VOCATIONAL TRAINING PROJECT REPORT ON

EQUIPMENT MAINTAINENCE TRACKER

UNDER COMPUTERIZATION & AUTOMATION (C&A) DEPARTMENT
AT BOKARO STEEL PLANT
(STEEL AUTHORITY OF INDIA LIMITED)

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY (B.TECH)

IN COMPUTER SCIENCE AND DESIGN ENGINEERING
(SESSION: 2023–2027)

SUBMITTED BY:

ANIKET DAS

BSL URL NO.: 5914641

UNDER THE GUIDANCE OF:
Department of Computerization & Automation (C&A)
Steel Authority of India Limited (SAIL)
Bokaro Steel Plant, Jharkhand

CERTIFICATE

This is to certify that the project report entitled “EQUIPMENT MAINTENANCE TRACKER SYSTEM” submitted by ANIKET DAS (BSL URL No: 5914641), a student of B.Tech in Computer Science and Design, Batch 2023–2027, 3rd Year 5th Semester from Dr. B. C. Roy Engineering College , Durgapur in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology, is a Bonafide record of the work carried out under my guidance and supervision at SAIL Bokaro Steel Plant, Jharkhand.

Signature of Supervisor

GOURAV KUMAR
Manager (C & A)
SAIL, BOKARO STEEL PLANT

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the Steel Authority of India Limited for providing an opportunity to undergo Vocational Training at SAIL Bokaro Steel Plant, Bokaro Steel City, Jharkhand.

I am deeply indebted to my project guide Mr. GOURAV KUMAR, Manager (C & A), for his valuable guidance, constant support, and encouragement during the training period. His mentorship and insights have been instrumental in the successful completion of my project titled “EQUIPMENT MAINTENANCE TRACKER SYSTEM”.

I also extend my gratitude to all the employees of the department for their cooperation, practical demonstrations, and technical guidance throughout the training duration.

Lastly, I would like to thank the entire management and staff of SAIL Bokaro Steel Plant for their help and support, both directly and indirectly, during my internship period.

Table of Contents

1. Introduction to Bokaro Steel Plant
2. Role of Computerization and Automation in BSL
3. Objective of the Project
4. Tools and Technologies Used
5. System Design and Flowchart
6. Project Overview: Equipment Maintenance Tracker
7. Implementation Details
    - Creating app.py (Flask Application)
    - Designing index.html (User Interface)
    - Database Setup (SQLite)
    - Running the Application via CMD
8. Input and Output Screenshots
9. Sample Data and Entries
10. Results and Discussion
11. Advantages of the System
12. Limitations
13. Future Scope
14. Conclusion
15. References

1. Introduction to Bokaro Steel Plant

Located in Bokaro, Jharkhand, the Bokaro Steel Plant (BSL) is a flagship steel-manufacturing unit under Steel Authority of India Limited (SAIL). The plant was conceived in the late 1950s and formally incorporated on 29 January 1964, with construction beginning on 6 April 1968, in collaboration with the Soviet Union. Initially designed for an annual ingot capacity of 1.7 million tonnes (MT) in Stage-I and 4 MT in Stage-II, today BSL is a fully integrated facility producing a wide range of flat steel products — from hot-rolled coils and plates to cold-rolled sheets and galvanized steel, serving industries such as automotive, construction and infrastructure.

One of BSL's core strengths lies in its robust infrastructure and technological maturity. The plant features coke-oven batteries, sinter plants, blast furnaces, steel-melting shops, continuous casting machines and finishing mills — all interconnected in a seamless production chain. Continuous modernization efforts have augmented its capacity and productivity; for example, the Hot Strip Mill modernization introduced features like high-pressure de-scalers, automatic gauge control and hydraulic coilers.

From a strategic viewpoint, BSL plays a vital role in national industrial development. By leveraging the rich mineral resources of the Chota Nagpur plateau (iron ore, coal and fluxes) and adopting local manufacturing of plant equipment, BSL has been promoted as India's first "Swadeshi" steel plant with maximum indigenous content. Over the years, it has expanded capacity and stabilized operations, contributing significantly to the steel output of India and supporting allied industries such as LPG cylinders, automotive components and structural steel.

The plant is also culturally and socially anchored in its region. Situated amidst a large township and backed by extensive infrastructure, BSL is not just a manufacturing unit but a catalyst for local development and employment. Its workforce, training programs and community outreach initiatives further strengthen its status as a model industry town.

In summary, Bokaro Steel Plant stands out as an integrated, high-technology steel manufacturing complex. With its historically rooted creation, advanced infrastructure, continuous modernization and significant national impact, it forms an apt setting for your vocational training project. Your engagement with its Computerization & Automation department offers a window into modern industrial operations, automation systems and maintenance practices in a real-world steel-making environment.

Department Overview

a) Raw Materials & Material Handling Plant
   Responsible for receiving, storing and supplying raw materials (iron-ore, coal, limestone, etc.) to the various process units of BSL.

b) Coke Ovens & By-product Plant
   Converts coking coal into coke (for blast furnaces) and recovers by-products like benzene, toluene, etc.

c) Sinter Plant
   Produces sinter from ore fines, fluxes and other metallurgical wastes.

d) Blast Furnaces
   Produces molten iron (hot metal) from raw materials. Use of automation and PLC systems for charging, monitoring, etc.

e) Steel Melting Shops (SMS)
   Converts hot metal into steel via processes like LD converter, continuous casting, etc.

f) Rolling Mills
   Includes hot strip mill, cold rolling mills, finishing mills where steel sheets/coils are produced with precise tolerances and automation controls.

g) Service Departments / Auxiliary Departments
E.g., Energy Management, Water Management, Maintenance Departments (mechanical/electrical), Auxiliary Shops like Machine Shop, Foundry, etc. These provide essential support to production units.

h) Computerization & Automation (C&A) Department
The C&A department deals with automation systems (PLCs, DCS, SCADA), digital monitoring, control systems, data acquisition, process optimization and maintenance tracking. You can focus on this department in your report since your project is under C&A.

2. Role of Computerization and Automation in Bokaro Steel Plant

In the modern industrial era, the role of computerization and automation has become indispensable for achieving efficiency, accuracy, and reliability in largest scale manufacturing units. The Computerization & Automation (C&A) Department of Bokaro Steel Plant (BSL) plays a pivotal role in integrating digital technology with industrial processes to ensure uninterrupted, safe, and optimized steel production.

At Bokaro Steel Plant, automation begins right from the raw material handling system and extends up to finished product dispatch. The plant's massive scale of operation involves numerous complex and high-temperature processes that demand precise control and monitoring. The C&A department provides the backbone for these operations by deploying Programmable Logic Controllers (PLC), Distributed Control Systems (DCS), Supervisory Control and Data Acquisition (SCADA) systems, and various real-time monitoring tools. These systems collect data from thousands of sensors and instruments installed across the plant, process it in real time, and display it in control rooms for supervision and corrective action.

One of the key contributions of the C&A department is in the maintenance domain. Automated maintenance tracking systems monitor the health of critical equipment such as furnaces, pumps, compressors, and conveyors. When a fault or irregularity is detected, the system triggers alerts, thereby minimizing downtime and optimizing productivity. Your project, *Equipment Maintenance Tracker*, aligns perfectly with this principle by digitalizing maintenance operations at a smaller, demo level—mirroring what BSL implements at a larger industrial scale.
In conclusion, the Computerization & Automation Department at BSL serves as the technological nerve center of the entire plant. It not only supports the steel production process but also drives innovation, safety, sustainability, and efficiency—making BSL one of the most advanced and automated steel plants in India.

3. Objective of the Project

The main objective of the Equipment Maintenance Tracker project is to design and develop a simple, user-friendly system that helps record, monitor, and manage the maintenance activities of industrial equipment in an organized way. During the Vocational Training at SAIL Bokaro Steel Plant (C&A Department), the need for such a system was identified to reduce manual record-keeping, avoid data loss, and improve equipment reliability through timely maintenance updates.

This project aims to demonstrate how computerization and automation can simplify maintenance tracking using basic tools like Python, Flask, Notepad, and CMD. The system allows users to add equipment details, report maintenance actions, view maintenance history, and update the status of each machine. By storing all information in a centralized database, it ensures easy accessibility and better decision-making for engineers and operators.

The project also focuses on showcasing how automation can prevent equipment downtime, enhance productivity, and support preventive maintenance culture in industries. Though developed as a demo project, it reflects the real-time applications used in industrial environments like SAIL, where monitoring and timely maintenance play a crucial role in ensuring smooth plant operations.

4. Tools and Technologies Used

During the development of the Equipment Maintenance Tracker project, several basic yet powerful tools and technologies were utilized to build and run the application efficiently. Each of these tools played an important role in coding, testing, and executing the project.

1. Command Prompt (CMD)

The Command Prompt is a built-in command-line interface in Windows used to execute programs and scripts. In this project, CMD was used to run Python programs, install necessary libraries using pip, and start the local development server for testing the application. It helped in executing commands directly without needing any graphical interface.

2. Notepad

Notepad is a simple text editor used for writing and editing code. It was used to create and edit Python files (.py), HTML templates, and configuration files for the project. Being lightweight and easily accessible, Notepad served as the primary coding tool throughout the project development.

3. Python

Python is a high-level programming language known for its simplicity and readability. It was used to build the backend logic of the Equipment Maintenance Tracker. Python handled data processing, form handling, and integration with the database. Its wide range of libraries made the development faster and more efficient.

4. Flask Framework

Flask is a micro web framework written in Python. It was used to develop the web interface for the maintenance tracker. Flask allowed the creation of routes for adding, updating, and viewing maintenance data. Its lightweight structure

made it ideal for small projects like this, providing flexibility and easy deployment.

5. SQLite Database

SQLite is a lightweight, file-based database system used to store equipment details and maintenance records. It was integrated with Flask to handle all data storage and retrieval operations. Using SQLite made it possible to maintain structured information like equipment ID, name, last maintenance date, and status without needing a separate server.

5. System Design and Flowchart

The Equipment Maintenance Tracker System was designed to provide a simple, efficient, and reliable way to manage industrial equipment details and maintenance schedules. The system design ensures that data flows smoothly from user input to database storage and back to the user interface for display.

System Overview

The system is divided into three main layers:

a. User Interface (Frontend)

- The user interacts with the system through a web interface built using HTML and Flask templates.
- Users can add, update, or view equipment maintenance data.
- It is accessible through any web browser.

b. Application Logic (Backend)

- The backend, written in Python (Flask), handles requests from users.
- It processes form submissions, validates inputs, and communicates with the database.
- This layer also controls routing and ensures that correct pages are displayed to the user.

c. Database Layer

- The SQLite database stores all equipment information, including:
    - Equipment ID and name
    - Maintenance date
    - Condition/status
    - Remarks or responsible engineer
- The database allows fast retrieval and updating of data.

System Workflow

The flow of operations in the Equipment Maintenance Tracker works as follows:

I. User opens the web interface.

II. User enters details of a machine or updates existing data.

III. The data is submitted to the Flask server.

IV. Flask processes and validates the input.

V. The valid data is stored in the SQLite database.

VI. On request, data is retrieved from the database.

VII. Flask sends the data back to the frontend for display.

VIII. The user can view or edit maintenance records as needed.

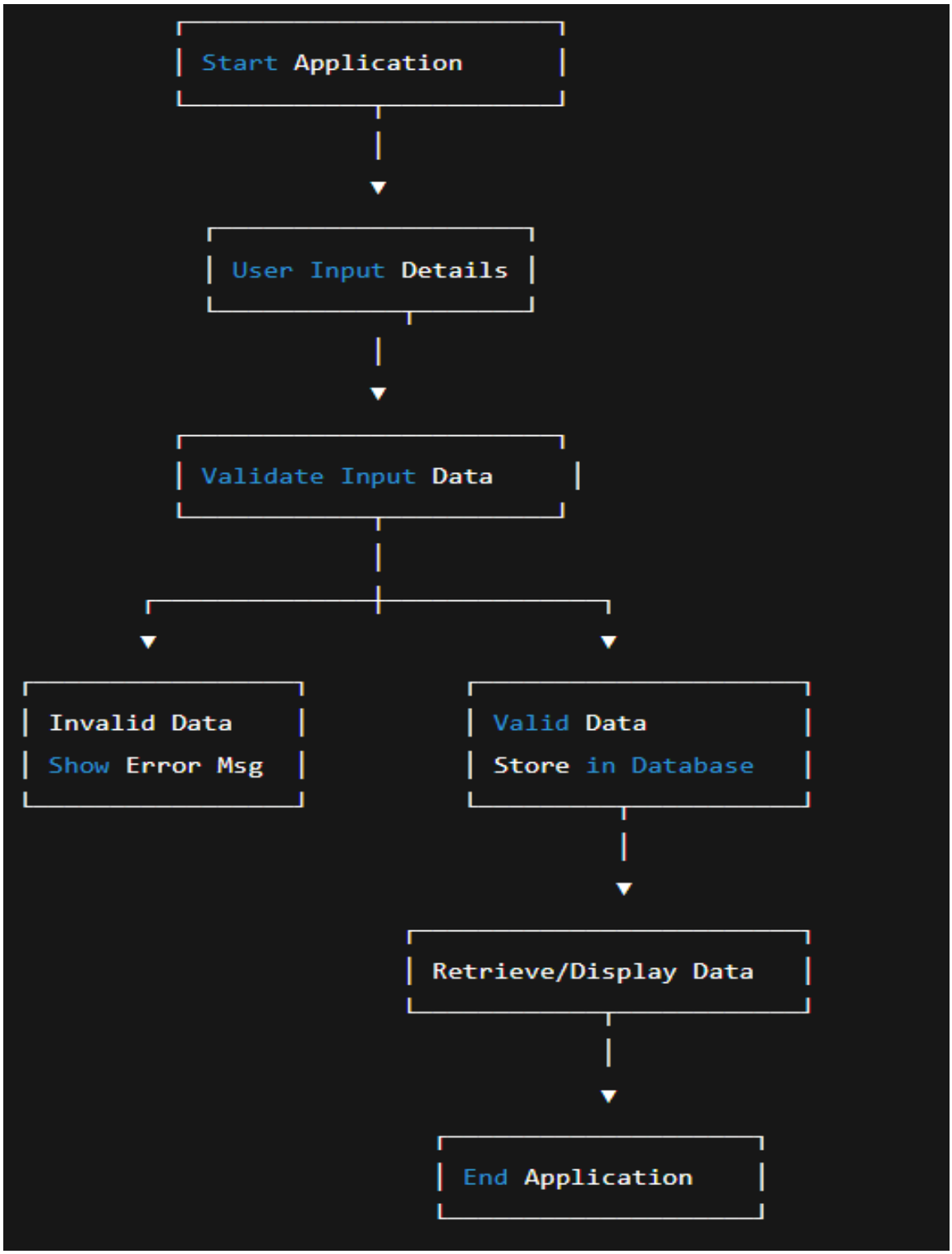
Design Principles Followed

- Simplicity: Minimal design for easy use and understanding.

  Scalability: Can be extended to more departments or integrated with IoT sensors.

- Reliability: Data stored securely and can be retrieved any time.

Flowchart :


Start Application
User Input Details
Validate Input Data
Invalid Data
Show Error Msg
Valid Data
Store in Database
Retrieve/Display Data
End Application

6. Project Overview: Equipment Maintenance Tracker

Introduction

The Equipment Maintenance Tracker is a lightweight web-based application developed to simplify the management of industrial machinery maintenance schedules at Bokaro Steel Plant.
This system was designed using Python (Flask), HTML, and SQLite. It helps track machine details such as equipment name, last maintenance date, next scheduled maintenance, and the current status of the machine.

Purpose

- The goal of this project is to:
- Maintain an organized record of each equipment's maintenance.
- Provide easy data entry, modification, and retrieval through a browser.
- Reduce manual paperwork and improve operational efficiency in the Computerization & Automation (C&A) department.

Features

- Add, view, and update machine data easily.
- Store all data in a secure local database (SQLite).
- Can be accessed through browser or local network.
- Extendable to use with IoT data (future scope).

Tools and Technologies Used

Component	Tool/Technology Used
Programming Language	Python
Framework	Flask
Database	SQLite
Editor Used	Notepad, VS Code
Execution Environment	CMD (Command Prompt)
Output Interface	Web Browser (Google Chrome)

7. Implementation Details

This section explains how the system was built step-by-step with source code and workflow.

Creating app.py (Flask Application)

The app.py file is the main backend file that connects the HTML form to the SQLite database.
It handles data submission, retrieval, and rendering of web pages.

```python
from flask import Flask, render_template, request, redirect, send_file
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
import io

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///data.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# ---------------- MODELS ----------------
class Asset(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120))
    location = db.Column(db.String(120))
    status = db.Column(db.String(50), default='Working')
    last_maintenance = db.Column(db.Date, default=datetime.utcnow)

class MaintenanceLog(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    asset_id = db.Column(db.Integer, db.ForeignKey('asset.id'))
    date = db.Column(db.Date, default=datetime.utcnow)
    notes = db.Column(db.String(200))
    cost = db.Column(db.Float, default=0.0)

# ---------------- CREATE TABLES ----------------
with app.app_context():
    db.create_all()

# ---------------- ROUTES ----------------
@app.route('/')
def index():
    assets = Asset.query.all()
    return render_template('index.html', assets=assets)

# ---------------- ADD ASSET ----------------
@app.route('/add_asset', methods=['GET', 'POST'])
def add_asset():
    if request.method == 'POST':
        name = request.form['name']
        location = request.form['location']
        status = request.form['status']  # get status from form
        asset = Asset(name=name, location=location, status=status)
        db.session.add(asset)
        db.session.commit()
        return redirect('/')
    return render_template('add_asset.html')

# ---------------- ADD LOG ----------------
@app.route('/add_log/<int:asset_id>', methods=['GET', 'POST'])
def add_log(asset_id):
    asset = Asset.query.get_or_404(asset_id)
    if request.method == 'POST':
        notes = request.form['notes']
        cost = float(request.form['cost'] or 0)
        date_str = request.form['date']
        if date_str:
            log_date = datetime.strptime(date_str, "%Y-%m-%d")
        else:
            log_date = datetime.utcnow()
        log = MaintenanceLog(asset_id=asset_id, notes=notes, cost=cost, date=log_date)
        asset.last_maintenance = log_date
        db.session.add(log)
        db.session.commit()
        return redirect('/')
    return render_template('add_log.html', asset=asset)

# ---------------- VIEW LOGS ----------------
@app.route('/view_logs/<int:asset_id>')
def view_logs(asset_id):
    asset = Asset.query.get_or_404(asset_id)
    logs = MaintenanceLog.query.filter_by(asset_id=asset_id).all()
    return render_template('index.html', assets=[asset], logs=logs, show_logs=True)

# ---------------- PDF REPORT ----------------
@app.route('/report/<int:asset_id>')
def generate_report(asset_id):
    asset = Asset.query.get_or_404(asset_id)
    logs = MaintenanceLog.query.filter_by(asset_id=asset_id).all()

    buffer = io.BytesIO()
    c = canvas.Canvas(buffer, pagesize=A4)
    width, height = A4

    c.setFont("Helvetica-Bold", 16)
    c.drawString(50, height-50, f"Maintenance Report for {asset.name}")

    c.setFont("Helvetica", 12)
    c.drawString(50, height-80, f"Location: {asset.location}")
    c.drawString(50, height-100, f"Status: {asset.status}")
    c.drawString(50, height-120, f"Last Maintenance: {asset.last_maintenance.strftime('%Y-%m-%d')}")

    y = height - 160
    c.setFont("Helvetica-Bold", 12)
    c.drawString(50, y, "Date")
    c.drawString(150, y, "Notes")
    c.drawString(400, y, "Cost")
    y -= 20

    c.setFont("Helvetica", 12)
    total_cost = 0
    for log in logs:
        if y < 50:
            c.showPage()
            y = height - 50
        c.drawString(50, y, log.date.strftime('%Y-%m-%d'))
        c.drawString(150, y, log.notes[:40])
        c.drawString(400, y, f"{log.cost}")
        total_cost += log.cost
        y -= 20

    # Show total cost at bottom
    y -= 20
    c.setFont("Helvetica-Bold", 12)
    c.drawString(50, y, f"Total Maintenance Cost: {total_cost}")

    c.save()
    buffer.seek(0)
    return send_file(buffer, as_attachment=True, download_name=f"{asset.name}_report.pdf", mimetype='application/pdf')

# ---------------- RUN APP ----------------
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Explanation:

- Flask is used to create a lightweight web server.
- The SQLite database (equipment.db) is automatically created if it doesn't exist.
- The / route shows all the data in a table.
- The /add route stores new entries submitted by the user.
- init_db() ensures the database is ready before the app starts.
- The app runs locally on port 5000 and can be accessed in any browser.

Designing index.html (User Interface)

This is the frontend file that provides the user form and displays stored data.

```
<!DOCTYPE html>
<html>
<head>
  <title>Equipment Maintenance Tracker</title>
  <style>
    body { font-family: Arial; margin: 30px; }
    table, th, td { border: 1px solid black; border-collapse: collapse; padding: 8px; }
    a { text-decoration: none; color: blue; }
    button { padding: 6px 10px; margin: 5px; }
  </style>
</head>
<body>
  <h2>Equipment Maintenance Tracker (Demo)</h2>
  <a href="/add_asset"><button>Add New Asset</button></a>
  <br><br>
  <table>
    <tr><th>ID</th><th>Name</th><th>Location</th><th>Status</th><th>Last Maintenance</th><th>Actions</th></tr>
    {% for a in assets %}
    <tr>
      <td>{{ a.id }}</td>
      <td>{{ a.name }}</td>
      <td>{{ a.location }}</td>
      <td>{{ a.status }}</td>
      <td>{{ a.last_maintenance.strftime('%Y-%m-%d') }}</td>
      <td>
        <a href="/add_log/{{ a.id }}">Add Log</a> |
        <a href="/view_logs/{{ a.id }}">View Logs</a> |
        <a href="/report/{{ a.id }}"><button>Download PDF</button></a>
      </td>
    </tr>
    {% endfor %}
  </table>

  {% if show_logs %}
    <h3>Maintenance Logs for {{ assets[0].name }}</h3>
    <table>
      <tr><th>Date</th><th>Notes</th><th>Cost</th></tr>
      {% for l in logs %}
      <tr><td>{{ l.date.strftime('%Y-%m-%d') }}</td><td>{{ l.notes }}</td><td>{{ l.cost }}</td></tr>
      {% endfor %}
    </table>
  {% endif %}
</body>
</html>
```

Explanation:

- A form allows users to add equipment details.
- All inputs are styled for a clean professional look.
- The table displays existing data fetched from the database.
- Flask's template engine (Jinja2) is used ({% for eq in equipments %}) to loop through all data dynamically.

Database Setup (SQLite)

- Database file: equipment.db
- Table: equipment
- Fields: id, name, department, maintenance_date, status, remarks

SQLite database is lightweight and needs no external installation. It's automatically created when you first run the app.

Running the Application via CMD

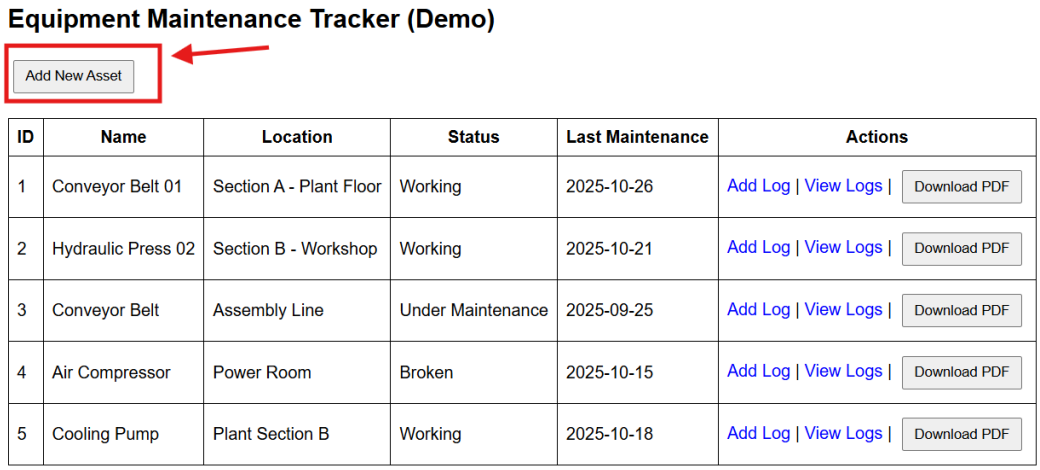
Steps:

1. Open CMD.
2.
3. Navigate to the project folder:

```
Microsoft Windows [Version 10.0.22631.5909]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Prime>D:

D:\>cd "D:\Equipment Maintainance Tracker"

D:\Equipment Maintainance Tracker>venv\Scripts\activate

(venv) D:\Equipment Maintainance Tracker> pip install -r requirements.txt
Requirement already satisfied: Flask==2.3.3 in d:\equipment maintainance tracker\venv\lib\site-packages (from -r requirements.txt (line 1)) (2.3.3)
Requirement already satisfied: Flask-SQLAlchemy==3.0.3 in d:\equipment maintainance tracker\venv\lib\site-packages (from -r requirements.txt (line 2)) (3.0.3)
Requirement already satisfied: reportlab==4.0.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from -r requirements.txt (line 3)) (4.0.0)
Requirement already satisfied: Werkzeug>=2.3.7 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask==2.3.3->-r requirements.txt (line 1)) (3.1.3)
Requirement already satisfied: Jinja2>=3.1.2 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask==2.3.3->-r requirements.txt (line 1)) (3.1.6)
Requirement already satisfied: itsdangerous>=2.1.2 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask==2.3.3->-r requirements.txt (line 1)) (2.2.0)
Requirement already satisfied: click>=8.1.3 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask==2.3.3->-r requirements.txt (line 1)) (8.3.0)
Requirement already satisfied: blinker>=1.6.2 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask==2.3.3->-r requirements.txt (line 1)) (1.9.0)
Requirement already satisfied: SQLAlchemy>=1.4.18 in d:\equipment maintainance tracker\venv\lib\site-packages (from Flask-SQLAlchemy==3.0.3->-r requirements.txt (line 2)) (2.0.44)
Requirement already satisfied: pillow>=9.0.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from reportlab==4.0.0->-r requirements.txt (line 3)) (12.0.0)
Requirement already satisfied: rlPyCairo<1,>=0.2.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from reportlab==4.0.0->-r requirements.txt (line 3)) (0.4.0)
Requirement already satisfied: freetype-py<2.4,>=2.3.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from reportlab==4.0.0->-r requirements.txt (line 3)) (2.3.0)
Requirement already satisfied: pycairo>=1.20.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from rlPyCairo<1,>=0.2.0->reportlab==4.0.0->-r requirements.txt (line 3)) (1.28.0)
Requirement already satisfied: colorama in d:\equipment maintainance tracker\venv\lib\site-packages (from click>=8.1.3->Flask==2.3.3->-r requirements.txt (line 1)) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from Jinja2>=3.1.2->Flask==2.3.3->-r requirements.txt (line 1)) (3.0.3)
Requirement already satisfied: greenlet>=1 in d:\equipment maintainance tracker\venv\lib\site-packages (from SQLAlchemy>=1.4.18->Flask-SQLAlchemy==3.0.3->-r requirements.txt (line 2)) (3.2.4)
Requirement already satisfied: typing-extensions>=4.6.0 in d:\equipment maintainance tracker\venv\lib\site-packages (from SQLAlchemy>=1.4.18->Flask-SQLAlchemy==3.0.3->-r requirements.txt (line 2)) (4.15.0)

[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) D:\Equipment Maintainance Tracker> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.2.121:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 769-626-574
```

8. Input and Output

Step 1 : Adding an asset .

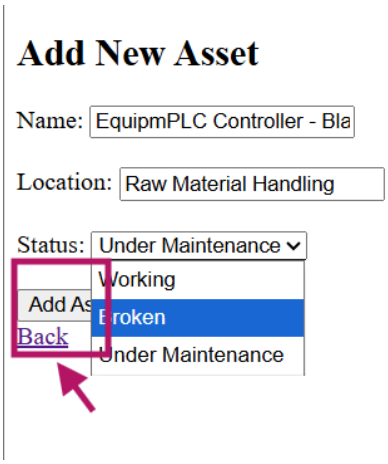


Equipment Maintenance Tracker (Demo)

Add New Asset

ID	Name	Location	Status	Last Maintenance	Actions
1	Conveyor Belt 01	Section A - Plant Floor	Working	2025-10-26	Add Log | View Logs | Download PDF
2	Hydraulic Press 02	Section B - Workshop	Working	2025-10-21	Add Log | View Logs | Download PDF
3	Conveyor Belt	Assembly Line	Under Maintenance	2025-09-25	Add Log | View Logs | Download PDF
4	Air Compressor	Power Room	Broken	2025-10-15	Add Log | View Logs | Download PDF
5	Cooling Pump	Plant Section B	Working	2025-10-18	Add Log | View Logs | Download PDF

Step 2 : Input entries for Equipment Maintenance Tracker and press add Asset.



Add New Asset

Name: EquipmPLC Controller - Bla

Location: Raw Material Handling

Status: Under Maintenance

Working

Broken

Under Maintenance

Add As

Back

Step 3 : by clicking on add log you can simply add remarks and maintainence date about the asset and by view log you can just view it.

ID	Name	Location	Status	Last Maintenance	Actions
1	Conveyor Belt 01	Section A - Plant Floor	Working	2025-10-26	Add Log | View Logs | Download PDF
2	Hydraulic Press 02	Section B - Workshop	Working	2025-10-21	Add Log | View Logs | Download PDF
3	Conveyor Belt	Assembly Line	Under Maintenance	2025-09-25	Add Log | View Logs | Download PDF
4	Air Compressor	Power Room	Broken	2025-10-15	Add Log | View Logs | Download PDF
5	Cooling Pump	Plant Section B	Working	2025-10-18	Add Log | View Logs | Download PDF
6	EquipmPLC Controller - Blast Furnace 2	Raw Material Handling	Under Maintenance	2025-10-28	Add Log | View Logs | Download PDF

Step 4 : adding revelant information in log for that particular asset and click save.

Add Maintenance Log for EquipmPLC Controller - Blast Furnace 2

Notes:

Blade vibration under observation

Cost: 4500

Date: 15-10-2025

Save

Back

October, 2025

Su	Mo	Tu	We	Th	Fr	Sa
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Clear Today

Step 5 : Download the PDF Maintainence report of the file for further use.

Maintenance Report for EquipmPLC Controller - Blast Furnace 2

Location: Raw Material Handling

Status: Under Maintenance

Last Maintenance: 2025-10-15

Date	Notes	Cost
2025-10-15	Blade vibration under observation	4500.0

Total Maintenance Cost: 4500.0

9. Advantages and Limitations

Category	Details
User-Friendly Interface	The web interface is simple and easy to operate. Even non-technical staff can manage equipment data efficiently.
Paperless Maintenance Tracking	All records are stored digitally, eliminating the need for paper-based logbooks and reducing errors.
Quick Access to Data	Maintenance history and equipment status can be retrieved instantly, improving decision-making and workflow.
Cost-Effective Solution	Developed entirely with open-source tools like Python, Flask, and SQLite, requiring no commercial licenses.
Data Consistency & Security	The use of a structured database ensures data accuracy and prevents loss or duplication.
Customizable Design	The project can be easily extended with features like report generation or IoT integration.
Lightweight & Portable	Runs smoothly on basic systems without heavy configuration or internet requirements.
Supports Real-Time Monitoring (Future Ready)	Can later integrate with IoT devices for live equipment tracking and predictive maintenance.

Category	Limitations
Single-User Access	The current version supports only one user at a time, with no multi-user or login feature.
Limited to Localhost	Works only on the local system; remote access requires additional tools like Ngrok for temporary hosting.
No Automated Alerts	The system does not yet send notifications or reminders for upcoming maintenance.
Manual Data Entry	Equipment details must be entered manually, which may lead to occasional data entry errors.
No Advanced Analytics	The system currently lacks graphs, reports, or predictive maintenance analytics.
Limited Mobile Support	The user interface is not fully optimized for mobile or tablet devices.

10. Future Scope

The *Equipment Maintenance Tracker* project has strong potential for future development and industrial-scale deployment. As SAIL Bokaro continues to expand its automation and digital infrastructure, this system can be enhanced with advanced technologies to achieve complete smart maintenance management.

The following improvements can be considered for future upgrades:

Future Enhancement	Description
1. Multi-User Authentication System	Introduce secure login features for different user roles — such as Engineers, Managers, and Technicians — to ensure data privacy and controlled access.
2. Cloud Integration	Deploy the system on cloud platforms (AWS, Azure, or Google Cloud) for remote access, centralized data storage, and cross-department sharing.
3. IoT-Based Equipment Monitoring	Integrate Internet of Things (IoT) sensors with the system to automatically capture real-time data like machine temperature, vibration, and run-time hours.

Future Enhancement	Description
4. Predictive Maintenance with AI	Implement AI and Machine Learning models to predict equipment failures in advance and schedule maintenance proactively.
5. Email/SMS Notifications	Add automatic alerts to notify staff of upcoming maintenance or overdue tasks, improving reliability and accountability.
6. Mobile and Web Dashboard	Develop a responsive web and mobile interface for easy access to data anytime, anywhere.
7. Data Analytics and Reports	Integrate dashboards for visualizing performance trends, cost analysis, and maintenance frequency reports.
8. Backup and Recovery System	Enable automatic data backup and recovery mechanisms to protect against system failures or data loss.

11. Conclusion

The *Equipment Maintenance Tracker* project successfully demonstrates how computerization and automation can simplify industrial maintenance processes within **Bokaro Steel Plant's C&A Department**. By developing this system using basic tools such as **Python (Flask framework)**, **SQLite database**, and **HTML interface**, an efficient and user-friendly platform was created for managing equipment records, maintenance schedules, and operational statuses.

Through this project, it became clear how digital transformation helps minimize manual errors, improve efficiency, and ensure timely maintenance of critical machinery. The system's ability to record equipment data, track maintenance history, and provide status updates can significantly enhance productivity and reduce downtime.

Moreover, the project experience provided valuable practical exposure to **web development**, **database management**, and **automation concepts** in a real industrial environment. It also emphasized the importance of integrating IT solutions with industrial operations to achieve greater reliability and performance.

In the future, by adding advanced technologies such as **AI-based predictive maintenance**, **IoT integration**, and **cloud access**, the project can evolve into a smart maintenance ecosystem, contributing to **SAIL's goal of digital modernization and industrial excellence.**

12. References

- **Bokaro Steel Plant (SAIL) Official Website** – https://sail.co.in/bokaro-steel-plant
(For information on plant overview, departments, and digital initiatives.)

- **Flask Documentation** – https://flask.palletsprojects.com/
(For understanding Python Flask framework and web development concepts.)

- **SQLite Documentation** – https://www.sqlite.org/docs.html
(For database integration and management techniques.)

- **GitHub Open Source Repositories** –
(For reference to open-source Flask projects and structure understanding.)

- **SAIL HRD VT Training Materials** –
(For guidance and understanding of real-time plant automation and computerization concepts.)