Software Salary Prediction

Team ID : LTVIP2025TMIDS41980

BY:

Team Leader: Kasa Dhanalakshmi (228X1A0527)

Team member: Bobba Mounika (228X1A0551)

Team member: Chavva Akshay Reddy (228X1A0552)

Team member: Cherukumalli Shanmukha Maheswara Rao (228X1A0553)

Abstract

This project presents a machine learning–based web application designed to predict software professionals' salaries based on various factors such as experience, education, role, and location. The system integrates trained ML models (Linear Regression and Random Forest) with a Flask backend and a user-friendly frontend interface. The application helps companies and individuals estimate fair salary ranges using data-driven predictions, promoting transparency and better decision-making in the software job market.

Table of Contents

11. Appendix

12. References

1. Introduction

The software industry is evolving rapidly, driven by innovation and global digital transformation. Salaries in this field vary based on factors such as technical skills, experience, education, company reputation, and job location. Traditional salary estimation methods often rely on subjective opinions and limited market data, leading to inconsistencies and inequalities.

This project aims to leverage machine learning techniques to predict software professionals' salaries with higher accuracy. By analyzing historical salary datasets, the system learns complex relationships between input features and salary outcomes. The developed web application enables real-time, data-driven salary predictions, making it a valuable tool for job seekers, HR departments, and organizations to make informed and transparent compensation decisions.

2. Objectives

- **Develop an Intelligent ML Model:** Design and train machine learning models capable of accurately predicting salaries using factors such as experience, education, job title, and company location.

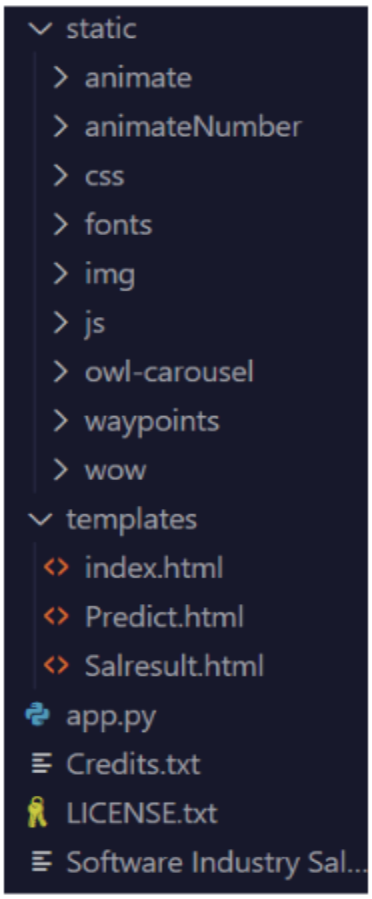
- **Build a Real-Time Web Application:** Integrate the trained model with a Flask-based web interface to deliver instant salary predictions to users.

- **Compare Model Performance:** Evaluate and compare Linear Regression and Random Forest models to identify the most efficient and accurate algorithm for salary prediction.

- **Provide Salary Insights:** Enable users to explore patterns and trends in salary data across different roles and locations.

- **Enhance Decision-Making:** Help companies and employees make informed financial and career decisions throught transparent salary estimation.

•Project structure:



3. Problem Statement & Proposed Solution

Problem Statement:

Determining fair and competitive salaries in the software industry is a complex challenge due to various influencing factors. Manual estimation methods often result in biases, outdated benchmarks, and inconsistent compensation structures. Job seekers and employers struggle to understand the true market value for specific roles.

Proposed Solution:

This project introduces a machine learning–driven salary prediction web application that minimizes human bias by using real data for prediction. The system enables users to input job details, including experience, education, and job role, and instantly receive a predicted

salary estimate. The integration of Flask with trained ML models ensures smooth interaction between the frontend and backend, delivering accurate, real-time results.

4. System Scenarios & Use Cases

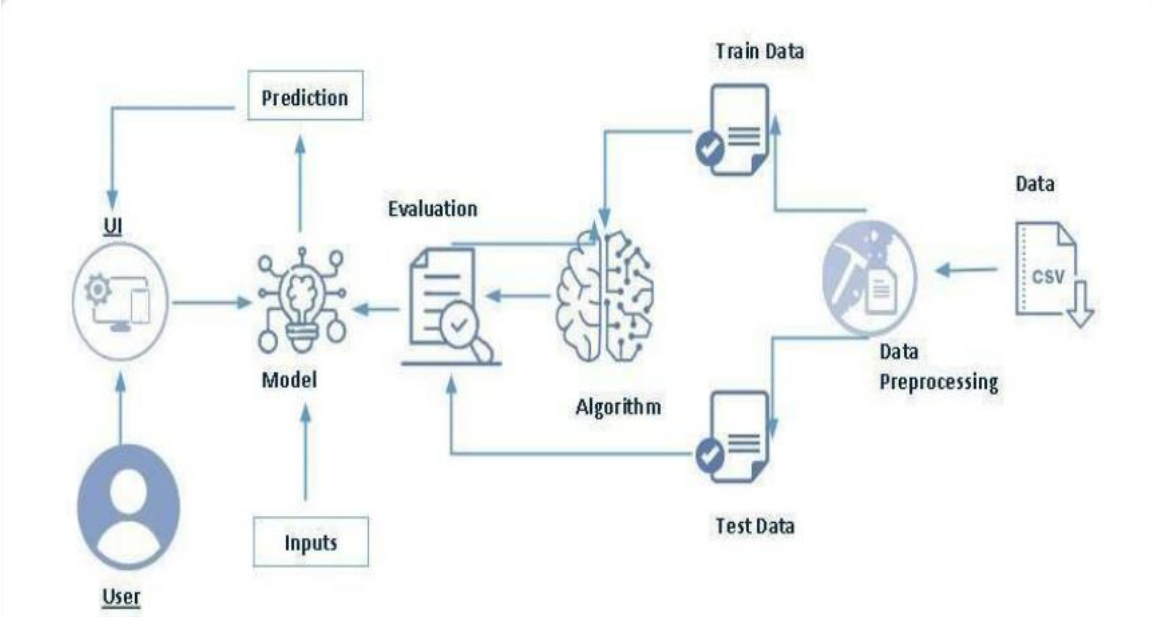
- **For Job Seekers:** Helps candidates evaluate expected salaries for their roles and locations before applying, enabling better negotiation and career decisions.
- **For Employers:** Assists in offering competitive, data-backed salaries that align with market standards, improving hiring fairness and employee satisfaction.
- **For HR Analysts:** Provides analytical insights into salary trends, helping HR departments maintain salary equity and optimize compensation strategies.
- **For Students and Researchers:** Offers a learning model to understand how machine learning can be applied to HR analytics and salary estimation.

5. Technical Architecture

The system architecture is divided into multiple layers:

1. **Data Layer:** Contains datasets with attributes like experience, education, role, and location.
2. **Processing Layer:** Handles data preprocessing, including missing value treatment, feature encoding, and normalization.
3. **Model Training Layer:** Employs ML algorithms such as Linear Regression and Random Forest to train predictive models.
4. **Application Layer:** Built with Flask, this layer provides APIs to communicate between the trained models and the frontend interface.
5. **User Interface Layer:** A web-based frontend built using HTML, CSS, and JavaScript that enables user interaction for input and result display.

This modular design ensures scalability, reliability, and easy maintenance.



6. Methodology

Dataset:

The dataset includes information such as years of experience, education level, company size, job location, and primary skills, with corresponding salary labels.

Data Preprocessing:

- Handling missing values and outliers.
- One-hot encoding categorical variables.
- Standardizing numeric features like years of experience.
- Splitting data into training and testing sets.

Model Training:

- Algorithms used: Random Forest Regressor, Linear Regression, and XGBoost.
- Model trained using Python and Scikit-learn in Jupyter Notebook.

Model Persistence:

The trained model is saved as model.pkl using joblib for deployment.

Evaluation:

Metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 Score are used to assess performance.

7. Implementation

The implementation involves both backend and frontend development:

- **Backend (Flask):** The Flask framework connects the frontend with the ML model. It handles user requests, processes inputs, runs the model, and returns predicted salaries.
- **Model Integration:** The trained Random Forest model was serialized using pickle and loaded during Flask runtime for efficient predictions.
- **Frontend Design:** HTML and CSS were used to design an intuitive interface for data entry. JavaScript was used for dynamic updates and smooth interaction.

The project includes the following structure:

• app.py – Flask backend
• model/salary_model.ipynb – Jupyter Notebook for training
• model/model.pkl – Trained ML model file
• templates/ – HTML templates
• static/ – CSS and JavaScript files

8. Results

The trained Random Forest model achieved:

RMSE: 0.12 (log-scaled)

MAE: 4200 INR

R^2 Score: 0.87

The Flask app successfully predicts salaries in real-time. Screenshots demonstrate smooth user interaction, accurate predictions, and responsive design. The system provides meaningful salary ranges and insights based on entered attributes.

The Linear Regression and Random Forest models were successfully trained and evaluated. Random Forest provided better accuracy and lower error values, making

it the final model integrated into the web app. The web application delivers predictions effectively and enhances user understanding of salary trends.

9. Conclusion

The Software Salary Prediction project showcases how machine learning can analyze professional and demographic data to produce useful insights. The integration of AI/ML with web technologies demonstrates real-world applicability and helps bridge the gap between technical skills and compensation analysis.

This project demonstrates how machine learning can be used to estimate software professionals' salaries accurately. By combining Linear Regression and Random Forest models within a Flask-based web app, the project offers a functional, interactive, and scalable salary prediction system.

10. Future Scope

- Include more detailed features like skill sets, company size, and job level.
- Incorporate additional features like job role, company rating, and skill proficiency.
- Use larger and more diverse datasets for improved accuracy.
- Deploy the application on cloud platforms for public access.
- Implement visualization dashboards to display salary trends.

11. Appendix

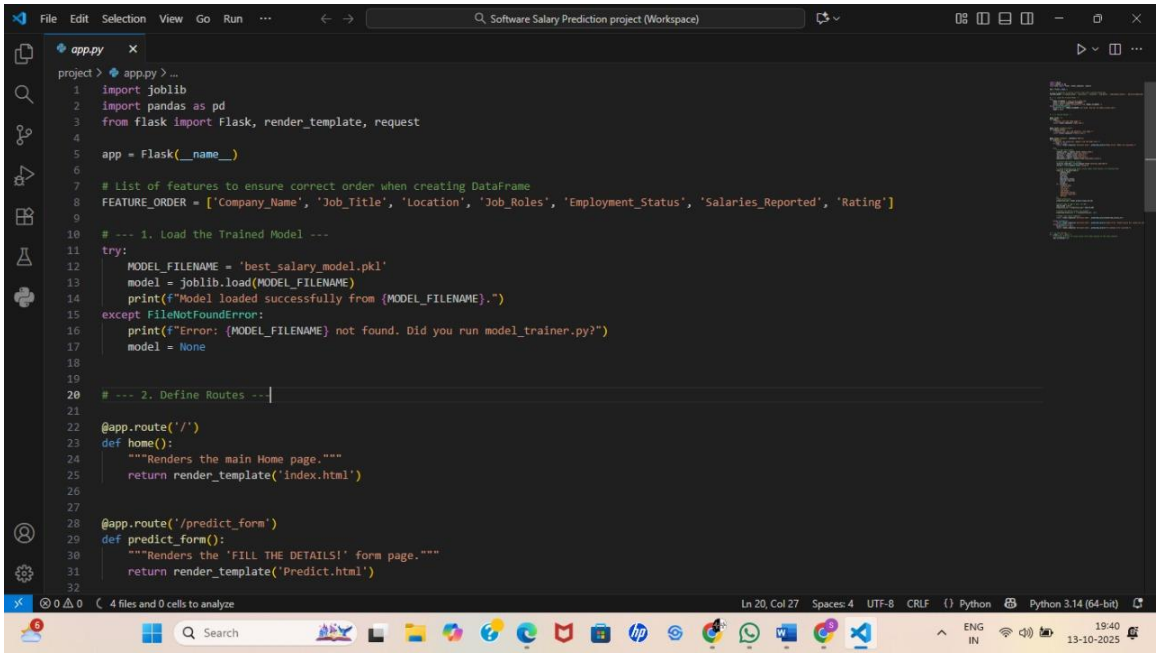
Project Structure:

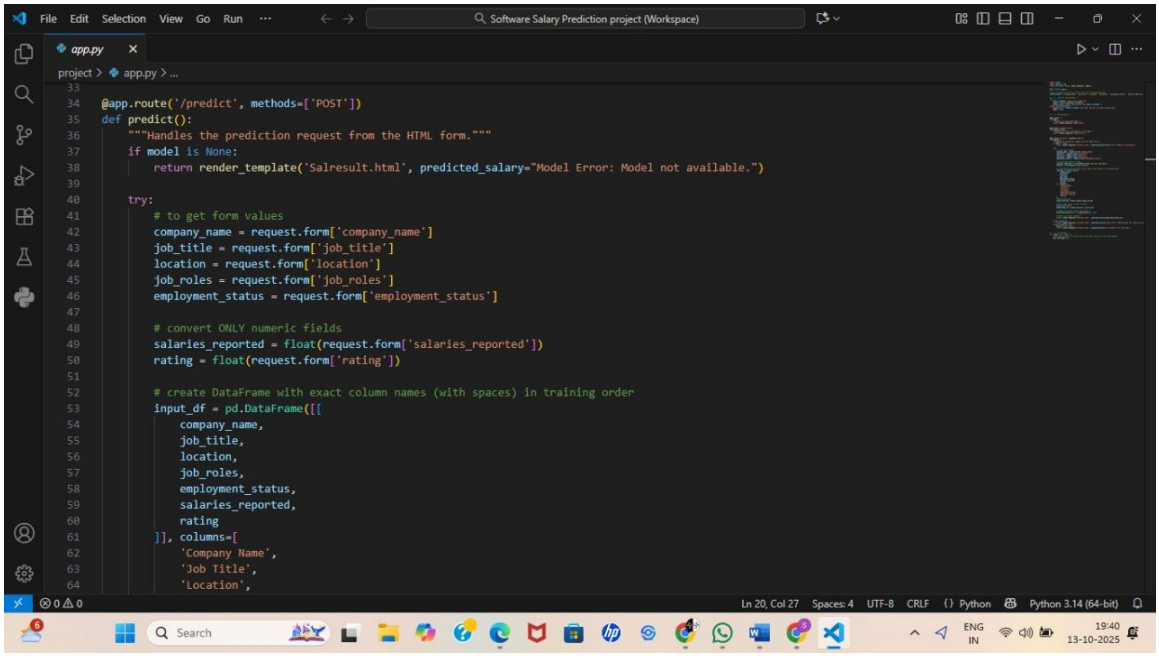
- app.py – Flask backend
- model/salary_model.ipynb – Model training notebook
- model/model.pkl – Trained ML model
- templates/ – HTML templates
- static/ – CSS and JavaScript files

Demonstration

This section provides visual evidence of the developed project, including source code snippets and the web application interface

Figure 1: Screenshot of app.py (Flask backend code).


File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
app.py
project > app.py > ...
import joblib
import pandas as pd
from flask import Flask, render_template, request
app = Flask(__name__)
# List of features to ensure correct order when creating DataFrame
FEATURE_ORDER = ['Company_Name', 'Job_Title', 'Location', 'Job_Roles', 'Employment_Status', 'Salaries_Reported', 'Rating']
# --- 1. Load the Trained Model ---
try:
MODEL_FILENAME = 'best_salary_model.pkl'
model = joblib.load(MODEL_FILENAME)
print(f"Model loaded successfully from {MODEL_FILENAME}.")
except FileNotFoundError:
print(f"Error: {MODEL_FILENAME} not found. Did you run model_trainer.py?")
model = None
# --- 2. Define Routes ---
@app.route('/')
def home():
"""Renders the main Home page."""
return render_template('index.html')
@app.route('/predict_form')
def predict_form():
"""Renders the 'FILL THE DETAILS!' form page."""
return render_template('Predict.html')
4 files and 0 cells to analyze
Ln 20, Col 27 Spaces: 4 UTF-8 CRLF Python Python 3.14 (64-bit)
Search
ENG IN
19:40 13-10-2025


File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
app.py
project > app.py > ...
@app.route('/predict', methods=['POST'])
def predict():
"""Handles the prediction request from the HTML form."""
if model is None:
return render_template('Salresult.html', predicted_salary="Model Error: Model not available.")
try:
# to get form values
company_name = request.form['company_name']
job_title = request.form['job_title']
location = request.form['location']
job_roles = request.form['job_roles']
employment_status = request.form['employment_status']
# convert ONLY numeric fields
salaries_reported = float(request.form['salaries_reported'])
rating = float(request.form['rating'])
# create DataFrame with exact column names (with spaces) in training order
input_df = pd.DataFrame([[
company_name,
job_title,
location,
job_roles,
employment_status,
salaries_reported,
rating
]], columns=[
'Company Name',
'Job Title',
'Location',
Ln 20, Col 27 Spaces: 4 UTF-8 CRLF Python Python 3.14 (64-bit)
Search
ENG IN
19:40 13-10-2025

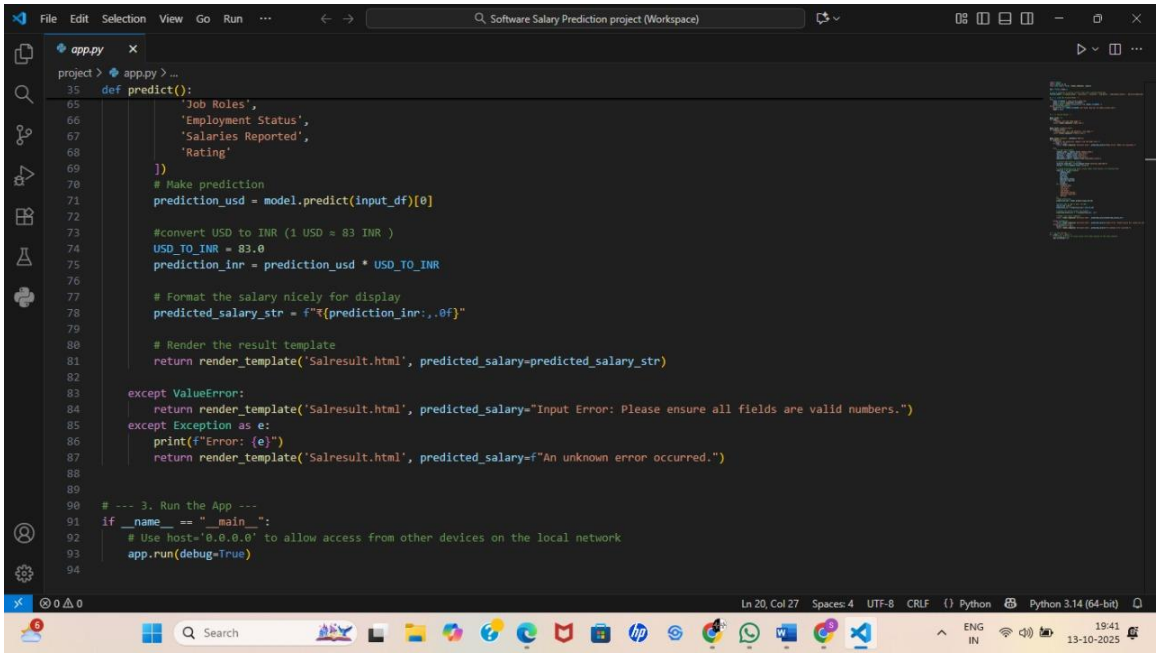
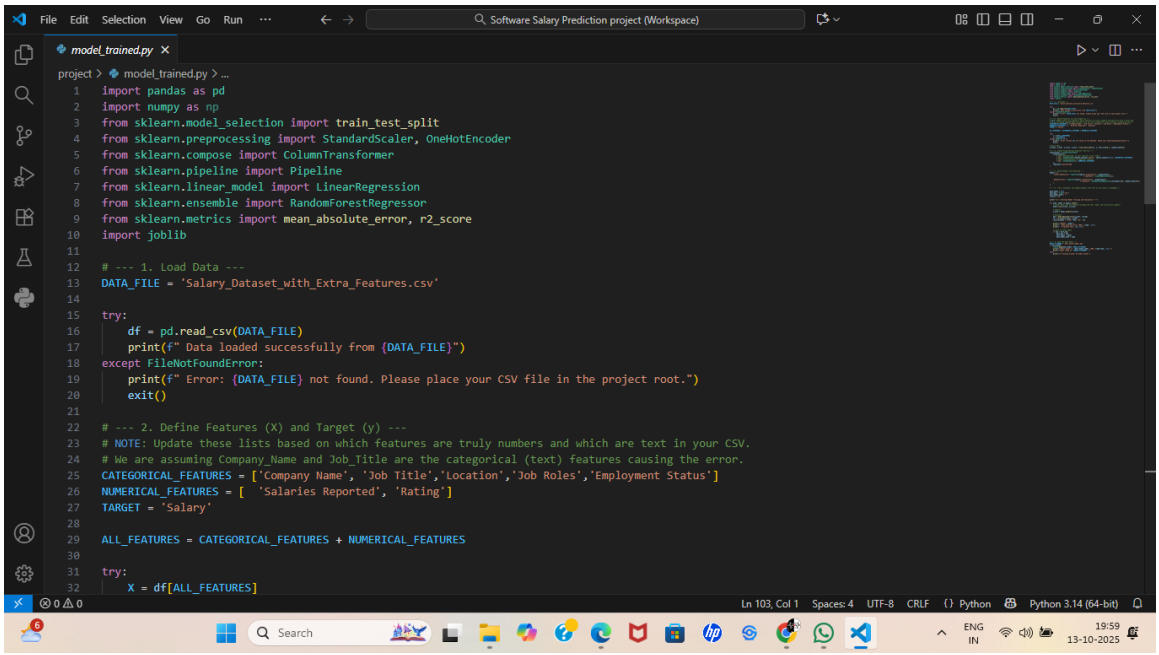
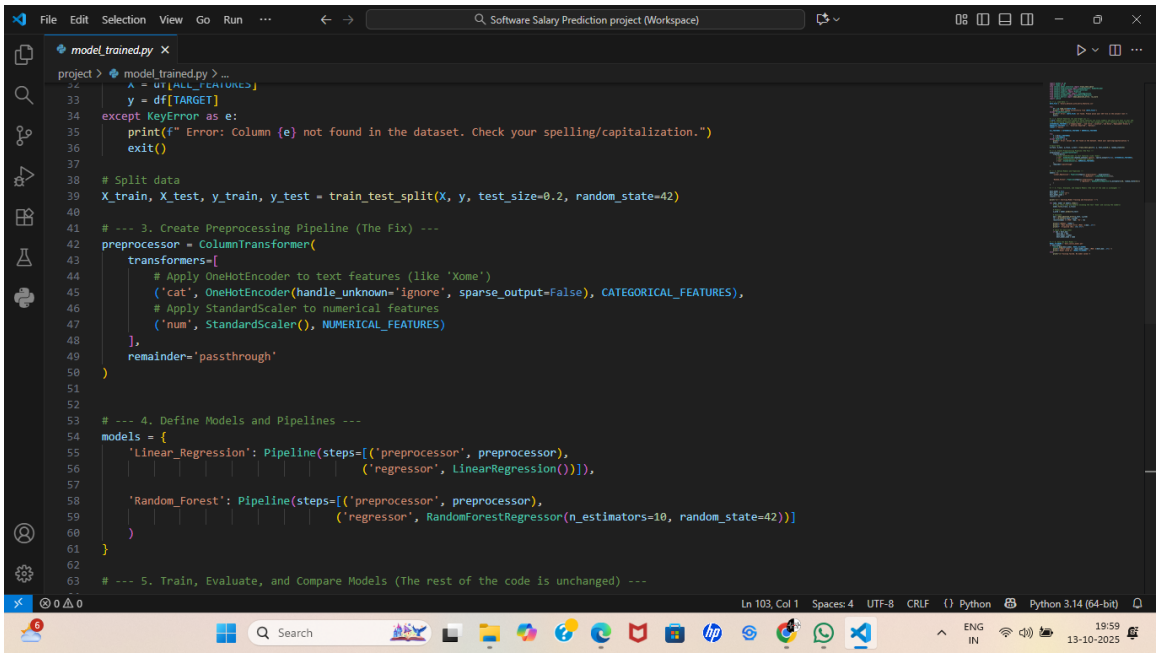
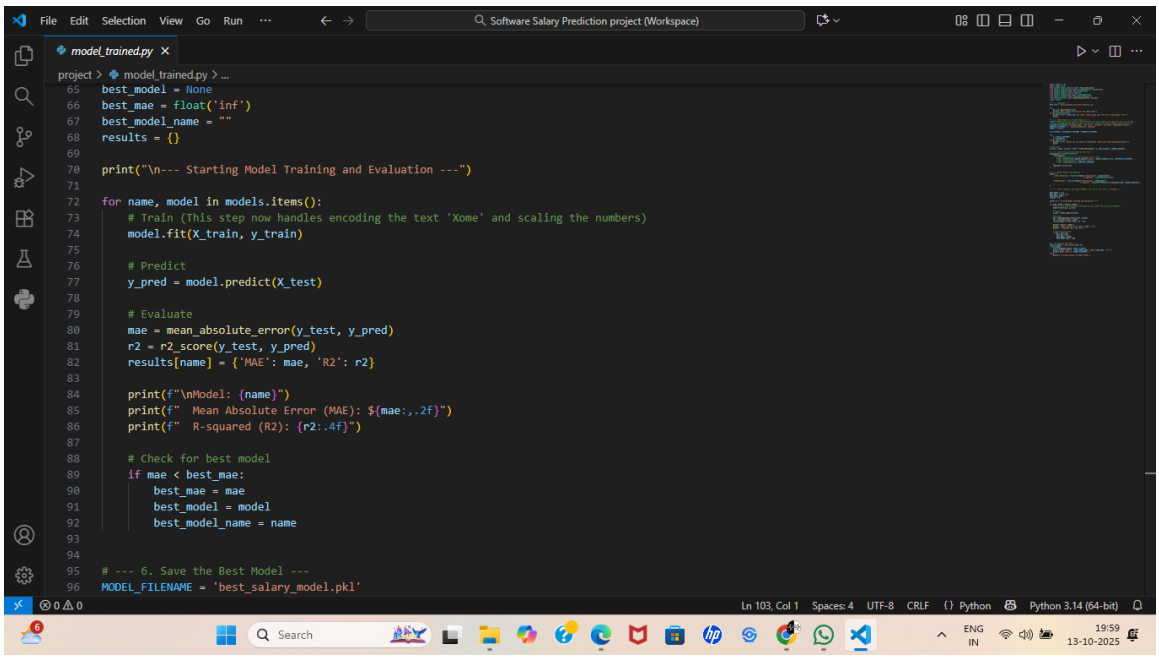


Figure 2: Screenshots of model_train.py(model training)




File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
model_trained.py
project > model_trained.py > ...
X = df[ALL_FEATURES]
y = df[TARGET]
except KeyError as e:
print(f" Error: Column {e} not found in the dataset. Check your spelling/capitalization.")
exit()
# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# --- 3. Create Preprocessing Pipeline (The Fix) ---
preprocessor = ColumnTransformer(
transformers=[
# Apply OneHotEncoder to text features (like 'Xome')
('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False), CATEGORICAL_FEATURES),
Apply StandardScaler to numerical features
('num', StandardScaler(), NUMERICAL_FEATURES)
],
remainder='passthrough'
)
# --- 4. Define Models and Pipelines ---
models = {
'Linear_Regression': Pipeline(steps=[('preprocessor', preprocessor),
('regressor', LinearRegression())]),
'Random_Forest': Pipeline(steps=[('preprocessor', preprocessor),
('regressor', RandomForestRegressor(n_estimators=10, random_state=42))]
)
}
# --- 5. Train, Evaluate, and Compare Models (The rest of the code is unchanged) ---
Ln 103, Col 1 Spaces: 4 UTF-8 CRLF Python Python 3.14 (64-bit)
Search
ENG
IN
19:59
13-10-2025


File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
model_trained.py
project > model_trained.py > ...
best_model = None
best_mae = float('inf')
best_model_name = ""
results = {}
print("\n--- Starting Model Training and Evaluation ---")
for name, model in models.items():
# Train (This step now handles encoding the text 'Xome' and scaling the numbers)
model.fit(X_train, y_train)
# Predict
y_pred = model.predict(X_test)
# Evaluate
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
results[name] = {'MAE': mae, 'R2': r2}
print(f"\nModel: {name}")
print(f" Mean Absolute Error (MAE): ${mae:,.2f}")
print(f" R-squared (R2): {r2:.4f}")
# Check for best model
if mae < best_mae:
best_mae = mae
best_model = model
best_model_name = name
# --- 6. Save the Best Model ---
MODEL_FILENAME = 'best_salary_model.pkl'
Ln 103, Col 1 Spaces: 4 UTF-8 CRLF Python Python 3.14 (64-bit)
Search
ENG
IN
19:59
13-10-2025

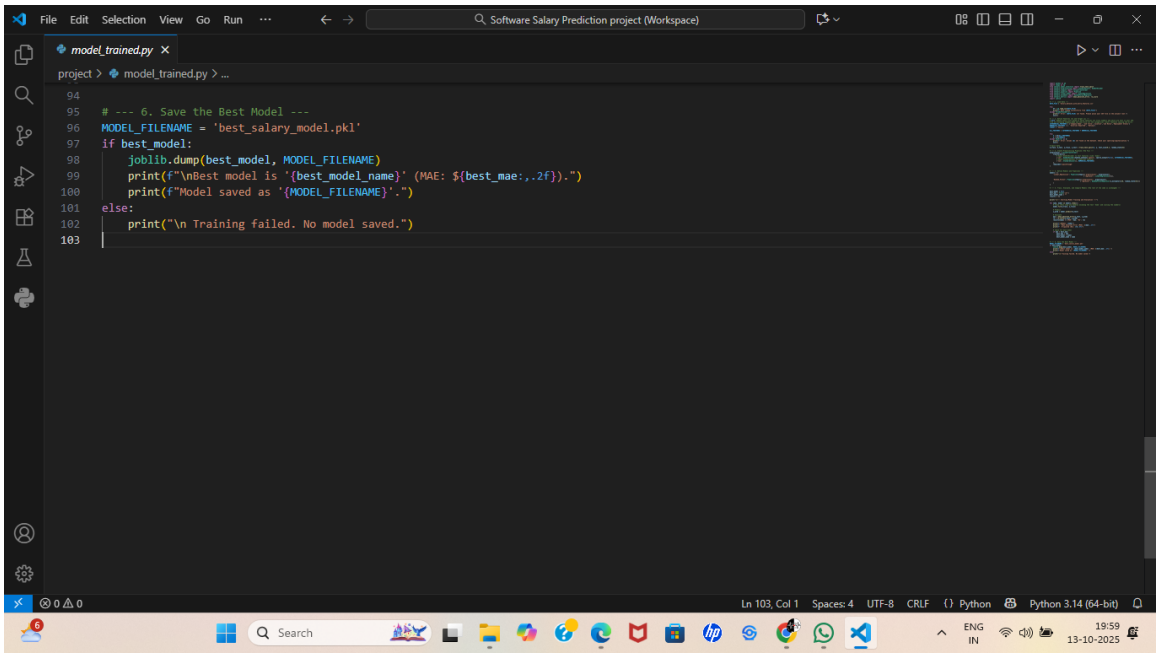
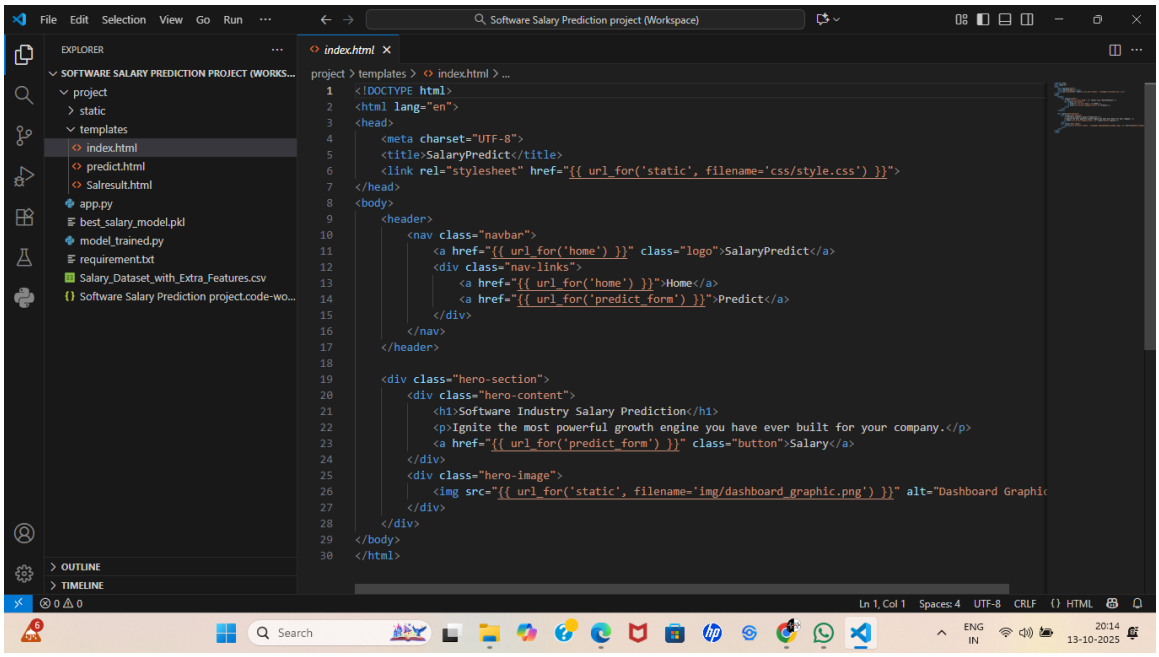
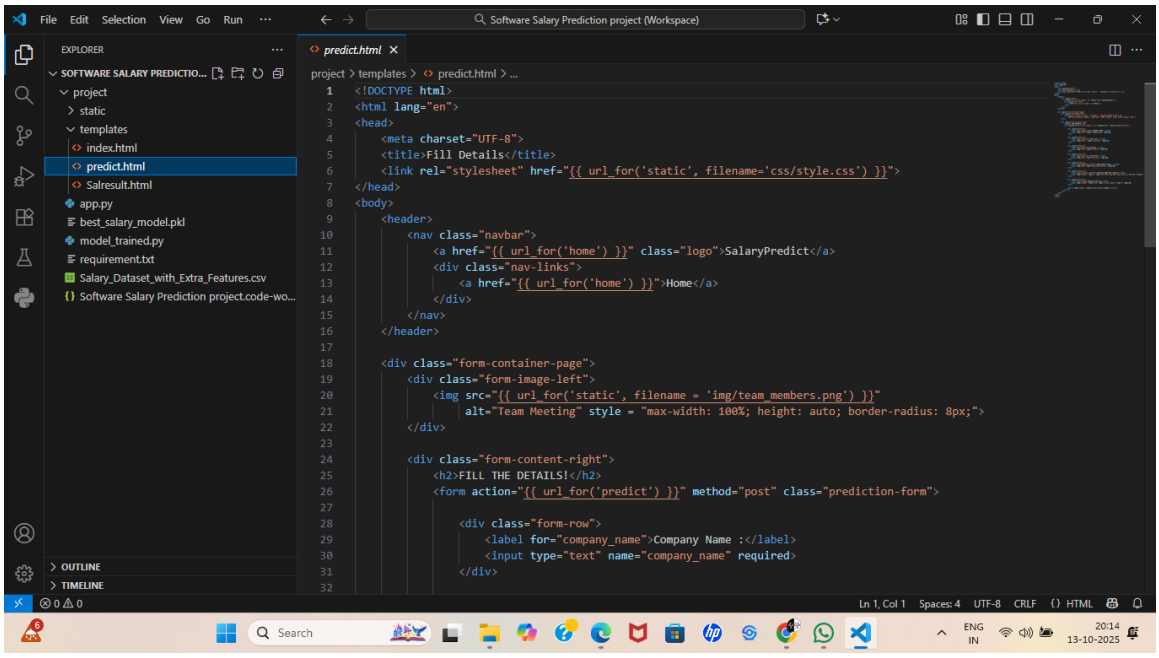


Figure 3: Screenshot of Index.html, predict.html,result.html.




File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
EXPLORER
SOFTWARE SALARY PREDICTIO...
project
static
templates
index.html
predict.html
Salresult.html
app.py
best_salary_model.pkl
model_trained.py
requirement.txt
Salary_Dataset_with_Extra_Features.csv
Software Salary Prediction project.code-wo...
predict.html
project > templates > predict.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Fill Details</title>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
<header>
<nav class="navbar">
SalaryPredict

Home

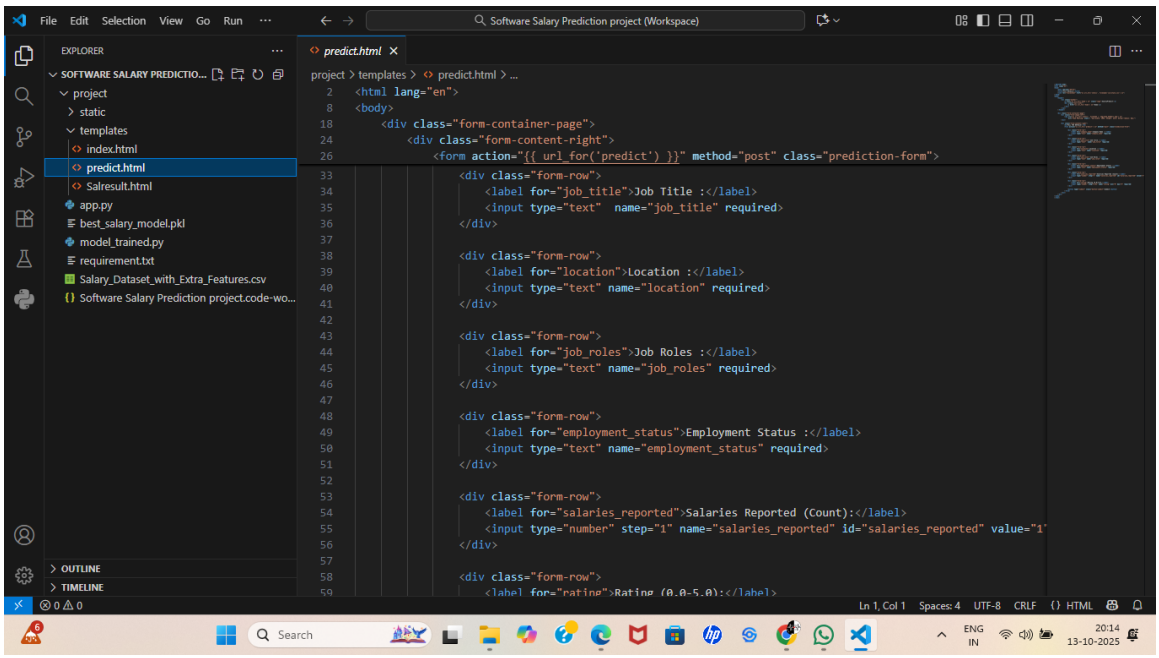
</nav>
</header>

<img src="{{ url_for('static', filename = 'img/team_members.png') }}"
alt="Team Meeting" style = "max-width: 100%; height: auto; border-radius: 8px;">

<h2>FILL THE DETAILS!</h2>
<form action="{{ url_for('predict') }}" method="post" class="prediction-form">

<label for="company_name">Company Name :</label>
<input type="text" name="company_name" required>

OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML
Search
ENG
IN
20:14
13-10-2025


File Edit Selection View Go Run
Software Salary Prediction project (Workspace)
EXPLORER
SOFTWARE SALARY PREDICTIO...
project
static
templates
index.html
predict.html
Salresult.html
app.py
best_salary_model.pkl
model_trained.py
requirement.txt
Salary_Dataset_with_Extra_Features.csv
Software Salary Prediction project.code-wo...
predict.html
project > templates > predict.html > ...
<html lang="en">
<body>

<form action="{{ url_for('predict') }}" method="post" class="prediction-form">

<label for="job_title">Job Title :</label>
<input type="text" name="job_title" required>

<label for="location">Location :</label>
<input type="text" name="location" required>

<label for="job_roles">Job Roles :</label>
<input type="text" name="job_roles" required>

<label for="employment_status">Employment Status :</label>
<input type="text" name="employment_status" required>

<label for="salaries_reported">Salaries Reported (Count):</label>
<input type="number" step="1" name="salaries_reported" id="salaries_reported" value="1"

OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF HTML
Search
ENG
IN
20:14
13-10-2025

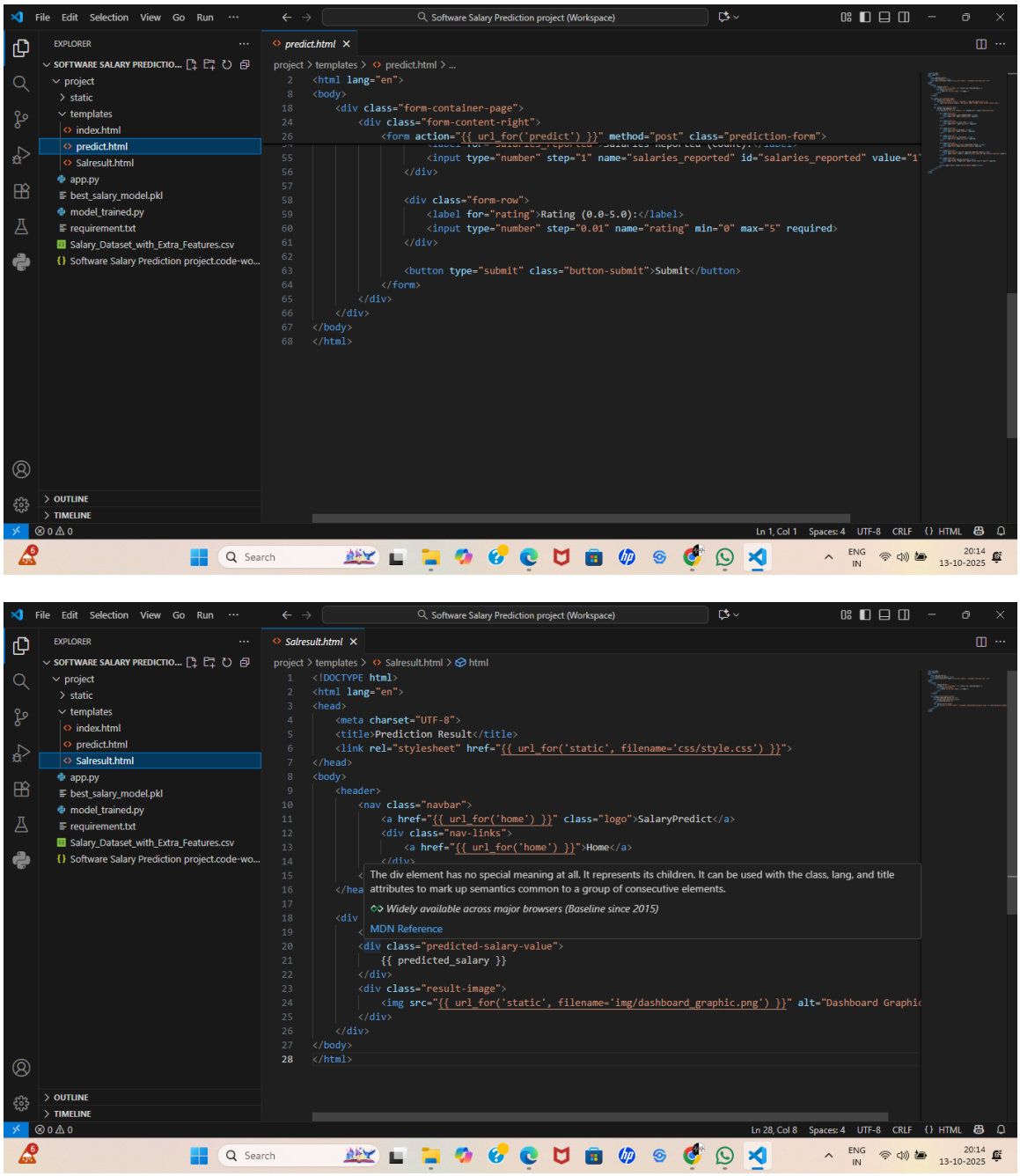


Figure 4: Web application -Index page

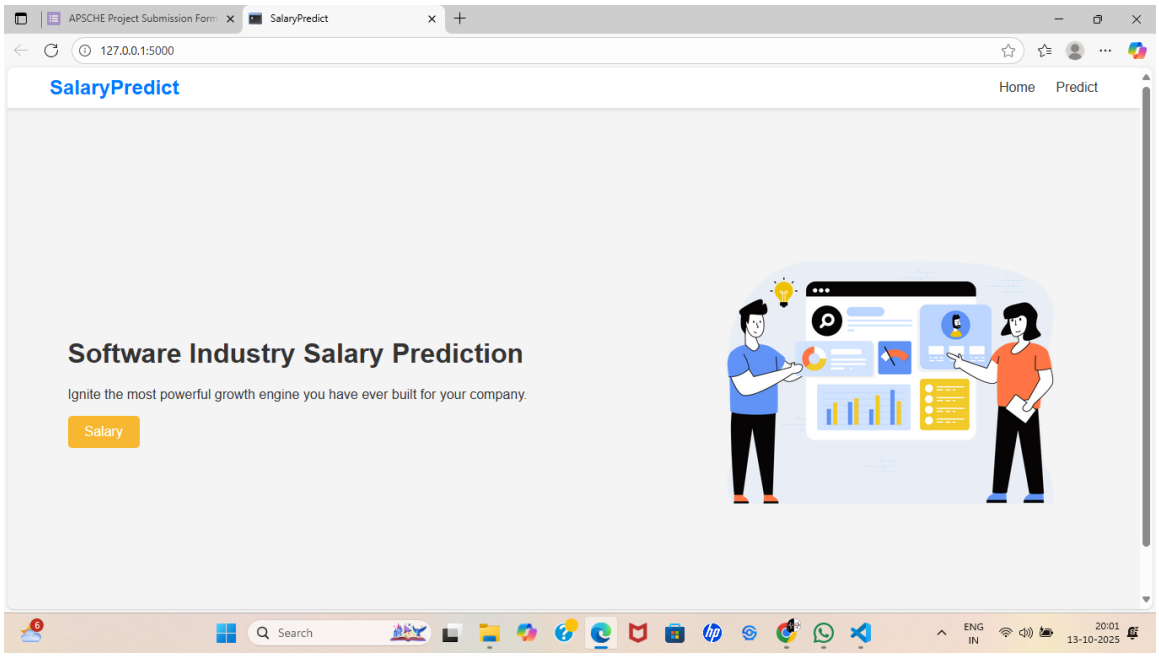


Figure 5: Web application- details page

R

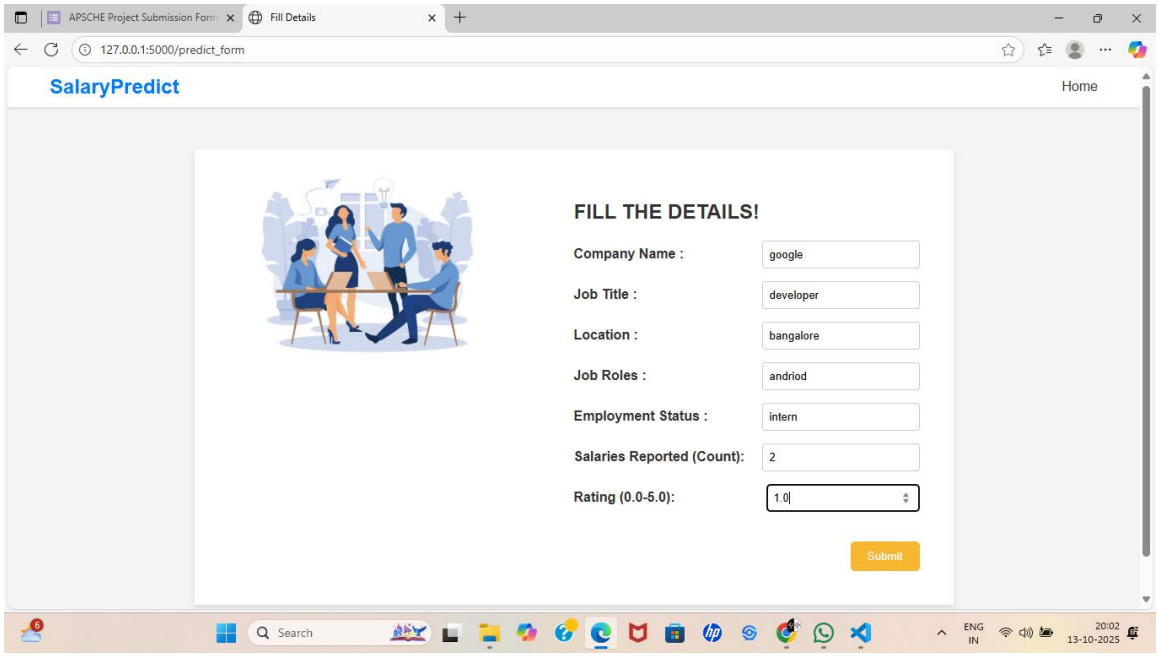
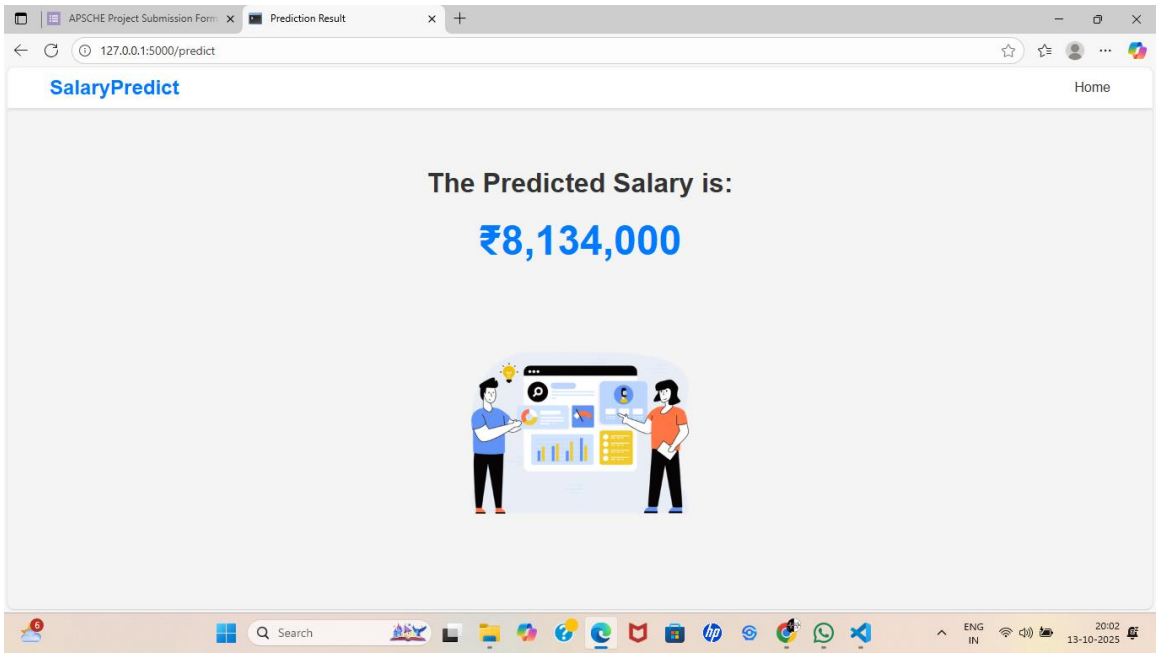


Figure 6: Web application- Result page.



12. References

• Flask documentation – https://flask.palletsprojects.com/

• Scikit-learn – https://scikit-learn.org/

• Python official documentation – https://docs.python.org/3/

• GitHub Repository - https://github.com/kasadhanalakshmi/Software-salary-prediction

• Demo Video - https://drive.google.com/file/d/1_KBOLCNdQi7tr5I-lM2pLotEnSbHrvxT/view?usp=sharing