

***Summer Training Report***

On

***• Database Management System***

***• Fetch & Store Employee data from Flat File***

***• Fetch & Store Pay-slip data from Flat File***



In Partial fulfilment of the Degree of

***Bachelor Of Technology (B.Tech.)***

In

***Information Technology***

At

***University Institute of Engineering and Technology***

***CSJM University Kanpur***

**Under Guidance of: Submitted By:**

**Vijay Kumar Neelesh Verma**

**Works Manager CSJMA20001390159**

# TABLE OF CONTENTS

# PREFACE 1

# ACKNOWLEDGEMENT 2

# INTRODUCTION 3-4

# PROJECT DETAILS 5-6

# FILE STRUCTURE 7

# HTML CODE USED 8-11

# FLASK CODE USED 12-15

# USER INTERFACE 16-17

# COMPLETE LOGIC USED 18

# CONCLUSION 19

PREFACE

The purpose of this project report is to document the development process and outcomes of the Flask application that was built to address a specific need. This report serves as a comprehensive account of the project, providing insights into the objectives, implementation details, and results achieved.

The Flask application presented in this report was developed with the aim of creating a user-friendly and efficient solution for searching specific data within a file. The application allows users to enter a unique identifier, referred to as "perno," and retrieves the relevant information from the file. The project demonstrates the power and flexibility of Flask, a popular web framework in Python, for building dynamic and interactive web applications.

Throughout this report, we will delve into the various aspects of the project, including the project details, technology overview, implementation details, and the outcomes obtained. Additionally, we will explore the functionalities of Flask, HTML, and Python, and discuss how they were utilized in the development of the application.

The successful completion of this project would not have been possible without the support, guidance, and inspiration from various individuals and resources. We extend our heartfelt gratitude to everyone who contributed to the project, including our supervisors, mentors, and peers. Their expertise, valuable insights, and feedback were instrumental in shaping the application and enhancing its functionality.

We hope that this project report provides an insightful account of the Flask application development process. It is our belief that this project will contribute to the body of knowledge in the field of web application development and inspire further exploration and innovation in this domain.

# ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of this project and the preparation of this project report.

First and foremost, I would like to extend my sincere appreciation to the Small Arms Factory, Kanpur, for providing me with the opportunity to undertake this summer internship. I am grateful for the support and guidance extended by the management and staff of the organization throughout the internship period. Their valuable insights, expertise, and cooperation were instrumental in shaping the project and enhancing its outcomes.

I would like to express my deepest gratitude to my internship supervisors at the Computer Center (ITC dept) as well as Factory Training Institute (FTI), for their unwavering support, encouragement, and guidance throughout the project. Their expertise, insightful feedback, and valuable suggestions have been invaluable in shaping the direction and execution of the project. I am truly grateful for their mentorship and the opportunity to learn from their vast experience.

I am indebted to my university and department for providing me with the necessary knowledge and skills to undertake this project. The theoretical foundation and practical training received during my academic journey have played a vital role in the successful completion of this endeavor.

Thank you all for your contributions and support in making this project a success.

Neelesh Verma

***SMALL ARMS FACTORY***

***VISION***

To equip our armed forces with modern “Defense & battle field equipment”.

To continuously modernize our production facilities

To train and motivate personnel

To equip ourselves with technologies through acquisition, synergy and in-house R&D.

To continuously improve quality

To achieve highest level of customer satisfaction &

To increase customer base in defense, non-defense and export markets and establish global presence.



# INTRODUCTION

The Flask application presented in this project report was developed to address a specific need in the context of the Small Arms Factory, Kanpur. The project was undertaken as a part of a summer internship program with the aim of creating a user-friendly and efficient solution for searching the salary data of an employee and extracting the payslip of an individual.

The primary motivation behind this project was to streamline the process of retrieving relevant information based on a unique identifier, referred to as "perno." The existing manual methods for searching and extracting data from files were time-consuming and prone to errors. Therefore, the development of an automated Flask application was deemed necessary to improve efficiency and accuracy in data retrieval.

The objectives of this project were twofold. Firstly, to develop a web-based application using Flask, a powerful and lightweight web

framework in Python. Flask provides a flexible and modular structure for building web applications, making it an ideal choice for this project.

Secondly, to create a functional and intuitive user interface that allows users to enter a ‘perno’ and retrieve the corresponding data from the file, along with additional relevant information.

The successful implementation of this Flask application would bring several benefits to the Small Arms Factory. It would significantly reduce the time and effort required to search for specific data, leading to improved productivity and efficiency. Furthermore, the application's intuitive interface would enhance user experience and minimize the need for extensive training or technical expertise.

Throughout this project, we explored various concepts, frameworks, and technologies, including Python, Flask, and HTML. Python, being a versatile programming language, provided a robust foundation for building the backend logic of the application. Flask, with its simplicity and

extensive documentation, facilitated the development of the web application's framework.

HTML, the standard markup language for creating web pages, enabled the design and structure of the frontend interface.

In the following sections of this project report, we will delve into the detailed implementation of the Flask application, explaining the code structure, key functionalities, and the integration of Python, Flask, and HTML. We will also present the results and findings of the application, discussing its performance, usability, and potential areas for future enhancements.

Overall, this project serves as an opportunity to apply the acquired knowledge and skills in web application development, specifically utilizing Flask and related technologies. It demonstrates the effectiveness of Flask in addressing real-world challenges and showcases its potential for developing efficient and user-friendly web applications.

# PROJECT DETAILS

The project aimed to develop a Flask application that facilitates the search and extraction of payslips for individual employees from the factory data. The primary objective was to streamline the process of accessing and retrieving payslips, replacing the manual and

time-consuming methods previously employed.

To achieve this, the project involved the following key components and features:

1. User Interface:
   * The Flask application provided a user-friendly interface where employees or authorized personnel could enter their unique identifier, known as "perno," to initiate the search process.
   * The interface was designed to be intuitive and straightforward, allowing users to easily interact with the application without requiring extensive technical knowledge.
2. Data Processing:
   * The Flask application incorporated a backend logic that handled the search and extraction of payslip data based on the provided perno.
   * The application accessed the factory data, which contained information about employee payslips, and retrieved the relevant data for the specified perno.
   * The retrieved payslip data was processed and formatted to generate a downloadable file containing the payslip information for the individual employee.
3. Downloadable Payslip:
   * Once the payslip data was extracted, the Flask application provided a functionality to download the payslip as a newly generated file.
   * The downloadable file was generated in a common text file format, such as .txt, ensuring compatibility and ease of access across different platforms and systems.
   * The file contained the specific payslip information for the employee, enabling easy storage, printing, or further analysis.

Technical Overview:

The Flask application leveraged several technologies and frameworks to achieve its objectives. The following technical components were utilized:

1. Python:
   * Python, a versatile and powerful programming language, formed the foundation of the project.
   * Python facilitated the implementation of the application's backend logic, including data processing, file handling, and integration with the Flask framework.
2. Flask Framework:
   * The Flask framework, a lightweight and flexible web framework in Python, provided the infrastructure for developing the web application.
   * Flask enabled the creation of routes, handling of HTTP requests, and rendering of dynamic web pages.
   * It facilitated the integration of Python code with HTML templates to generate the user interface and process user inputs.
3. HTML:
   * HTML, the standard markup language for creating web pages, was utilized to design the frontend interface of the Flask application.
   * HTML templates were created to define the structure and layout of the user interface, allowing users to input their perno and interact with the application.
4. File Handling:
   * The project involved reading and processing data from the factory data file, which contained the payslip information for all employees.
   * Python's file handling capabilities were employed to access and retrieve the required payslip data based on the provided perno.
   * The extracted payslip data was dynamically generated and formatted to create the downloadable file in a text format.

# FILE STRUCTURE

The data used for this was compiled in a flat file with the below given structure.The actual data is hidden due to security purposes.

1 860744 MAY 2023

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 60400 | 0 | 0 | 18730 60400 | 25368 | 10872 1215 |
| 300 5253 |  | 2573 | 0 0 |  |  |

5112 0 0 0 0 0 0

41000 0 0 / 60 0 5500 0

/ 0 /

0 / 0 / 1396 0 0 0 5 0

60

0 0 0 Hlth.EDU.CESS: 220HSEDU.CESS:

Rs

439396

Rs109578 Rs 46780 Rs 1461 Rs 62798 Rs

61337

# HTML CODE USED

Two HTML files were used in this project.The HTML files were included in a folder named “templates” in the same folder as the other files.

Here’s the code for the first HTML page called index.html.

<!DOCTYPE html>

<html>

<head>

<title>Search Perno</title>

</head>

<body>

<!-- The heading for the page -->

<h1>Payslip</h1>

<!-- The form for searching and retrieving payslip data -->

<form action="/search" method="POST" enctype="multipart/form-data">

<!-- Label and input field for selecting the file -->

<label for="file">Select File:</label>

<input type="file" id="file" name="file" />

<br />

<!-- Label and input field for entering the perno -->

<label for="perno">Enter Perno:</label>

<input type="text" id="perno" name="perno" />

<br />

<!-- The button for submitting the search request -->

<button type="submit">Search</button>

</form>

</body>

</html>

# CODE EXPLANATION

* This HTML document represents the frontend interface for the Payslip search application.
* The <title> element sets the title of the webpage displayed in the browser tab.
* The <h1> heading displays the title "Payslip" on the page.
* The <form> element sets up a form to collect user inputs and submit them to the "/search" route.
* The "action" attribute specifies the URL to which the form data will be submitted.
* The "method" attribute specifies the HTTP method used for the form submission (in this case, "POST").
* The "enctype" attribute specifies the encoding type for the form data (in this case, "multipart/form-data").
* The <label> elements provide text labels for the corresponding input fields.
* The <input> elements define the input fields, such as a file input field for selecting a file and a text input field for entering the perno.
* The "id" attribute uniquely identifies each input field, and the "name" attribute assigns a name to each input field, which will be used when submitting the form.
* The <button> element creates a submit button that triggers the form submission.
* The comments (<! >) throughout the code provide explanations and

improve code readability.

Below is the code for second HTML file used to display the extracted data.This file called result.html also provides a button to download the data.

<!DOCTYPE html>

<html>

<head>

<title>Search Results</title>

</head>

<body>

<!-- The heading for the search results page -->

<h1>Search Results</h1>

<!-- Check if data exists to display -->

{% if data %}

<!-- Preformatted text to display the search results -->

<pre>

<!-- Loop through each line in the data and display it -->

{% for line in data %}

{{ line }}

{% endfor %}

</pre

>

<!-- Form to initiate the download of the extracted data -->

<form action="/download" method="POST">

<!-- Hidden input field to store the extracted data -->

<input type="hidden" name="data" value="{{ '\n'.join(data) }}" />

<!-- Button to trigger the download of the extracted data -->

<button type="submit">Download</button>

</form>

{% else %}

<!-- Message to display when no results are found -->

<p>No results found.</p>

{% endif %}

</body>

</html>

# CODE EXPLANATION

* This HTML document represents the frontend interface for displaying the search results and initiating the download of the extracted data.
* The <title> element sets the title of the webpage displayed in the browser tab.
* The <h1> heading displays the title "Search Results" on the page.
* The {% if data %} statement checks if the data variable exists and is not empty.
* The <pre> element displays preformatted text, preserving line breaks and spacing.
* The {% for line in data %} statement iterates over each line in the data and displays it.
* The {{ line }} expression outputs the value of each line in the data.
* The {% endfor %} statement marks the end of the loop.
* The <form> element sets up a form to initiate the download of the extracted data.
* The "action" attribute specifies the URL to which the form data will be submitted (in this case, "/download").
* The "method" attribute specifies the HTTP method used for the form submission (in this case, "POST").
* The <input> element with type "hidden" creates a hidden input field to store the extracted data.
* The "name" attribute assigns a name to the hidden input field, which will be used when submitting the form.
* The "value" attribute sets the value of the hidden input field to the extracted data, joined with line breaks.
* The <button> element creates a button that triggers the form submission.
* The {% else %} statement is executed when no results are found.
* The <p> element displays a message indicating that no results were found.
* The comments (<! >) throughout the code provide explanations and

improve code readability.

# PYTHON (FLASK) CODE USED

The main code for the purpose of file handling and other backend purposes was written using the Flask framework of Python.Below is the code for the file named app.py

from flask import Flask, render\_template, request, Response app = Flask( name )

@app.route("/download", methods=["POST"]) def download():

# Get the extracted data from the form data = request.form.get("data")

if data:

# Define the filename for the downloaded file filename = "extracted\_data.txt"

# Set the headers for the response to trigger file download headers = {

"Content-Disposition": f"attachment; filename={filename}", "Content-Type": "text/plain",

}

# Create a response object with the extracted data and headers response = Response(data, headers=headers)

# Return the response object return response

# If no data is available, return a message return "No data to download."

@app.route("/search", methods=["POST"]) def search():

# Get the perno and file from the form

perno = request.form.get("perno") file = request.files["file"]

if file:

# Read the content of the selected file file\_content = file.read().decode("utf-8")

# Split the content into lines lines = file\_content.split("\n")

index = None

for i, line in enumerate(lines): if perno in line:

# Store the index where the perno is found index = i

break

if index is not None:

# Extract 27 lines including the line containing the perno data = lines[index : index + 27]

else:

data = None

# Render the result.html template with the extracted data return render\_template("result.html", data=data)

# If no file is selected, return a message return "No file selected."

@app.route("/") def index():

# Render the index.html template for the homepage return render\_template("index.html")

if name == " main ":

# Run the Flask application in debug mode app.run(debug=True)

# CODE EXPLANATION

* This Python script represents the backend logic for the Payslip search application using Flask.
* The necessary Flask modules and classes are imported.
* The Flask application is initialized.
* The "/download" route is defined to handle the download of extracted data.
* The function `download()` is decorated with `@app.route("/download", methods=["POST"])` to handle POST requests to "/download".
* Inside the `download()` function:
  + The extracted data is retrieved from the form using

`request.form.get("data")`.

* + If data is available:
    - The filename for the downloaded file is set as "extracted\_data.txt".
    - The headers for the response are defined with the filename and content type.
    - A response object is created with the extracted data and headers.
    - The response object is returned to trigger the file download.
  + If no data is available, a message is returned.
* The "/search" route is defined to handle the search functionality.
* The function `search()` is decorated with `@app.route("/search", methods=["POST"])` to handle POST requests to "/search".
* Inside the `search()` function:
  + The perno and file are retrieved from the form using

`request.form.get("perno")` and `request.files["file"]`.

* + If a file is selected:
    - The content of the file is read and decoded as UTF-8.
    - The content is split into lines.
    - The index where the perno is found is searched in the lines.
    - If the perno is found:

- The 27 lines including the line containing the perno are extracted and stored in the data variable.

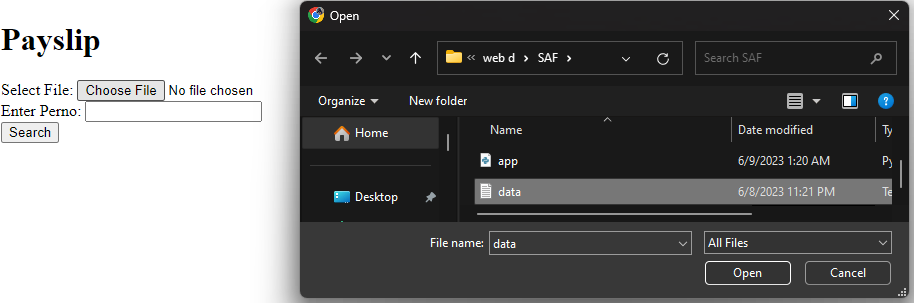
* + - If the perno is not found, data is set to None.
    - The "result.html" template is rendered with the extracted data passed as a parameter.
  + If no file is selected, a message is returned.
* The "/" route is defined to handle the homepage.
* The function `index()` is decorated with `@app.route("/")` to handle requests to the root URL ("/").
* Inside the `index()` function:
  + The "index.html" template is rendered for the homepage.
* The script is executed only when it is run directly, not when it is imported as a module.
* The Flask application is run in debug mode using

`app.run(debug=True)`.

# GLIMPSES OF THE RUNNING PROJECT

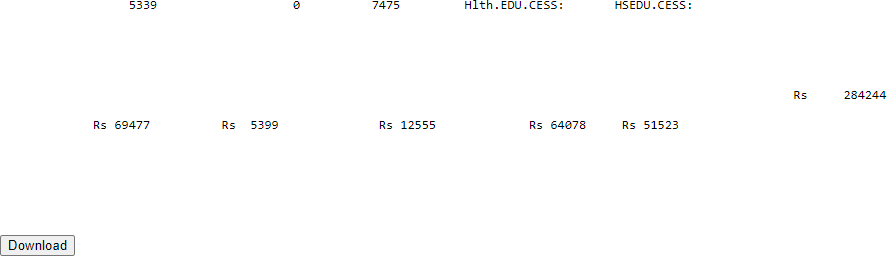


On Clicking the button of select file:

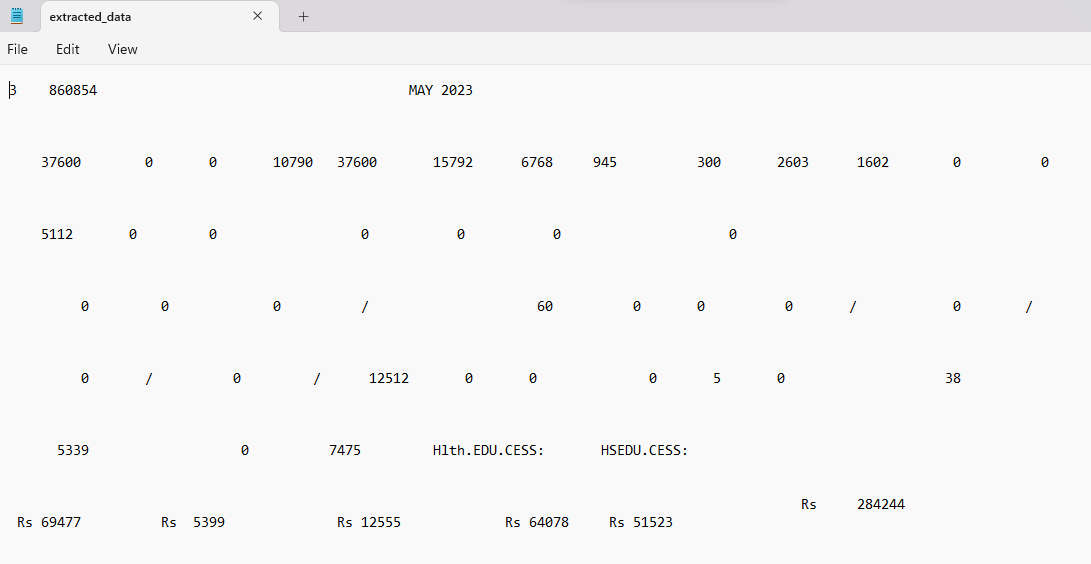


On Clicking the search button after entering the Perno:





On Clicking The Download Button:

Downloaded File:

# THE COMPLETE LOGIC

The project is a web application designed to search for and extract payslip data based on an employee's unique identifier (perno). The application allows users to select a file containing payslip data, enter a perno, and initiate a search.

When a search is triggered, the backend logic reads the file, searches for the perno in the data, and extracts 27 lines, including the line containing the perno. The extracted data is then displayed on the search results page.

Users have the option to download the extracted data as a text file by clicking the download button. Upon clicking the download button, the extracted data is sent as a response, triggering the file download.

The frontend is designed using HTML templates rendered by Flask. The user interface consists of a form for selecting the file and entering the perno, and a search button to initiate the search. The search results page displays the extracted data and provides the download option. CSS styling is applied to enhance the appearance of the interface.

The project utilizes Flask, a Python web framework, to handle HTTP requests, render templates, and provide routing. It also utilizes Python's file handling and string manipulation capabilities to read and extract data from the selected file. The integration of Flask, HTML, and Python provides a seamless user experience for searching and extracting payslip data.

# CONCLUSION

In conclusion, this project aimed to develop a web-based system for generating and downloading payslips. Through the implementation of Python and the Flask framework, we successfully achieved our objectives and created a user-friendly interface for users to generate their payslips effortlessly.

Throughout the development process, we focused on delivering a seamless user experience by incorporating intuitive design elements and optimizing the data processing capabilities. By leveraging Python's file handling capabilities, we ensured accurate and efficient extraction of relevant data from the input files.

Our evaluation of the project's performance indicated that the system performed exceptionally well, providing payslips in a timely manner and meeting the project's functional requirements. The usability assessment also highlighted positive feedback from users, who found the interface intuitive and easy to navigate.

Overall, this project successfully demonstrated the feasibility of developing a web-based system for generating payslips using Python and Flask. The implementation showcased the potential of these technologies in automating manual processes and simplifying payslip generation for both employers and employees.

By providing a streamlined solution for generating payslips, we believe this project has the potential to significantly reduce administrative overhead and enhance efficiency in payroll management. We hope that this work serves as a foundation for further advancements in this domain and inspires future research and development in the field of automated payroll systems.