style="display:block">  
 <div class="card card-5">  
 <div class="card-heading">  
 <h2 class="title">UPDATE EQUIPMENTS</h2>  
 </div>  
 <div class="card-body">  
 <form action="/alocationafterror" method="POST">  
 {% csrf\_token %}  
 <div class="form-row">  
 <div class="name">QUALITY</div>  
 <div class="value">  
 <div class="input-group">  
 <div class="rs-select2 js-select-simple select--no-search">  
 <select name="empname2">  
 {% for datas in data %}  
 <option value="{{datas.name}}">{{datas.name}}</option>  
 {% endfor %}  
 </select>  
 <div class="select-dropdown"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="name">QUALITY</div>  
 <div class="value">  
 <div class="input-group">  
 <div class="rs-select2 js-select-simple select--no-search">  
 <select name="machname2">  
 {% for datas in data %}  
 {% if datas.machine == "CRUSHER" %}  
 <option value="CRUSHER2">CRUSHER2</option>  
 <option value="CRUSHER3">CRUSHER3</option>  
 {% elif datas.machine == 'MILLS' %}  
 <option value="MILLS2">MILLS2</option>  
 <option value="MILLS3">MILLS3</option>  
 {% elif datas.machine == 'GRINDER' %}  
 <option value="GRINDER2">GRINDER2</option>  
 <option value="GRINDER3">GRINDER3</option>  
 {% elif datas.machine == 'BLENDER' %}  
 <option value="BLENDER2">BLENDER2</option>  
 <option value="BLENDER3">BLENDER3</option>  
 {% elif datas.machine == 'HEATER' %}  
 <option value="HEATER2">HEATER2</option>  
 <option value="HEATER3">HEATER3</option>  
 {% endif %}{% endfor %}  
 </select>  
 <div class="select-dropdown"></div>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div>  
 <button class="btn btn--radius-2 btn--red" type="submit">UPDATE</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
{% endif %}

<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/employee">Home</a></li>  
 <li><a href="/empupdate">Update</a></li>  
 <li><a class="active" href="/emprequirement">Requirement</a></li>  
 <li><a href="/empupload">Upload</a></li>  
 <li><a href="/emprun">Run Machine</a></li>  
 <li><a href="/employee/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Employee</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Employee</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
  
{% if requirement %}  
<div>  
 <table align="center">  
 <tr>  
 <th colspan="2">REQUIREMENT</th>  
 </tr>  
 <tr>  
 <th>MACHINE</th>  
 <th>QUANTITY</th>  
 </tr>  
 <tr>  
 {% for j in requirement%}  
 <td >{{j.machines}}</td>  
 <td>{{j.quantity}}</td>  
 </tr>  
 {% endfor %}  
 </table >  
 </div>  
 {% else %}  
 <h1 style="color:red" align="center">THERE IS NO REQUIREMENT FROM THE EMPLOYEE</h1>  
 {% endif %}  
 </div><br>

<div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/employee">Home</a></li>  
 <li><a href="/empupdate">Update</a></li>  
 <li><a href="/emprequirement">Requirement</a></li>  
 <li><a href="/empupload">Upload</a></li>  
 <li><a class="active" href="/emprun">Run Machine</a></li>  
 <li><a href="/employee/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1" style="padding-top:40px">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Employee</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Employee</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
{% if alocations %}  
 <div class="wrapper wrapper--w790" id="a1" style="display:block;">  
 <div class="card card-5">  
 <div class="card-heading">  
 <h2 class="title">EMPLOYEE MACHINE INFO</h2>  
 </div>  
 <div class="card-body">  
 <form action="/errorupdate/" method="POST">  
 {% csrf\_token %}  
 <div class="form-row">  
 <div class="name">EMPLOYEE</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" value="{{alocations.name}}" name="name" readonly>  
 </div>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="name">MACHINE</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" value="{{alocations.machine\_name}}" name="quan" readonly>  
 </div>  
 </div>  
 </div>  
 <div>  
 {% if error %}  
 <button class="btn btn--radius-2 btn--red" type="submit" disabled>RUN</button>  
 <br><br>  
 <p style="color:red">Error Has Been Updated Already</p>  
 {% else %}  
 <input value="employee" name="type" hidden>  
 <button class="btn btn--radius-2 btn--red" type="submit">RUN</button>  
 {% endif %}  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
{% else %}  
 <center><h1 style="padding:100px;color:orange">MACHINE HASN'T ALLOCATED FOR <br>THE EMPLOYEE</h1></center>  
{% endif %}  
 <br>

<link href="{% static 'vendor/select2/select2.min.css'%}" rel="stylesheet" media="all">  
 <link href="{% static 'vendor/datepicker/daterangepicker.css'%}" rel="stylesheet" media="all">  
<script>function myFunction1() {  
 var x = document.getElementById("a1");  
 var y = document.getElementById("a2");  
 x.style.display = "block";  
 y.style.display = "none";  
}  
function myFunction2() {  
 var x = document.getElementById("a1");  
 var y = document.getElementById("a2");  
 y.style.display = "block";  
 x.style.display = "none";  
}</script>  
 <!-- Main CSS-->  
 <link href="{% static 'css/main.css'%}" rel="stylesheet" media="all">  
 <link href="{% static 'css/main.min.css'%}" rel="stylesheet" media="all">  
</head>  
<body style="color:black">  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/employee">Home</a></li>  
 <li><a href="/empupdate">Update</a></li>  
 <li><a href="/emprequirement">Requirement</a></li>  
 <li><a class="active" href="/empupload">Upload</a></li>  
 <li><a href="/emprun">Run Machine</a></li>  
 <li><a href="/employee/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
 <div class="banner-area banner-bg-1" style="padding-top:40px">  
 <center><div>  
 <button class="btn btn-primary" onclick="myFunction1()">UPDATE</button>  
 <button class="btn btn-primary" onclick="myFunction2()">REQUIREMENT</button></div></center>  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Employee</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Employee</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="wrapper wrapper--w790" id="a1" style="display:block">  
 <div class="card card-5">  
 <div class="card-heading">  
 <h2 class="title">UPDATE EQUIPMENTS</h2>  
 </div>  
 <div class="card-body">  
 <form action="/employeeupdate/" method="POST">  
 {% csrf\_token %}  
 <div class="form-row">  
 <div class="name">MACHINE OR TOOL NAME</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" name="mach" placeholder="Enter Machine or Tool" required>  
 </div>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="name">QUANTITY</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" name="quan" placeholder="Enter Quantity" required>  
 </div>  
 </div>  
 </div>  
 <div>  
 <button class="btn btn--radius-2 btn--red" type="submit">UPDATE</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div>  
 <div class="wrapper wrapper--w790" id="a2" style="display:none">  
 <div class="card card-5">  
 <div class="card-heading">  
 <h2 class="title">REQUEST EQUIPMENTS</h2>  
 </div>  
 <div class="card-body">  
 <form action="/employeerequirements/" method="POST">  
 {% csrf\_token %}  
 <div class="form-row">  
 <div class="name">MACHINE OR TOOL NAME</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" name="mach1" placeholder="Enter Machine or Tool" required>  
 </div>  
 </div>  
 </div>  
 <div class="form-row">  
 <div class="name">QUANTITY</div>  
 <div class="value">  
 <div class="input-group">  
 <input class="input--style-5" type="text" name="quan1" placeholder="Enter Quantity" required>  
 </div>  
 </div>  
 </div>  
 <div>  
 <button class="btn btn--radius-2 btn--red" type="submit">REQUEST</button>  
 </div>  
 </form>  
 </div>  
 </div>  
 </div><BR>

<link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
<head>  
<script>  
window.onload = function () {  
  
var chart = new CanvasJS.Chart("chartContainer", {  
 animationEnabled: true,  
 zoomEnabled: true,  
 title:{  
 text: "CONTROL MONITORING"  
 },  
 data: data // random generator below  
});  
chart.render();  
  
}  
  
var limit = 70;  
var x = 0;  
var y = JSON.parse("{{data|escapejs}}");  
var data = [];  
var dataSeries = { type: "line" };  
var dataPoints = [];  
for (var i = 0; i < y.length; i += 1) {  
 dataPoints.push({  
 x: x++,  
 y: y[i]  
 });  
}  
dataSeries.dataPoints = dataPoints;  
data.push(dataSeries);  
  
</script>  
<script type="text/javascript" src="{% static 'graph.js'%}"></script></head>  
<body><br><br>  
<div id="chartContainer" style="height: 500px; width: 100%;">  
</div>  
{% if error %}  
 <div align="center">  
 <h2 style="color:red">{{error}}</h2>  
 <form action="/updateadmin/{{error}}/{{machine}}/{{type}}" method="POST">  
 {% csrf\_token %}  
 <input value="{{name}}" name="name" hidden>  
 <div><button type="submit" class="btn btn-primary">SEND ERROR</button></form>  
<button type="button" style="float:center" onclick="history.back()" class="btn btn-primary" >BACK</button></div></div>  
{% else %}  
<button type="button" style="float:center" onclick="history.back()" class="btn btn-primary" >BACK</button>  
{% endif %}  
</body>

<div class="container p-0">  
 <div class="card px-4">  
 <p class="h8 py-3">Payment Details</p>  
 <form action="/payment/{{detail.orderid}}" method="post">  
 {% csrf\_token %}  
 <div class="row gx-3">  
 <div class="col-12">  
 <div class="d-flex flex-column">  
 <p class="text mb-1">User Name</p> <input style="background:#223c60" class="form-control mb-3" type="text" placeholder="Name" value="{{detail.name}}" readonly>  
 </div>  
 </div>  
 <div class="col-12">  
 <div class="d-flex flex-column">  
 <p class="text mb-1">Card Number</p> <input class="form-control mb-3" name="card" type="text" placeholder="1234 5678 435678" required>  
 </div>  
 </div>  
 <div class="col-6">  
 <div class="d-flex flex-column">  
 <p class="text mb-1">Expiry</p> <input class="form-control mb-3" name="expire" type="month" placeholder="MM/YYYY" required>  
 </div>  
 </div>  
 <div class="col-6">  
 <div class="d-flex flex-column">  
 <p class="text mb-1">CVV</p> <input class="form-control mb-3 pt-2" type="password" placeholder="\*\*\*" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <button class="btn btn-success" style="height:65px;width:100%">Pay $ {{detail.amount}} <span class="fas fa-arrow-right" style="padding-left:30px"></span></button>  
 </div><br><br><br><br><br>  
 </div></form>  
 </div>  
</div>

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />  
 <link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
 <!-- Site CSS -->  
 <link rel="stylesheet" href="{% static 'style.css'%}">  
 <link rel="stylesheet" href="{% static 'table.css'%}">  
 <!-- Responsive CSS -->  
 <link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
  
</head>  
<body>  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.png'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/customer">Customer</a></li>  
 <li><a href="/cusupdate">Update</a></li>  
 <li><a href="/purchaselist">Purchase List</a></li>  
 <li><a class="active" href="/purchasedmaterial">Purchased Material List</a></li>  
 <li><a href="/customer/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Customer</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Customer</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 {% if pur %}  
 <table align="center" style="width:70%">  
 <tr>  
 <th colspan="7">PURCHASED LIST</th>  
 </tr>  
 <tr>  
 <th>TYPE</th>  
 <th>ORDER ID</th>  
 <th>QUANTITY</th>  
 <th>QUALITY</th>  
 <th>TIME TO NEED</th>  
 <th>WARRANTY</th>  
 <th>AMOUNT PAID</th>  
 </tr>  
 <tr>  
 {% for i in pur %}  
 <td>{{i.type}}</td>  
 <td>{{i.orderid}}</td>  
 <td>{{i.quantity}}</td>  
 <td>{{i.quality}}</td>  
 <td>{{i.time}}</td>  
 <td>{{i.warranty}}</td>  
 <td>{{i.amount}}</td>  
 </tr>  
 {% endfor %}  
 </table >  
 {% else %}  
 <h1 style="color:orange" align="center">THERE IS NO PURCHASED LIST TO VIEW</h1>  
 {% endif %}  
 </div>  
<br>

<link rel="shortcut icon" href="{% static 'images/favicon.ico' %}" type="image/x-icon" />  
 <link rel="apple-touch-icon" href="{%static 'images/apple-touch-icon.png'%}">  
  
 <!-- Bootstrap CSS -->  
 <link rel="stylesheet" href="{% static 'css/bootstrap.min.css'%}">  
 <!-- Site CSS -->  
 <link rel="stylesheet" href="{% static 'style.css'%}">  
 <link rel="stylesheet" href="{% static 'table.css'%}">  
 <!-- Responsive CSS -->  
 <link rel="stylesheet" href="{% static 'css/responsive.css'%}">  
 <!-- Custom CSS -->  
 <link rel="stylesheet" href="{% static 'css/custom.css'%}">  
  
 <!-- Modernizer for Portfolio -->  
 <script src="{% static 'js/modernizer.js'%}"></script>  
  
 <!--[if lt IE 9]>  
 <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  
 <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  
 <![endif]-->  
  
</head>  
<body>  
  
 <!-- LOADER -->  
 <div id="preloader">  
 <div class="loader">  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_bar"></div>  
 <div class="loader\_\_ball"></div>  
 </div>  
 </div><!-- end loader -->  
 <!-- END LOADER -->  
  
 <div class="top-bar">  
 <div class="container-fluid">  
 <div class="row">  
 <div class="col-md-6 col-sm-6">  
 <div class="left-top">  
 <div class="email-box">  
 <a href="#"><i class="fa fa-envelope-o" aria-hidden="true"></i> youremail@gmail.com</a>  
 </div>  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </div>  
 <header class="header header\_style\_01">  
 <nav class="megamenu navbar navbar-default">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar" aria-expanded="false" aria-controls="navbar">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <a class="navbar-brand" href="index.html"><img src="{% static 'images/logos/logo.jpg'%}" alt="image"></a>  
 </div>  
 <div id="navbar" class="navbar-collapse collapse">  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/production">Home</a></li>  
 <li><a href="/productionview">Production</a></li>  
 <li><a class="active" href="/purchaseview">Delivery</a></li>  
 <li><a href="/monitor">Monitor</a></li>  
 <li><a href="/production/logout">Logout</a></li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
 </header>  
  
 <div class="banner-area banner-bg-1">  
 <div class="container">  
 {% if messages %}  
 {% for message in messages %}  
 <b><h2 align="center" style="color:green">{{message}}</h2></b>  
 {% endfor %}  
 {% endif %}  
 <div class="row">  
 <div class="col-md-12">  
 <div class="banner">  
 <h2>Production</h2>  
 <ul class="page-title-link">  
 <li><a href="#">Home</a></li>  
 <li><a href="#">Production</a></li>  
 </ul>  
 </div>  
 </div>  
 </div>  
 </div>  
 {% if purchaselists %}  
 <table align="center">  
 <tr>  
 <th colspan="7">PRODUCTION LIST</th>  
 </tr>  
 <tr>  
 <th>NAME</th>  
 <th>TYPE</th>  
 <th>QUANTITY (TONS)</th>  
 <th>TIME TO NEED</th>  
 <th>PRODUCTION</th>  
 <th>PAYMENT</th>  
 </tr>  
 <tr>  
 {% for i in purchaselists %}  
 <td>{{i.name}}</td>  
 <td>{{i.type}}</td>  
 <td>{{i.quantity}}</td>  
 <td>{{i.time}}</td>  
 {% if i.type == 'SCRAP' %}  
 {% if i.quantity > scrap %}  
 <td>UNDER PROCESS.....</td>  
 {% elif i.quantity < scrap and i.payment is False %}  
 <td>PRODUCTION COMPLETE BUT WAITING FOR PAYMENT</td>  
 {% else %}  
 <td>PRODUCTION COMPLETE<a href="/delivery/{{i.orderid}}/{{i.name}}/{{i.type}}"><button type="button" class="btn btn-primary" >SEND</button></a></td>  
 {% endif %}  
 {% elif i.type == 'SHEET' %}  
 {% if i.quantity > sheet %}  
 <td>UNDER PROCESS.....</td>  
 {% elif i.quantity <= sheet and i.payment is False %}  
 <td>PRODUCTION COMPLETE BUT WAITING FOR PAYMENT</td>  
 {% else %}  
 <td>PRODUCTION COMPLETE<a href="/delivery/{{i.orderid}}/{{i.name}}/{{i.type}}"><button type="button" class="btn btn-primary" >SEND</button></a></td>  
 {% endif %}  
 {% elif i.type == 'ROD' %}  
 {% if i.quantity > rod %}  
 <td>UNDER PROCESS.....</td>  
 {% elif i.quantity <= rod and i.payment is False %}  
 <td>PRODUCTION COMPLETE BUT WAITING FOR PAYMENT</td>  
 {% else %}  
 <td>PRODUCTION COMPLETE<a href="/delivery/{{i.orderid}}/{{i.name}}/{{i.type}}"><button type="button" class="btn btn-primary" >SEND</button></a></td>  
 {% endif %}  
 {% endif %}  
 {% if i.payment is False %}  
 <td>NOT PAID</td>  
 {% else %}  
 <td>PAID</td>  
 {% endif %}  
  
  
  
  
 </tr>  
 {% endfor %}  
 </table >  
 {% else %}  
 <h1 style="color:orange" align="center">THERE IS NO PRODUCTIONLIST TO VIEW</h1>  
 {% endif %}  
 </div>

**Backend Codings:**

from django.shortcuts import render,redirect  
from .models import \*  
from django.contrib import messages  
from django.db import connection  
import numpy as np  
import random  
import pandas as pd  
from sklearn.linear\_model import LinearRegression  
import time  
from django.conf import settings  
from json import dumps  
import string  
  
  
cursor=connection.cursor()  
standout = 200  
totproduct = 50000  
def home(request):  
 return render(request,'home.html')  
def customer(request):  
 return render(request,'customer.html')  
def cusupdate(request):  
 return render(request,'cusupdate.html')  
def purchaselists(request):  
 cusdetails = purchaselist.objects.filter(name=request.session['cname']).order\_by('name')  
 dict = {}  
 for i in cusdetails:  
 dict[i.type] = int(i.quantity)  
 prodlist = productionlist.objects.all()  
 dict2 = {}  
 for j in prodlist:  
 dict2[j.machine] = (j.quantity)  
 return render(request,'purchaselist.html',{'cusdetails':cusdetails,'scrap':dict2.get('CRUSHER'),'sheet':dict2.get('MILLS'),'rod':dict2.get('GRINDER')})  
def employee(request):  
 update=updates.objects.filter(name=request.session['ename'])  
 requirement = requirements.objects.filter(name=request.session['ename'])  
 return render(request,'employee.html',{'update':update,'requirement':requirement})  
def empupdate(request):  
 update = updates.objects.filter(name=request.session['ename'])  
 return render(request, 'empupdate.html', {'update': update})  
def emprequirement(request):  
 requirement = requirements.objects.filter(name=request.session['ename'])  
 return render(request, 'emprequirement.html', {'requirement':requirement})  
def empupload(request):  
 return render(request, 'empupload.html')  
def emprun(request):  
 try:  
 alocations = alocation.objects.get(name=request.session['ename'])  
 try:  
 error=errorupdate.objects.get(name=request.session['ename'])  
 return render(request, 'emprun.html',{'alocations':alocations,'error':error})  
 except:  
 return render(request, 'emprun.html', {'alocations': alocations})  
 except:  
 return render(request, 'emprun.html')  
def datatrainer(request):  
 data = datatrainerview.objects.all()  
 return render(request,'datatrainer.html',{'data':data})  
def datatrain(request):  
 return render(request,'datatrain.html')  
def datachangemachine(request):  
 data = datatrainerview.objects.all()  
 return render(request,'datachangemachine.html',{'data':data})  
def admin(request):  
 employee = employees.objects.all().order\_by('name')  
 access=adminaccess.objects.all()  
 update = updates.objects.all().order\_by('name','machines')  
 requirement = requirements.objects.all().order\_by('name','machines')  
 error=errorupdate.objects.all()  
 path = f'{settings.MEDIA\_ROOT[0]}/efficiency.csv'  
 effdata = pd.read\_csv(path)  
 tons = effdata['Tons']  
 tonlist = tons.tolist()  
 avgoutput = sum(tonlist) / len(tonlist)  
 avgoutput = round(avgoutput, 3)  
 prodtodone = totproduct - sum(tonlist)  
 eff = (avgoutput / standout) \* 100  
 eff = round(eff, 3)  
 return render(request,'admin.html',{'update':update,'requirement':requirement,'error':error,'access':access,'employee':employee,'totproduct': totproduct, 'proddone': sum(tonlist), 'prodtodone': prodtodone,  
 'avgprod': avgoutput, 'eff': eff})  
def adminupdate(request):  
 update = updates.objects.all().order\_by('name', 'machines')  
 return render(request,'adminupdate.html',{'update':update})  
def adminreq(request):  
 requirement = requirements.objects.all().order\_by('name','machines')  
 return render(request,'adminreq.html',{'requirement':requirement})  
def adminerror(request):  
 error=errorupdate.objects.all()  
 return render(request,'adminerror.html',{'error':error})  
def alocations(request):  
 employee = employees.objects.all().order\_by('name')  
 return render(request, 'alocation.html', {'employee':employee})  
def registerqueue(request):  
 access=adminaccess.objects.all()  
 return render(request, 'registerqueue.html', {'access':access})  
def calculation(request):  
 path = f'{settings.MEDIA\_ROOT[0]}/efficiency.csv'  
 effdata = pd.read\_csv(path)  
 tons = effdata['Tons']  
 tonlist = tons.tolist()  
 avgoutput = sum(tonlist) / len(tonlist)  
 avgoutput = round(avgoutput, 3)  
 prodtodone = totproduct - sum(tonlist)  
 eff = (avgoutput / standout) \* 100  
 eff = round(eff, 3)  
 return render(request, 'calculation.html', {'totproduct': totproduct, 'proddone': sum(tonlist), 'prodtodone': prodtodone,  
 'avgprod': avgoutput, 'eff': eff})  
def customerlogin(request):  
 if request.session.get('cname'):  
 return redirect('/customer')  
 else:  
 if request.method=='POST':  
 try:  
 details=customers.objects.get(name=request.POST['cname'],password=request.POST['cpass'])  
 messages.success(request,"SUCCESSFULY LOGIN TO %s's"%details.name)  
 request.session['cname'] = details.name  
 return redirect('/customer')  
 except customers.DoesNotExist as e:  
 messages.info(request,'INVALID USERNAME OR PASSWORD')  
 return redirect('/customerlogin')  
 return render(request,'customerlogin.html')  
def employeelogin(request):  
 if request.session.get('ename'):  
 return redirect('/employee')  
 else:  
 if request.method=='POST':  
 try:  
 details=employees.objects.get(name=request.POST['ename'],password=request.POST['epass'])  
 messages.success(request,"SUCCESSFULY LOGIN TO %s's"%details.name)  
 request.session['ename']=details.name  
 return redirect('/employee',)  
 except employees.DoesNotExist as e:  
 messages.info(request,'INVALID USERNAME OR PASSWORD')  
 return redirect('/employeelogin')  
 return render(request,'employeelogin.html')  
def datatrainerlogin(request):  
 if request.session.get('datatrainer'):  
 return redirect('/datatrainer')  
 else:  
 if request.method=='POST':  
 try:  
 details=datatrainers.objects.get(name=request.POST['dname'],password=request.POST['dpass'])  
 messages.success(request,"SUCCESSFULY LOGIN TO %s's"%details.name)  
 request.session['datatrainer'] = details.name  
 return redirect('/datatrainer')  
 except datatrainers.DoesNotExist as e:  
 messages.info(request,'INVALID USERNAME OR PASSWORD')  
 return redirect('/datatrainerlogin')  
 return render(request,'datatrainerlogin.html')  
def productionlogin(request):  
 if request.session.get('production'):  
 return redirect('/production')  
 else:  
 if request.method=='POST':  
 try:  
 details=productions.objects.get(name=request.POST['pname'],password=request.POST['ppass'])  
 messages.success(request,"SUCCESSFULY LOGIN TO %s's"%details.name)  
 request.session['production'] = details.name  
 return redirect('/production')  
 except productions.DoesNotExist as e:  
 messages.info(request,'INVALID USERNAME OR PASSWORD')  
 return redirect('/productionlogin')  
 return render(request,'productionlogin.html')  
def adminlogin(request):  
 if request.session.get('admin'):  
 return redirect('/admin1')  
 else:  
 if request.method=='POST':  
 try:  
 details=admins.objects.get(name=request.POST['aname'],password=request.POST['apass'])  
 messages.success(request,"SUCCESSFULY LOGIN TO %s's"%details.name)  
 request.session['admin'] = details.name  
 return redirect('/admin1')  
 except admins.DoesNotExist as e:  
 messages.info(request,'INVALID USERNAME OR PASSWORD')  
 return redirect('/adminlogin')  
 return render(request,'adminlogin.html')  
def customerregister(request):  
 if request.method=='POST':  
 name=request.POST['cname']  
 email=request.POST['cemail']  
 password = request.POST['cpassword']  
 date = request.POST['cdate']  
 number = int(request.POST['cno'])  
 address = request.POST['caddress']  
 customers(name=name,email=email,password=password,date=date,number=number,address=address).save()  
 messages.success(request, "SUCCESSFULY REGISTERED AS %s" %name)  
 return redirect('/')  
 return render(request, 'customerregister.html')  
def register(request,designation):  
 if request.method=='POST':  
 name=request.POST['ename']  
 email=request.POST['eemail']  
 password = request.POST['epassword']  
 date=request.POST['edate']  
 number=request.POST['eno']  
 address = request.POST['eaddress']  
 adminaccess(designation=designation,name=name,email=email,password=password,date=date,number=number,address=address).save()  
 messages.success(request, "SUCCESSFULY REGISTERED AS %s BUT TO WAIT FOR CONFIRMATION" %name)  
 return redirect('/')  
 return render(request, 'employeeregister.html')  
def adminacc(request,designation,id):  
 t=adminaccess.objects.get(id=id)  
 if designation=='employee':  
 employees(name=t.name,email=t.email,password=t.password,date=t.date,number=t.number,address=t.address).save()  
 delete = adminaccess.objects.get(id=id)  
 delete.delete()  
 messages.success(request, f"EMPLOYEE REGISTERATION FOR {t.name} IS DONE SUCCESSFULLY")  
 return redirect('/adminqueue')  
 elif designation=='datatrainer':  
 datatrainers(name=t.name,email=t.email,password=t.password,date=t.date,number=t.number,address=t.address).save()  
 delete = adminaccess.objects.get(id=id)  
 delete.delete()  
 messages.success(request, f"DATATRAINER REGISTERATION FOR {t.name} IS DONE SUCCESSFULLY")  
 return redirect('/adminqueue')  
 return render(request,"admins.html")  
  
def employeeregister(request):  
 if request.method=='POST':  
 name=request.POST['ename']  
 email=request.POST['eemail']  
 password = request.POST['epassword']  
 adminaccess(name=name,email=email,password=password).save()  
 messages.success(request, "SUCCESSFULY REGISTERED AS %s AND TO WAIT FOR CONFIRMATION" %name)  
 return redirect('/')  
 return render(request, 'employeeregister.html')  
def datatrainerregister(request):  
 if request.method=='POST':  
 name=request.POST['dname']  
 email=request.POST['demail']  
 password = request.POST['dpassword']  
 adminaccess(name=name,email=email,password=password).save()  
 messages.success(request, "SUCCESSFULY REGISTERED AS %s AND TO WAIT FOR THE CONFIRMATION" %name)  
 return redirect('/')  
 return render(request, 'datatrainerregister.html')

def startmachine(request):  
 if request.method=='POST':  
 name=request.POST['name']  
 type=request.POST['type']  
 train = alocation.objects.filter(name=name)  
 tmachine = list()  
 for i in train:  
 tmachine.append(i.machine\_name)  
 tmachine.append(i.status)  
 cname=[]  
 cqty=[]  
 lname=purchaselist.objects.all()  
 for j in lname:  
 cname.append(j.name)  
 cqty.append(j.quantity)  
 path = f'{settings.MEDIA\_ROOT[0]}/factory1.csv'  
 data = pd.read\_csv(path)  
 crusher = LinearRegression()  
 mills = LinearRegression()  
 grinder = LinearRegression()  
 blender = LinearRegression()  
 heater = LinearRegression()  
  
 x1 = data.loc[:,  
 ['Machine1.RawMaterial.Property1', 'Machine1.RawMaterial.Property2', 'Machine1.RawMaterial.Property3',  
 'Machine1.RawMaterial.Property4', 'Machine1.RawMaterialFeederParameter.U.Actual',  
 'Machine1.Zone1Temperature.C.Actual', 'Machine1.Zone2Temperature.C.Actual', 'Machine1.MotorRPM.C.Actual',  
 'Machine1.MaterialPressure.U.Actual']]  
 x2 = data.loc[:,  
 ['Machine2.RawMaterial.Property1', 'Machine2.RawMaterial.Property2', 'Machine2.RawMaterial.Property3',  
 'Machine2.RawMaterial.Property4', 'Machine2.RawMaterialFeederParameter.U.Actual',  
 'Machine2.Zone1Temperature.C.Actual', 'Machine2.Zone2Temperature.C.Actual', 'Machine2.MotorRPM.C.Actual',  
 'Machine2.MaterialPressure.U.Actual']]  
 x3 = data.loc[:,  
 ['Machine3.RawMaterial.Property1', 'Machine3.RawMaterial.Property2', 'Machine3.RawMaterial.Property3',  
 'Machine3.RawMaterial.Property4', 'Machine3.RawMaterialFeederParameter.U.Actual',  
 'Machine3.Zone1Temperature.C.Actual', 'Machine3.Zone2Temperature.C.Actual', 'Machine3.MotorRPM.C.Actual',  
 'Machine3.MaterialPressure.U.Actual']]  
 x4 = data.loc[:,  
 ['Machine4.Temperature1.C.Actual', 'Machine4.Temperature2.C.Actual', 'Machine4.Pressure.C.Actual',  
 'Machine4.Temperature3.C.Actual', 'Machine4.Temperature4.C.Actual']]  
 x5 = data.loc[:,  
 ['Machine5.Temperature1.C.Actual', 'Machine5.Temperature2.C.Actual',  
 'Machine5.Temperature3.C.Actual',  
 'Machine5.Temperature4.C.Actual', 'Machine5.Temperature5.C.Actual',  
 'Machine5.ExitTemperature.U.Actual']]  
 loops=500  
 if tmachine[0] == 'CRUSHER' or tmachine[0]=='CRUSHER2' or tmachine[0]=='CRUSHER3':  
 crusher.fit(x1, data['Machine1.MaterialTemperature.U.Actual'])  
 if tmachine[1]=='changed':  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 xtest1 = data.loc[  
 data["index"] == rande, ['Machine1.RawMaterial.Property1', 'Machine1.RawMaterial.Property2',  
 'Machine1.RawMaterial.Property3', 'Machine1.RawMaterial.Property4',  
 'Machine1.RawMaterialFeederParameter.U.Actual',  
 'Machine1.Zone1Temperature.C.Actual',  
 'Machine1.Zone2Temperature.C.Actual', 'Machine1.MotorRPM.C.Actual',  
 'Machine1.MaterialPressure.U.Actual']]  
  
 pred = crusher.predict(xtest1)  
 demo.append(float(pred))  
 r = random.randint(1, 30)  
 pdata=productionlist.objects.filter(machine='CRUSHER')  
 list1=[]  
 for i in pdata:  
 list1.append(i.quantity)  
 add=int(list1[0])+r  
 productionlist.objects.filter(machine='CRUSHER').update(quantity=add)  
 data = dumps(demo)  
 return render(request, 'graph.html', {'data': data,'pdata': pdata})  
 else:  
 demo=[]  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 ran9 = [np.random.randint(1, 300, 9)]  
 xtest1 = data.loc[  
 data['index'] == rande, ['Machine1.RawMaterial.Property1', 'Machine1.RawMaterial.Property2',  
 'Machine1.RawMaterial.Property3', 'Machine1.RawMaterial.Property4',  
 'Machine1.RawMaterialFeederParameter.U.Actual',  
 'Machine1.Zone1Temperature.C.Actual',  
 'Machine1.Zone2Temperature.C.Actual', 'Machine1.MotorRPM.C.Actual',  
 'Machine1.MaterialPressure.U.Actual']]  
 mainr = random.choice([xtest1, ran9, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1, xtest1])  
 pred = crusher.predict(mainr)  
 if pred < 76 or pred > 84:  
 demo.append(float(pred))  
 r=random.randint(1,20)  
 productionlist.objects.filter(machine='CRUSHER').update(quantity=r)  
 data=dumps(demo)  
 pdata = productionlist.objects.filter(machine='CRUSHER')  
 error=f"ERROR OCCOURED IN {tmachine[0]} MACHINE DUE TO THE VARIATION OF INPUT PARAMETERSWHERE THE ERROR IS {pred}"  
 return render(request, 'graph.html',{'data':data,'error':error,'machine':tmachine[0],'pdata': pdata,'name':name,'type':type})  
 else:  
 demo.append(float(pred))  
  
 elif tmachine[0] == 'MILLS' or tmachine[0]=='MILLS2' or tmachine[0]=='MILLS3':  
 mills.fit(x2, data['Machine2.MaterialTemperature.U.Actual'])  
 if tmachine[1] == 'changed':  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 xtest2 = data.loc[  
 data['index'] == rande, ['Machine2.RawMaterial.Property1', 'Machine2.RawMaterial.Property2',  
 'Machine2.RawMaterial.Property3', 'Machine2.RawMaterial.Property4',  
 'Machine2.RawMaterialFeederParameter.U.Actual',  
 'Machine2.Zone1Temperature.C.Actual',  
 'Machine2.Zone2Temperature.C.Actual', 'Machine2.MotorRPM.C.Actual',  
 'Machine2.MaterialPressure.U.Actual']]  
  
 pred = mills.predict(xtest2)  
 demo.append(float(pred))  
 r = random.randint(1, 30)  
 pdata = productionlist.objects.filter(machine='MILLS')  
 list1 = []  
 for i in pdata:  
 list1.append(i.quantity)  
 add = int(list1[0]) + r  
 productionlist.objects.filter(machine='MILLS').update(quantity=add)  
 data = dumps(demo)  
 return render(request, 'graph.html', {'data': data,'pdata':pdata})  
 else:  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 ran9 = [np.random.randint(1, 300, 9)]  
 xtest2 = data.loc[  
 data['index'] == rande, ['Machine2.RawMaterial.Property1', 'Machine2.RawMaterial.Property2',  
 'Machine2.RawMaterial.Property3', 'Machine2.RawMaterial.Property4',  
 'Machine2.RawMaterialFeederParameter.U.Actual',  
 'Machine2.Zone1Temperature.C.Actual',  
 'Machine2.Zone2Temperature.C.Actual', 'Machine2.MotorRPM.C.Actual',  
 'Machine2.MaterialPressure.U.Actual']]  
 mainr = random.choice(  
 [xtest2, ran9, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2,  
 xtest2, xtest2, xtest2, xtest2, xtest2, xtest2, xtest2])  
 pred = mills.predict(mainr)  
 if pred < 68 or pred > 78:  
 demo.append(float(pred))  
 r = random.randint(1, 20)  
 productionlist.objects.filter(machine='MILLS').update(quantity=r)  
 data = dumps(demo)  
 pdata = productionlist.objects.filter(machine='MILLS')  
 error = f"ERROR OCCOURED IN {tmachine[0]} MACHINE DUE TO THE VARIATION OF INPUT PARAMETERSWHERE THE ERROR IS {pred}"  
 return render(request, 'graph.html', {'data': data, 'error': error, 'machine': tmachine[0],'pdata':pdata,'name':name,'type':type})  
 else:  
 demo.append(float(pred))  
  
 elif tmachine[0] == 'GRINDER' or tmachine[0]=='GRINDER2' or tmachine[0]=='GRINDER3':  
 grinder.fit(x3, data['Machine3.MaterialTemperature.U.Actual'])  
 if tmachine[1] == 'changed':  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 xtest3 = data.loc[  
 data['index'] == rande, ['Machine3.RawMaterial.Property1', 'Machine3.RawMaterial.Property2',  
 'Machine3.RawMaterial.Property3', 'Machine3.RawMaterial.Property4',  
 'Machine3.RawMaterialFeederParameter.U.Actual',  
 'Machine3.Zone1Temperature.C.Actual',  
 'Machine3.Zone2Temperature.C.Actual', 'Machine3.MotorRPM.C.Actual',  
 'Machine3.MaterialPressure.U.Actual']]  
  
 pred = grinder.predict(xtest3)  
 demo.append(float(pred))  
 r = random.randint(1, 30)  
 pdata = productionlist.objects.filter(machine='GRINDER')  
 list1 = []  
 for i in pdata:  
 list1.append(i.quantity)  
 add = int(list1[0]) + r  
 productionlist.objects.filter(machine='GRINDER').update(quantity=add)  
 data = dumps(demo)  
 return render(request, 'graph.html', {'data': data,'pdata':pdata})  
 else:  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 ran9 = [np.random.randint(60,90, 9)]  
 xtest3 = data.loc[  
 data['index'] == rande, ['Machine3.RawMaterial.Property1', 'Machine3.RawMaterial.Property2',  
 'Machine3.RawMaterial.Property3', 'Machine3.RawMaterial.Property4',  
 'Machine3.RawMaterialFeederParameter.U.Actual',  
 'Machine3.Zone1Temperature.C.Actual',  
 'Machine3.Zone2Temperature.C.Actual', 'Machine3.MotorRPM.C.Actual',  
 'Machine3.MaterialPressure.U.Actual']]  
 mainr = random.choice(  
 [xtest3, ran9, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3,  
 xtest3, xtest3, xtest3, xtest3, xtest3, xtest3, xtest3])  
 pred = grinder.predict(mainr)  
 if pred < 65 or pred > 80:  
 demo.append(float(pred))  
 r = random.randint(1, 20)  
 productionlist.objects.filter(machine='GRINDER').update(quantity=r)  
 data = dumps(demo)  
 pdata = productionlist.objects.filter(machine='GRINDER')  
 error = f"ERROR OCCOURED IN {tmachine[0]} MACHINE DUE TO THE VARIATION OF INPUT PARAMETERSWHERE THE ERROR IS {pred}"  
 return render(request, 'graph.html',  
 {'data': data, 'error': error, 'machine': tmachine[0], 'pdata': pdata,'name':name,'type':type})  
 else:  
 demo.append(float(pred))  
 elif tmachine[0] == 'BLENDER' or tmachine[0]=='BLENDER2' or tmachine[0]=='BLENDER3':  
 blender.fit(x4, data['Machine4.Temperature5.C.Actual'])  
 if tmachine[1] == 'changed':  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 xtest4 = data.loc[  
 data['index'] == rande, ['Machine4.Temperature1.C.Actual', 'Machine4.Temperature2.C.Actual',  
 'Machine4.Pressure.C.Actual',  
 'Machine4.Temperature3.C.Actual', 'Machine4.Temperature4.C.Actual']]  
  
 pred = blender.predict(xtest4)  
 demo.append(float(pred))  
 data = dumps(demo)  
 return render(request, 'graph.html', {'data': data})  
 else:  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 ran5 = [np.random.randint(1, 300, 5)]  
 xtest4 = data.loc[  
 data['index'] == rande, ['Machine4.Temperature1.C.Actual', 'Machine4.Temperature2.C.Actual',  
 'Machine4.Pressure.C.Actual',  
 'Machine4.Temperature3.C.Actual', 'Machine4.Temperature4.C.Actual']]  
 mainr = random.choice(  
 [xtest4, ran5, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4,  
 xtest4, xtest4, xtest4, xtest4, xtest4, xtest4, xtest4])  
 pred = blender.predict(mainr)  
 if pred < 255 or pred > 313:  
 demo.append(float(pred))  
 data = dumps(demo)  
 error = f"ERROR OCCOURED IN {tmachine[0]} MACHINE WHERE THE OUTPUT IS {pred}"  
 return render(request, 'graph.html', {'data': data, 'error': error, 'machine': tmachine[0],'name':name,'type':type})  
 else:  
 demo.append(float(pred))  
 elif tmachine[0] == 'HEATER' or tmachine[0]=='HEATER2' or tmachine[0]=='HEATER3':  
 heater.fit(x5, data['Machine5.Temperature6.C.Actual'])  
 if tmachine[1] == 'changed':  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 xtest5 = data.loc[  
 data['index'] == rande, ['Machine5.Temperature1.C.Actual', 'Machine5.Temperature2.C.Actual',  
 'Machine5.Temperature3.C.Actual',  
 'Machine5.Temperature4.C.Actual', 'Machine5.Temperature5.C.Actual',  
 'Machine5.ExitTemperature.U.Actual']]  
  
 pred = heater.predict(xtest5)  
 demo.append(float(pred))  
 data = dumps(demo)  
 return render(request, 'graph.html', {'data': data})  
 else:  
 demo = []  
 for i in range(loops):  
 rande = np.random.randint(1, (len(data) + 2))  
 ran6 = [np.random.randint(1, 300, 6)]  
 xtest5 = data.loc[  
 data['index'] == rande, ['Machine5.Temperature1.C.Actual', 'Machine5.Temperature2.C.Actual',  
 'Machine5.Temperature3.C.Actual',  
 'Machine5.Temperature4.C.Actual', 'Machine5.Temperature5.C.Actual',  
 'Machine5.ExitTemperature.U.Actual']]  
 mainr = random.choice(  
 [xtest5, ran6, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5,  
 xtest5, xtest5, xtest5, xtest5, xtest5, xtest5, xtest5])  
 pred = heater.predict(mainr)  
 if pred < 255 or pred > 313:  
 demo.append(float(pred))  
 data = dumps(demo)  
 error = f"ERROR OCCOURED IN {tmachine[0]} MACHINE WHERE THE OUTPUT IS {pred}"  
 return render(request, 'graph.html', {'data': data, 'error': error, 'machine': tmachine[0],'name':name,'type':type})  
 else:  
 demo.append(float(pred))  
 return redirect('/employee')  
def produc\_process(request,name,machine,error):  
 datatrainerview(name=name,machine=machine).save()  
 delete = errorupdate.objects.get(name=name,machines=machine,error=error)  
 delete.delete()  
 messages.success(request, f"ERRORS HAS SENT TO THE DATATRAINERS")  
 return redirect('/adminerror')  
def changemach(request):  
 if request.method=='POST':  
 name=request.POST['empname2']  
 machine=request.POST['machname2']  
 alocation.objects.filter(name=name).update(machine\_name=machine,status='changed')  
 delete = datatrainerview.objects.get(name=name)  
 delete.delete()  
 messages.success(request, f"{name} 'S MACHINE CHANGED TO {machine}")  
 return redirect('/changemach')  
  
 return render(request,'admins.html')

def production(request):  
 return render(request,"production.html")  
  
def purchaseview(request):  
 purchaselists = purchaselist.objects.all().order\_by('name')  
 dict = {}  
 for i in purchaselists:  
 dict[i.type] = int(i.quantity)  
 prodlist = productionlist.objects.all()  
 dict2 = {}  
 for j in prodlist:  
 dict2[j.machine] = (j.quantity)  
 return render(request, 'purchaseview.html',  
 {'purchaselists': purchaselists, 'scrap': dict2.get('CRUSHER'), 'sheet': dict2.get('MILLS'),  
 'rod': dict2.get('GRINDER')})  
  
def monitor(request):  
 f=alocation.objects.all()  
 error=errorupdate.objects.all()  
 g=[]  
 for i in error:  
 g.append(i.name)  
 d=[]  
 for i in f:  
 if i.name in g:  
 d.append(0)  
 else:  
 d.append(1)  
 n=zip(f,d)  
 return render(request, 'monitor.html',{'detail':n})  
  
def productionview(request):  
 pro=productionlist.objects.all()  
 return render(request,'productionview.html',{'production':pro})  
  
def paymentform(request,orderid):  
 r=purchaselist.objects.get(orderid=orderid)  
 return render(request,'paymentform.html',{'detail':r})  
  
def purchasedmateriallists(request):  
 pc=purchasedlist.objects.filter(name=request.session['cname'])  
 return render(request,'purchasedmateriallist.html',{'pur':pc})

**SYSTEM TESTING AND IMPLEMENTATION**

**9.1. 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.

**9.2. STRATEGIC APPROACH TO SOFTWARE TESTING**

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress is done by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally, we arrive at system testing, where the software and other system elements are tested as a whole.

UNIT TESTING

MODULE TESTING

SUB-SYSTEM TESING

SYSTEM TESTING

ACCEPTANCE TESTING

**Component testing**

**Integration Testing**

**User Testing**

**9.3. 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.

**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.

**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.

**3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generated on particular condition is traced to uncover any possible errors.

**4. 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.

**5. 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:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loop test the innermost loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of a connected loop.

|  |  |  |  |
| --- | --- | --- | --- |
| **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. |

**CHAPTER-10**

**SYSTEM SECURITY**

# 10.1 INTRODUCTION

Security system can be divided into four related issues: The protection of computer based resources that includes hardware, software, data, procedures and people against unauthorized use or natural.

Disaster is known as System Security.

* Security
* Integrity
* Privacy
* Confidentiality

**SYSTEM SECURITY** refers to the technical innovations and procedures applied to the hardware and operation systems to protect against deliberate or accidental damage from a defined threat.

**DATA SECURITY** is the protection of data from loss, disclosure, modification and destruction.

**SYSTEM INTEGRITY** refers to the power functioning of hardware and programs, appropriate physical security and safety against external threats such as eavesdropping and wiretapping.

**PRIVACY** defines the rights of the user or organizations to determine what information they are willing to share with or accept from others and how the organization can be protected against unwelcome, unfair or excessive dissemination of information about it.

**CONFIDENTIALITY** is a special status given to sensitive information in a database to minimize the possible invasion of privacy. It is an attribute of information that characterizes its need for protection.

## 10.2 SECURITY IN SOFTWARE

System security refers to various validations on data in the form of checks and controls to avoid the system from failing. It is always important to ensure that only valid data is entered and only valid operations are performed on the system. The system employs two types of checks and controls:

**CLIENT SIDE VALIDATION**

Various client side validations are used to ensure on the client side that only valid data is entered. Client side validation saves server time and load to handle invalid data. Some checks are imposed:

* JavaScript in used to ensure those required fields are filled with suitable data only. Maximum lengths of the fields of the forms are appropriately defined.
* Forms cannot be submitted without filling up the mandatory data so that manual mistakes of submitting empty fields that are mandatory can be sorted out at the client side to save the server time and load.
* Tab-indexes are set according to the need and taking into account the ease of use while working with the system.

**SERVER SIDE VALIDATION**

Some checks cannot be applied on the client side. Server side checks are necessary to save the system from failing and intimating the user that some invalid operation has been performed or the performed operation is restricted. Some of the server side checks imposed is:

* A server side constraint has been imposed to check for the validity of primary key and foreign key. A primary key value cannot be duplicated. Any attempt to duplicate the primary value results in a message intimating the user about those values through the forms using foreign key can be updated only of the existing foreign key values.
* The user is intimated through appropriate messages about the successful operations or exceptions occurring at server side.
* Various Access Control Mechanisms have been built so that one user may not agitate upon another. Access permissions to various types of users are controlled according to the organizational structure. Only permitted users can log on to the system and can have access according to their category. User- name, passwords and permissions are controlled the server side.
* Using server side validation, constraints on several restricted operations are imposed.

**CHAPTER-11**

**CONCLUSION**

**&**

**FUTURE ENHANCEMENT**

**CONCLUSION:**

Big data is the concept of which deals with a huge amount of data along with the data transferring at which the velocity for data speed, wherein here when the failure meets then the data of failure is visualized in the respective console for to intimate the failures to the maintainer or the trainer to change the machine or to repair the machine. In this, we have sorted the time required to find errors that occurred machine. By this production time is not prolonged for the production of material which has to deliver to the customer. So the customer will be getting the materials on time. By making a visualization platform, it can be interactive with, those who are in the monitoring section. If the error machine has been found, then the respective machine can be replaced by the same process machine.

**FUTURE WORK:**

In the future scope need to add some additional features which can assist the trainer or maintainer to reduce the effort for them. This means if failure is found in the console then it visualizes the failure of a particular machine and also it type along with that automatically change or allocate the fail occurred machine with an error-free machine this reduces the effort of the trainer or maintainer.