SAIL Vocational Training
Project Report
On
CSV Comparator

Implemented for developing a Python-Automation Tool for Comparing two CSV files. By

AKSHAT KUMAR SINGH (5913836)
B. Tech Computer Science & Engineering
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Submitted to
HRD, SAIL

Guided by

Mr. Chandan Kumar
Asstt. General Manager, C & IT

Steel Authority of India, Bokaro Steel City

स्टी अथॉर्ट्रि ऑफ इंडिया डीड्ि
STEEL AUTHORITY OF INDIA LIMITED

INDEX

SL NO.	Contents
1.	Introduction To The Steel Industry
2.	Overview Role Of IT In Steel Industry
3.	Abstract & Problem Statements
4.	Tools and Technologies Used
5.	Dataset
6.	Code Snippets
7.	Output
8.	Challenges Faced
9.	Conclusion & Future Work
10.	References

INTRDOUCTION

The **steel industry** plays a crucial role in the economic development of any nation. It is often referred to as the **backbone of modern industrial society** due to its widespread applications across infrastructure, construction, transportation, manufacturing, and defense sectors.

Steel is an alloy primarily composed of iron and carbon and is valued for its **high strength, durability, and versatility**. The industry includes a wide range of processes—from mining and refining raw materials, to producing finished steel products in integrated steel plants or mini-mills.

Global Significance:

- The steel industry is one of the **largest and most essential global industries**.
- Countries like **China, India, Japan, the U.S., and Russia** are among the top steel producers.
- The demand for steel continues to grow with urbanization, industrialization, and infrastructural development.

Steel Industry in India:

- India is currently the **second-largest producer of crude steel** in the world.
- The Indian steel sector is dominated by **public sector undertakings (PSUs)** such as **SAIL (Steel Authority of India Limited)** and **private players** like **Tata Steel** and **JSW Steel**.
- The industry supports over **2 million jobs** directly and millions more indirectly through associated industries.

Importance of Steel:

- **Construction**: Bridges, buildings, pipelines, railways, and roads.
- **Automotive**: Manufacturing of cars, trucks, and heavy vehicles.
- **Defense and Aerospace**: High-grade steels are used in defense equipment and aircraft.
- **Energy**: Used in the production and transport of oil, gas, and electricity.

Current Trends:

- **Emphasis on green steel and environmental sustainability.**
- Adoption of **Industry 4.0 technologies** like automation, data analytics, and machine learning.
- Focus on increasing **energy efficiency** and reducing **carbon emissions**.

BOKARO STEEL PLANT(BSL):

The **Bokaro Steel Plant (BSL)** is the **fourth integrated steel plant** in the public sector, established with assistance from the **Soviet Union**. It was initially incorporated as a limited company on **29 January 1964**, and later merged with **Steel Authority of India Limited (SAIL)**—first as a subsidiary and subsequently as a unit—under the **Restructuring & Miscellaneous Provisions Act, 1978**.

Construction officially commenced on **6 April 1968**. The plant currently operates **five blast furnaces** with a **liquid steel production capacity of 5.8 million tonnes (MT)**. Continuous modernization is underway, and production is expected to surpass **10 MT** in the near future.

During the 1990s, BSL underwent significant industrial development, including upgrades to its **refining units** and installation of **continuous casting machines**. More recently, the plant initiated an **expansion program in collaboration with POSCO (Pohang Iron and Steel Company)** of South Korea.

For the establishment of the plant, approximately **64 moujas** (land units, each potentially including multiple villages) were acquired. Of the total land acquired, **7,765 hectares** were utilized for the steel plant infrastructure. Some parts of the land were later allocated by SAIL to private entities, allegedly without formal government approval.

In terms of financial performance:

- In **FY 2003–04**, the plant recorded a revenue of **₹11.2 billion** (approx. **USD 150 million**).
- By **FY 2007–08**, revenue had increased to **₹84.26 billion**.

In the **financial year 2020–21**, under the leadership of Managing Director **Mr. Amarendu Prakash**, the Bokaro Steel Plant surpassed its production and financial targets. It reported a **Profit Before Tax (PBT)** of **₹2,251 crore**, a remarkable rise compared to the **₹48 crore PBT** during the **corresponding period last year (CPLY)**.

Social and Economic Impact:

- BSL has been a **major driver of employment**, industrial growth, and economic development in eastern India.
- The township around the plant includes schools, hospitals, recreational facilities, and housing for workers and their families.
- It contributes significantly to the local economy and plays a role in national steel supply chains.

Overview Role Of IT In Steel Industry

The **Information Technology (IT) sector** has become a transformative force in the **steel industry**, reshaping traditional manufacturing processes, improving efficiency, reducing costs,

and enhancing decision-making. In an era of digital transformation and Industry 4.0, IT acts as a **critical enabler of innovation** in steel production, logistics, quality control, and workforce management.

1. Automation and Process Control

- **Supervisory Control and Data Acquisition (SCADA)** and **Distributed Control Systems (DCS)** automate and monitor critical operations like blast furnace temperatures, rolling mill speeds, and cooling systems.
- These systems ensure **real-time process optimization**, reduce human errors, and improve safety and reliability.

2. Enterprise Resource Planning (ERP) Systems

- Steel plants like **SAIL, Tata Steel, and JSW** rely on ERP platforms (e.g., SAP, Oracle) for **integrated management** of supply chain, inventory, procurement, maintenance, finance, and HR.
- ERP streamlines business operations by providing **centralized access** to real-time data and performance metrics.

3. Predictive Maintenance

- IT enables **condition monitoring** of machines and equipment using IoT sensors and AI algorithms.
- Predictive maintenance minimizes **unplanned downtime**, extends asset lifespan, and reduces operational costs.

4. Quality Assurance and Data Analytics

- **Computer Vision and AI-based systems** are used to inspect steel surfaces and detect flaws in real time.
- Data analytics helps in analyzing product quality, performance trends, and customer feedback to drive **continuous improvement**.

5. Supply Chain and Logistics Optimization

- IT tools are used for **supply chain visibility**, **fleet tracking**, **inventory planning**, and **order fulfillment**.
- Integration of **RFID**, **GPS**, and **warehouse management systems** ensures timely delivery and reduces waste.

6. Energy Management and Sustainability

- IT helps monitor and optimize **energy consumption**, especially in energy-intensive units like furnaces and converters.

- Smart energy systems support **carbon footprint reduction** and ensure compliance with environmental norms.

7. Cybersecurity and Data Protection

- Steel companies are increasingly investing in **cybersecurity infrastructure** to protect industrial control systems (ICS) from cyberattacks.
- Security protocols and IT audits help prevent **data leaks**, **sabotage**, and **ransomware threats**.

8. Human Resource Management

- IT facilitates **digital attendance systems**, **resume screening tools** (like the ML-based one in your project), and **employee training portals**.
- Smart dashboards and AI can help HR teams **match candidates with job descriptions**, monitor performance, and plan training programs.

9. Research, Simulation, and Digital Twin

- IT allows simulation of production processes using **digital twin technology** to test changes virtually before real-world implementation.
- Helps in **cost analysis, layout planning**, and **equipment testing** without actual resource use.

10. Role in Industry 4.0 Integration

- **Steel plants are adopting smart factory principles, leveraging AI, ML, Big Data, IoT, and Cloud Computing.**
- Enables **real-time decision-making**, **remote monitoring**, and **end-to-end automation** of manufacturing and business processes.

Abstract & Problem Statements

Abstracts:

The **CSV Comparator GUI Tool** is a Python-based application developed during my internship at SAIL to automate payroll record audits by comparing master and slave employee datasets (CSV files). It identifies **discrepancies in employee details** (IFSC, DOJ, DOB) and detects **new joiners**, generating encrypted PDF reports for secure HR documentation. The tool features an intuitive Tkinter GUI with Light/Dark modes, searchable tables, and email integration to streamline compliance workflows. Built with **Pandas (data analysis)**, **ReportLab (PDF generation)**, and **PyPDF2 (password protection)**, it reduces manual verification errors while ensuring data security.

Problem Statement:

SAIL's payroll team manually compared monthly CSV files (10,000+ records) to detect discrepancies in employee details—a time-consuming (8+ hours/month) and error-prone process. No automated system existed to:

1. Flag modified fields (e.g., updated IFSC codes)
2. Track new joiners efficiently
3. Generate shareable audit reports with data protection
4. Email reports directly to HR/Finance teams

The CSV Comparator solves these challenges by automating comparisons, securing PDFs with passwords, and integrating email—reducing processing time by 90% while improving accuracy.

Tools and Technologies Used

1. Python: A versatile programming language used for developing the application, known for its readability and extensive libraries.
2. Tkinter: The standard GUI toolkit for Python, used to create the application's user interface, allowing for interactive file selection and data display.
3. Pandas: A powerful data manipulation library in Python, utilized for reading, processing, and comparing CSV files to identify discrepancies in employee records.
4. ReportLab: A library for generating PDFs in Python, used to create well-structured and formatted reports from the comparison results.
5. PyPDF2: A library for manipulating PDF files, employed to add password protection to the generated reports, ensuring sensitive data is secure.
6. smtplib: A built-in Python library for sending emails, used to facilitate the direct emailing of generated PDF reports to specified recipients.
7. ttk.Treeview: A widget from the Tkinter library that provides a table-like interface for displaying data, allowing users to search and filter results interactively.

8. ttk.Style: A module in Tkinter that enables customization of the application's appearance, allowing users to toggle between Light and Dark modes for improved usability.

Dataset:

The dataset used in the CSV Comparator project consists of two primary CSV files: the **Master** and the **Slave** datasets, each containing detailed employee information. The Master dataset serves as the authoritative source, encompassing essential fields such as **SAIL_PERNO** (employee number), **UNIT_PERNO** (unit number), **IFSC** (Indian Financial System Code), **DOJ** (Date of Joining), **DOB** (Date of Birth), and other relevant employee details. The Slave dataset represents the updated payroll records for a specific period, which may include modifications to existing records and new joiners not present in the Master dataset. By comparing these two datasets, the tool identifies discrepancies in employee details, flags modified records, and isolates new joiners, thereby facilitating accurate payroll audits and ensuring data integrity for HR management. The datasets are structured to allow for efficient processing and analysis using the Pandas library, enabling seamless integration into the application's comparison logic.

Code

```
import tkinter as tk
from tkinter import ttk, filedialog, messagebox, simpledialog
from tkinter.ttk import Style
import pandas as pd
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph
from reportlab.lib.pagesizes import letter
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet
import os
import smtplib
from email.message import EmailMessage
import PyPDF2  # Import PyPDF2 for PDF encryption

KEYS = ['SAIL_PERNO', 'UNIT_PERNO']
EXCLUDE_COLUMNS = ['YYYYMM']
NEW_JOINEE_FIELDS = ['DOJ_SAIL', 'DOB', 'IFSC_CD', 'DOA']

SENDER_EMAIL = ""
SENDGRID_API_KEY = ""

class CSVComparatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("CSV Comparator and New Joinees")
        self.root.geometry("1100x750")
        self.dark_mode = False

        self.master_file = ""
        self.slave_file = ""
        self.data = {'diff': [], 'new': []}
        self.diff_pdf_path = ""
        self.new_pdf_path = ""

        self.setup_style()
        self.create_widgets()

    def setup_style(self):
        self.style = Style()
        self.apply_light_mode()

    def apply_light_mode(self):
```

```
        self.style.theme_use("clam")
        self.style.configure("TFrame", background="#f4f4f4")
        self.style.configure("TLabel", font=("Segoe UI", 10), background="#f4f4f4")
        self.style.configure("TButton", font=("Segoe UI", 10), padding=6)
        self.style.configure("TNotebook", background="#ffffff")
        self.style.configure("TNotebook.Tab", padding=[10, 5], font=("Segoe UI", 10))
        self.style.configure("Treeview", font=("Segoe UI", 10), rowheight=25)
        self.style.configure("Treeview.Heading", font=("Segoe UI", 10, "bold"))
        self.root.configure(bg="#f4f4f4")

    def apply_dark_mode(self):
        self.style.theme_use("clam")
        self.style.configure("TFrame", background="#2e2e2e")
        self.style.configure("TLabel", font=("Segoe UI", 10), background="#2e2e2e",
foreground="#ffffff")
        self.style.configure("TButton", font=("Segoe UI", 10), padding=6)
        self.style.configure("TNotebook", background="#3c3f41")
        self.style.configure("TNotebook.Tab", padding=[10, 5], font=("Segoe UI", 10))
        self.style.configure("Treeview", font=("Segoe UI", 10), rowheight=25,
background="#3c3f41", foreground="#ffffff", fieldbackground="#3c3f41")
        self.style.configure("Treeview.Heading", font=("Segoe UI", 10, "bold"),
background="#2e2e2e", foreground="#ffffff")
        self.root.configure(bg="#2e2e2e")

    def toggle_dark_mode(self):
        if self.dark_mode:
            self.apply_light_mode()
            self.dark_mode = False
        else:
            self.apply_dark_mode()
            self.dark_mode = True

    def create_widgets(self):
        frm_top = ttk.Frame(self.root, padding=10)
        frm_top.pack(pady=10, fill='x')

        title = ttk.Label(frm_top, text="CSV Comparator and New Joinees", font=("Segoe UI",
16, "bold"))
        title.grid(row=0, column=0, columnspan=7, pady=(0, 20), sticky="w")

        ttk.Label(frm_top, text="Master CSV:").grid(row=1, column=0, sticky='e')
        self.master_entry = ttk.Entry(frm_top, width=60)
        self.master_entry.grid(row=1, column=1)
```

```
        ttk.Button(frm_top, text="Browse", command=self.browse_master).grid(row=1,
column=2, padx=5)

        ttk.Label(frm_top, text="Changes CSV:").grid(row=2, column=0, sticky='e')
        self.slave_entry = ttk.Entry(frm_top, width=60)
        self.slave_entry.grid(row=2, column=1)
        ttk.Button(frm_top, text="Browse", command=self.browse_slave).grid(row=2,
column=2, padx=5)

        btn_frame = ttk.Frame(frm_top)
        btn_frame.grid(row=3, column=0, columnspan=7, pady=10)
        ttk.Button(btn_frame, text="Compare", command=self.compare_csvs).pack(side='left',
padx=5)
        ttk.Button(btn_frame, text="Export Differences", command=lambda:
self.export_to_pdf('diff')).pack(side='left', padx=5)
        ttk.Button(btn_frame, text="Export New Joinees", command=lambda:
self.export_to_pdf('new')).pack(side='left', padx=5)
        ttk.Button(btn_frame, text="Toggle Dark Mode",
command=self.toggle_dark_mode).pack(side='left', padx=5)
        ttk.Button(btn_frame, text="Email Differences", command=lambda:
self.send_email('diff')).pack(side='left', padx=5)
        ttk.Button(btn_frame, text="Email New Joinees", command=lambda:
self.send_email('new')).pack(side='left', padx=5)

        self.status = tk.Label(self.root, text="Welcome to CSV Comparator", anchor='w',
bg="#e6e6e6", relief='sunken')
        self.status.pack(fill='x', side='bottom')

        self.notebook = ttk.Notebook(self.root)
        self.notebook.pack(expand=1, fill='both', padx=10, pady=10)

        self.tabs = {}
        self.create_table_tabs()

    def create_table_tabs(self):
        self.tables = {}
        self.search_entries = {}

        for key in ['diff', 'new']:
            tab = ttk.Frame(self.notebook)
            self.tabs[key] = tab
            self.notebook.add(tab, text=f"{key.title()} (0)")
```

```
        top = ttk.Frame(tab, padding=10)
        top.pack(fill='x')

        ttk.Label(top, text="Search:").pack(side='left')
        search_entry = ttk.Entry(top)
        search_entry.pack(side='left', fill='x', expand=True, padx=(5, 10))
        search_entry.bind('<KeyRelease>', lambda e, k=key: self.update_table_filter(k))
        self.search_entries[key] = search_entry

        tree_frame = ttk.Frame(tab)
        tree_frame.pack(fill='both', expand=True, padx=10, pady=5)

        cols = ("Employee Key", "Field", "Old Value", "New Value")
        tree = ttk.Treeview(tree_frame, columns=cols, show="headings")

        vsb = ttk.Scrollbar(tree_frame, orient="vertical", command=tree.yview)
        hsb = ttk.Scrollbar(tree_frame, orient="horizontal", command=tree.xview)
        tree.configure(yscroll=vsb.set, xscroll=hsb.set)

        tree.grid(row=0, column=0, sticky='nsew')
        vsb.grid(row=0, column=1, sticky='ns')
        hsb.grid(row=1, column=0, sticky='ew')

        tree_frame.grid_rowconfigure(0, weight=1)
        tree_frame.grid_columnconfigure(0, weight=1)

        for col in cols:
            tree.heading(col, text=col)
            tree.column(col, anchor='w', width=150)

        style = ttk.Style()
        style.configure("Treeview", bordercolor="black", borderwidth=1)
        style.configure("Treeview.Heading", bordercolor="black", borderwidth=1)

        self.tables[key] = tree

    def browse_master(self):
        file = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")])
        if file:
            self.master_entry.delete(0, tk.END)
            self.master_entry.insert(0, file)
            self.status.config(text=f"Selected Master: {os.path.basename(file)}")
```

```
    def browse_slave(self):
        file = filedialog.askopenfilename(filetypes=[("CSV Files", "*.csv")])
        if file:
            self.slave_entry.delete(0, tk.END)
            self.slave_entry.insert(0, file)
            self.status.config(text=f"Selected Slave: {os.path.basename(file)}")

    def compare_csvs(self):
        self.master_file = self.master_entry.get()
        self.slave_file = self.slave_entry.get()
        if not self.master_file or not self.slave_file:
            messagebox.showwarning("Warning", "Please select both Master and Slave CSV
files.")
            return

        try:
            df1 = pd.read_csv(self.master_file, dtype=str).fillna('')
            df2 = pd.read_csv(self.slave_file, dtype=str).fillna('')
        except Exception as e:
            messagebox.showerror("Error", f"Failed to read CSV files: {e}")
            return

        df1.set_index(KEYS, inplace=True)
        df2.set_index(KEYS, inplace=True)

        self.data = {'diff': [], 'new': []}
        matched_keys = df1.index.intersection(df2.index)

        for key in matched_keys:
            row1 = df1.loc[key]
            row2 = df2.loc[key]
            differences = []
            for col in df1.columns:
                if col in EXCLUDE_COLUMNS:
                    continue
                val1 = str(row1[col]) if col in row1 else ''
                val2 = str(row2[col]) if col in row2 else ''
                if val1 != val2:
                    differences.append((col, val1, val2))
            if differences:
                emp_key = key[0]
                self.data['diff'].append({'key': emp_key, 'field': '', 'old': '', 'new': ''})
                for col, val1, val2 in differences:
```

```
                self.data['diff'].append({'key': emp_key, 'field': col, 'old': val1, 'new': val2})

        new_keys = df2.index.difference(df1.index)
        for key in new_keys:
            emp_key = key[0]
            self.data['new'].append({'key': emp_key, 'field': '', 'old': '', 'new': ''})
            for col in NEW_JOINEE_FIELDS:
                val = df2.loc[key][col] if col in df2.columns and col in df2.loc[key] else ''
                self.data['new'].append({'key': emp_key, 'field': col, 'old': '', 'new': val})

        # Generate PDFs automatically during comparison
        try:
            self.diff_pdf_path = self.export_to_pdf('diff', return_path=True)
            self.new_pdf_path = self.export_to_pdf('new', return_path=True)
            self.status.config(text="Comparison and PDF generation completed successfully.")
        except Exception as e:
            self.status.config(text=f"Comparison completed but PDF generation failed: {str(e)}")

        self.update_tabs()

    def update_tabs(self):
        for tab_key, tree in self.tables.items():
            tree.delete(*tree.get_children())
            last_key = None
            for row in self.data[tab_key]:
                values = (row['key'], row['field'], row['old'], row['new'])
                tree.insert('', 'end', values=values)
                if last_key is not None and row['key'] != last_key:
                    sep_id = tree.insert('', 'end', values=('', '', '', ''))
                    tree.item(sep_id, tags=('separator',))
                last_key = row['key']

            tree.tag_configure('separator', background='black')

            if tab_key == 'diff':
                count = sum(1 for row in self.data[tab_key] if row['field'] == '' and row['key'])
            else:
                count = len(self.data[tab_key]) // (len(NEW_JOINEE_FIELDS) + 1)
            self.notebook.tab(self.tabs[tab_key], text=f"{tab_key.title()} ({count})")

    def update_table_filter(self, tab_key):
        query = self.search_entries[tab_key].get().lower()
        tree = self.tables[tab_key]
```

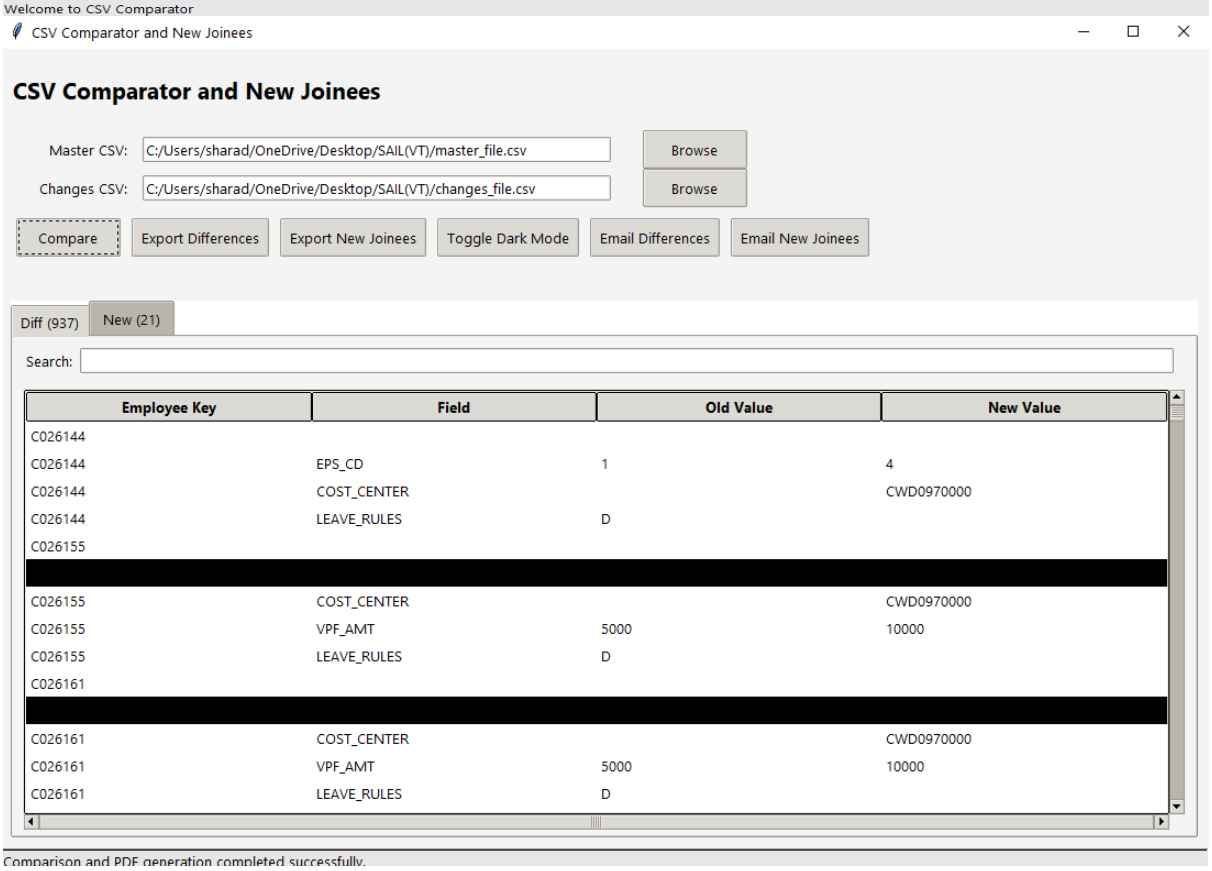
```
        tree.delete(*tree.get_children())
        last_key = None
        for row in self.data[tab_key]:
            if any(query in str(v).lower() for v in row.values()):
                values = (row['key'], row['field'], row['old'], row['new'])
                tree.insert('', 'end', values=values)
                if last_key is not None and row['key'] != last_key:
                    sep_id = tree.insert('', 'end', values=('', '', '', ''))
                    tree.item(sep_id, tags=('separator',))
                last_key = row['key']
        tree.tag_configure('separator', background='black')

    def export_to_pdf(self, tab_key, return_path=False):
        file_path = f"{tab_key}_report.pdf"  # Default filename in current directory
        if not return_path:
            file_path = filedialog.asksaveasfilename(
                defaultextension=".pdf",
                filetypes=[("PDF files", "*.pdf")],
                title=f"Save {tab_key.title()} Report",
                initialfile=f"{tab_key}_report.pdf"
            )
            if not file_path:
                return None

        # Ask for a password to encrypt the PDF
        password = simpledialog.askstring("Password", "Set a password to open the PDF:",
show='*')
        if not password:
            return None

        try:
            # Create the PDF with reportlab
            doc = SimpleDocTemplate(file_path, pagesize=letter)
            elements = []
            styles = getSampleStyleSheet()
            elements.append(Paragraph(f"{tab_key.title()} Report", styles['Title']))

            data = [["Employee Key", "Field", "Old Value", "New Value"]]
            prev_key = None
            for row in self.data[tab_key]:
                data.append([row['key'], row['field'], row['old'], row['new']])
                if prev_key is not None and row['key'] != prev_key:
                    data.append(['', '', '', ''])
```

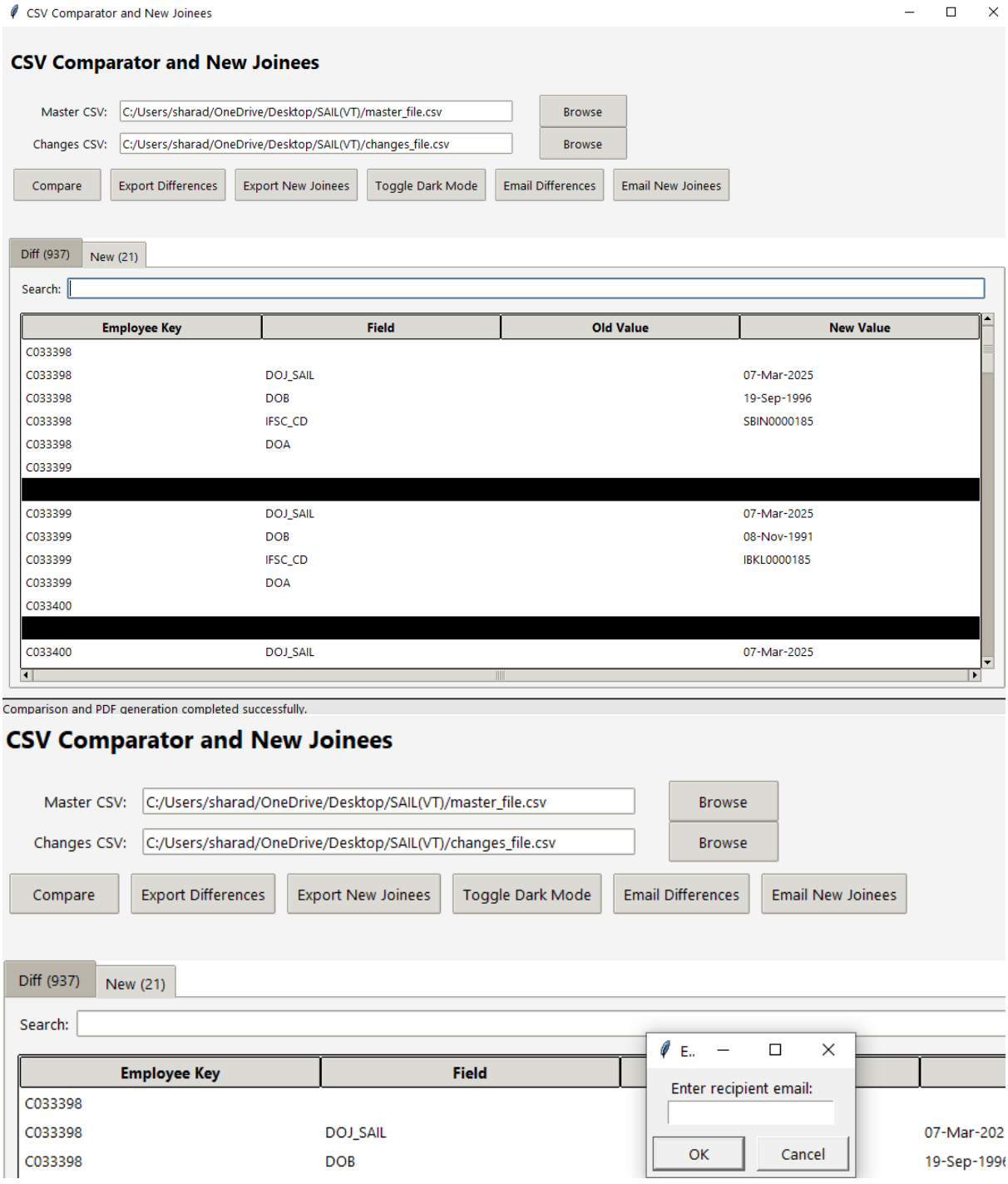
```
            prev_key = row['key']

        table = Table(data, repeatRows=1)
        table_style = TableStyle([
            ('BACKGROUND', (0, 0), (-1, 0), colors.grey),
            ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
            ('ALIGN', (0, 0), (-1, -1), 'LEFT'),
            ('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),
            ('BOTTOMPADDING', (0, 0), (-1, 0), 12),
            ('GRID', (0, 0), (-1, -1), 0.25, colors.black),
        ])

        for i in range(1, len(data)):
            if i < len(data) - 1:
                if data[i+1] == ['', '', '', '']:
                    table_style.add('LINEABOVE', (0, i), (-1, i), 2, colors.black)

        table.setStyle(table_style)
        elements.append(table)
        doc.build(elements)

        # Encrypt the PDF with the password
        with open(file_path, 'rb') as f:
            reader = PyPDF2.PdfReader(f)
            writer = PyPDF2.PdfWriter()

            # Add all pages to the writer
            for page in reader.pages:
                writer.add_page(page)

            # Encrypt the PDF
            writer.encrypt(password)

        # Save the encrypted PDF
        with open(file_path, 'wb') as f:
            writer.write(f)

        if not return_path:
            messagebox.showinfo("Success", f"{tab_key.title()} PDF exported to {file_path}")
        return file_path
    except Exception as e:
        if not return_path:
            messagebox.showerror("Error", f"Failed to export PDF: {e}")
```

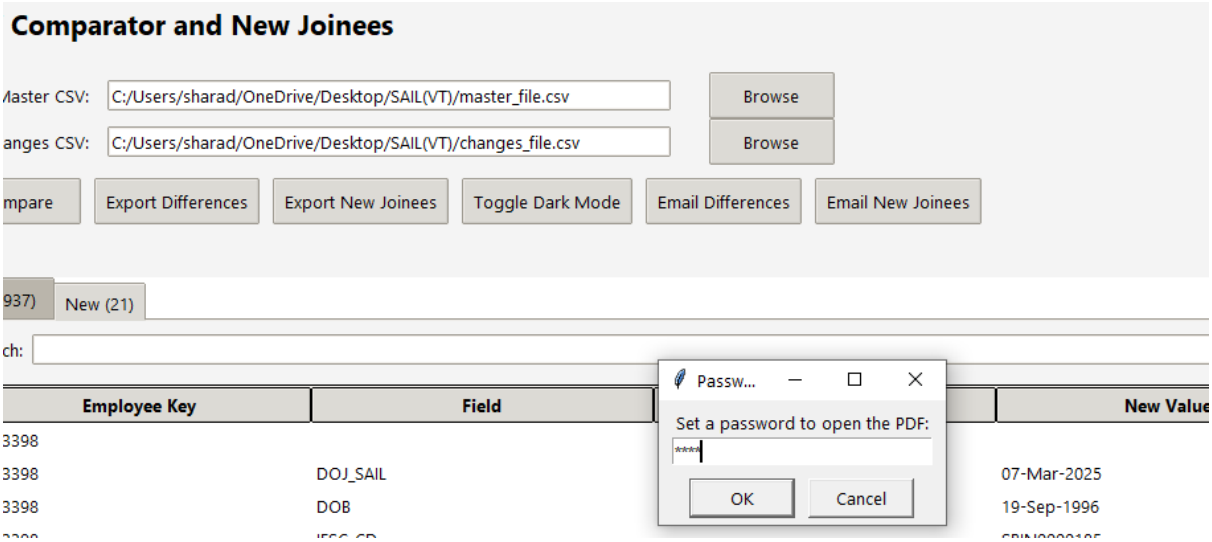
```
            return None

    def send_email(self, tab_key):
        recipient_email = simpledialog.askstring("Email", "Enter recipient email:")
        if not recipient_email:
            return

        pdf_path = self.diff_pdf_path if tab_key == 'diff' else self.new_pdf_path
        if not pdf_path or not os.path.exists(pdf_path):
            messagebox.showwarning("Warning", f"No PDF generated for {tab_key.title()}.
Please run comparison first.")
            return

        msg = EmailMessage()
        msg['Subject'] = f"{tab_key.title()} Report"
        msg['From'] = SENDER_EMAIL
        msg['To'] = recipient_email
        msg.set_content(f"Attached is the {tab_key.title()} report generated by CSV
Comparator.")

        with open(pdf_path, 'rb') as f:
            file_data = f.read()
            file_name = os.path.basename(pdf_path)
            msg.add_attachment(file_data, maintype='application', subtype='pdf',
filename=file_name)

        try:
            with smtplib.SMTP('smtp.sendgrid.net', 587) as smtp:
                smtp.starttls()
                smtp.login("apikey", SENDGRID_API_KEY)
                smtp.send_message(msg)
            messagebox.showinfo("Success", f"Email sent to {recipient_email}")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to send email: {e}")

if __name__ == "__main__":
    root = tk.Tk()
    app = CSVComparatorApp(root)
    root.mainloop()
```

OUTPUT


CSV Comparator and New Joinees
CSV Comparator and New Joinees
Master CSV:
Changes CSV:
Browse
Browse
Compare
Export Differences
Export New Joinees
Toggle Dark Mode
Email Differences
Email New Joinees
Diff (0)
New (0)
Search:
Employee Key
Field
Old Value
New Value
Welcome to CSV Comparator


CSV Comparator and New Joinees
CSV Comparator and New Joinees
Master CSV: C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/master_file.csv
Changes CSV: C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/changes_file.csv
Browse
Browse
Compare
Export Differences
Export New Joinees
Toggle Dark Mode
Email Differences
Email New Joinees
Diff (937)
New (21)
Search:
Employee Key
Field
Old Value
New Value
C026144
C026144 EPS_CD 1 4
C026144 COST_CENTER CWD0970000
C026144 LEAVE_RULES D
C026155
C026155 COST_CENTER CWD0970000
C026155 VPF_AMT 5000 10000
C026155 LEAVE_RULES D
C026161
C026161 COST_CENTER CWD0970000
C026161 VPF_AMT 5000 10000
C026161 LEAVE_RULES D
Comparison and PDF generation completed successfully.


CSV Comparator and New Joinees
CSV Comparator and New Joinees
Master CSV:
C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/master_file.csv
Browse
Changes CSV:
C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/changes_file.csv
Browse
Compare
Export Differences
Export New Joinees
Toggle Dark Mode
Email Differences
Email New Joinees
Diff (937)
New (21)
Search:
Employee Key
Field
Old Value
New Value
C033398
C033398 DOJ_SAIL 07-Mar-2025
C033398 DOB 19-Sep-1996
C033398 IFSC_CD SBIN0000185
C033398 DOA
C033399
C033399 DOJ_SAIL 07-Mar-2025
C033399 DOB 08-Nov-1991
C033399 IFSC_CD IBKL0000185
C033399 DOA
C033400
C033400 DOJ_SAIL 07-Mar-2025
Comparison and PDF generation completed successfully.
CSV Comparator and New Joinees
Master CSV:
C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/master_file.csv
Browse
Changes CSV:
C:/Users/sharad/OneDrive/Desktop/SAIL(VT)/changes_file.csv
Browse
Compare
Export Differences
Export New Joinees
Toggle Dark Mode
Email Differences
Email New Joinees
Diff (937)
New (21)
Search:
Employee Key
Field
C033398
C033398 DOJ_SAIL 07-Mar-202
C033398 DOB 19-Sep-199
E..
Enter recipient email:
OK
Cancel



CHALLENGES

1. **Data Consistency**: Ensuring that both Master and Slave datasets maintain consistent formatting and data types (e.g., date formats, string representations) to facilitate accurate comparisons.

2. **Handling Large Datasets**: Efficiently processing and comparing large CSV files (often exceeding 10,000 records) without significant performance degradation or memory issues.

3. **User Interface Design**: Creating an intuitive and user-friendly GUI that allows users to easily navigate file selection, view results, and utilize features like search and filtering.

4. **Error Handling**: Implementing robust error handling to manage potential issues such as file read/write errors, missing data, or invalid inputs, ensuring a smooth user experience.

5. **Security Measures**: Incorporating effective security features, such as password protection for generated PDF reports, to safeguard sensitive employee information from unauthorized access.

CONCLUSION

In conclusion, the CSV Comparator GUI Tool significantly enhances the efficiency and accuracy of payroll data audits by automating the comparison process between Master and Slave datasets. By leveraging powerful libraries such as Pandas for data manipulation and ReportLab for PDF generation, the application streamlines the identification of discrepancies and new joiners, ultimately reducing the time spent on manual verification. The integration of features like password protection for reports and direct email functionality further ensures that sensitive employee information is handled securely and shared conveniently.

Overall, this project not only addresses the challenges faced by the payroll team at SAIL but also sets a foundation for future enhancements, such as incorporating machine learning algorithms for predictive analytics or expanding the tool's capabilities to handle additional data formats. The successful implementation of this tool demonstrates the potential of automation in improving operational efficiency and accuracy in HR processes, paving the way for more data-driven decision-making in the organization.

FUTURE WORK

1. **Web-based Interface** - Migrate the desktop GUI to a web application using frameworks like Django/Flask for remote accessibility.

2. **Automated Scheduling** - Implement scheduled comparisons that run automatically at month-end payroll cycles without user intervention.

3. **AI-Powered Anomaly Detection** - Add machine learning to detect unusual patterns and potential payroll fraud in the comparison results.

4. **Multi-Factor Authentication** - Enhance security for PDF decryption by requiring SMS/email verification in addition to passwords.

5. **Integration with HRMS** - Develop API connections to directly pull data from HR management systems instead of CSV file uploads.

REFERENCES

1. **Pandas Documentation (2023)**
   Official guide for data manipulation using Pandas library, essential for CSV data processing in the project.

2. **ReportLab User Guide (2023)**
   Comprehensive documentation for PDF generation techniques used to create audit reports.

3. **Grayson (2000) - Python and Tkinter Programming**
   Foundational textbook for building the GUI interface with Tkinter widgets and event handling.

4. **PyPDF2 Documentation (2022)**
   Technical reference for implementing password-based encryption in generated PDF reports.

5. **Python SMTP Library Docs (2023)**
   Official Python documentation for email integration functionality using smtplib.