LABORATORY REPORT

Application Development Lab (CS33002)

B.Tech Program in ECSc

Submitted By

Name:-AMAN

Roll No: 2230063

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Experiment Number	1
Experiment Title	Build a Resume using HTML/CSS
Date of Experiment	07/01/2025
Date of Submission	13/01/1025

1. Objective:-To design and develop a professional resume using HTML and CSS.

2. Procedure:- (Steps Followed)

- **Setup the Project**:

- Create a folder for your project.
- Inside the folder, create two files: `index.html` and `style.css`.

- **HTML Structure**:

- Open the `index.html` file and define the basic HTML structure:
    - Add the `DOCTYPE` declaration for HTML5.
    - Create the `html`, `head`, and `body` sections.
    - In the `head`, link to the CSS file using `<link rel="stylesheet" href="style.css">`.

- **Design the Layout**:

- Use a `div` with a class of `container` to hold the resume structure.
- Create two sections:
    - **Profile Section**:
        - Add a `div` with the class `profile` for the left-hand sidebar.
        - Include a `profile-photo` and `profile-info` for personal details.
    - **Resume Section**:
        - Add a `div` with the class `resume` for the main content.
        - Include child sections like `about`, `education`, `projects`, and `skills`.

- **Style the Components**:

- Open `style.css` to define styles for the template:
    - Set up a responsive grid using Flexbox in `.container`.
    - Add custom styles for colors, font sizes, and spacing.
    - Style specific sections like `.profile`, `.resume`, and their child elements.

- **Profile Section**:

- Use a `div` with the class `profile-photo` to display a profile image.
- Add personal details (e.g., name, contact information) in `profile-info`.
- Include a `languages` section with styled circular progress bars.

- **Resume Section**:
  - **About Section**:
    - Add the candidate's name, position, and location in styled headings.
  - **Education Section**:
    - Use a list (`<ul>`) with list items (`<li>`) for institutions, dates, and qualifications.
  - **Projects Section**:
    - Display a list of projects, each with a title, date, and description.
  - **Skills Section**:
    - Use horizontal progress bars to visually represent skill levels.
- **Make It Responsive**:
  - Add `@media` queries in `style.css` to handle different screen sizes:
    - Adjust the layout for tablets (`max-width: 768px`) and phones (`max-width: 480px`).
    - Ensure content stacks vertically on smaller devices.
- **Testing and Optimization**:
  - Open the `index.html` file in a web browser to view the resume.
  - Test the responsiveness by resizing the browser window.
  - Validate the HTML and CSS using online tools like W3C Validator.
- **Deployment**:
  - Host the project using GitHub Pages, Netlify, or any static web hosting service.
  - Ensure the profile image (`profile.png`) is available in the project directory.

3. Code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Resume Template</title>
  <style>
    html {
 box-sizing: border-box;
}
*, *:after, *:before {
 box-sizing: inherit;
```

```
}
.container {
  display: flex;
  max-width: 960px;
  background-color: #eaeaea;
  justify-content: space-between;
  margin: 20px auto;
  box-shadow: 1px 1px 10px rgba(0,0,0,0.1)
}
.profile {
  flex-basis: 35%;
  background-color: #39383a;
  color: #ababab;
}
.profile-photo {
  height: 270px;
  background-image: url(./profile.png);
  background-size: cover;
  background-position: top;
  background-repeat: no-repeat;
}
.profile-info {
  padding-left: 30px;
  padding-right: 30px;
  padding-top: 50px;
  padding-bottom: 70px;
}
.profile-text {
  font-size: 13px;
  line-height: 24.19px;
  margin-bottom: 50px;
}
.heading {
  margin: 0;
  padding-bottom: 16px;
  text-transform: uppercase;
  font-weight: 700;
}
.heading-light {
```

```
  color: #ffffff;
  border-bottom: 2px #5a5a5a dashed;
}
.contacts {
  margin-bottom: 70px;
}
.contacts-title {
  color: #fff;
  margin-bottom: 13px;
  font-size: 16px;
  font-weight: 400;
}
.contacts-text {
  color: #ababab;
  text-decoration: none;
  padding-left: 27px;
  line-height: 20.97px;
}
.contacts-item {
  padding-top: 24px;
  padding-bottom: 24px;
  border-bottom: 2px #5a5a5a dashed;
}
address {
  font-style: normal;
}
.fas {
  margin-right: 7px;
}
.languages {
  display: flex;
  flex-wrap: wrap;
  padding-top: 40px;
}
.language {
  width: 100px;
  height: 100px;
  border: 6px solid #5c5c5c;
  border-radius: 50%;
```

```
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  margin-bottom: 30px;
  margin-right: 30px;
}
.language:nth-child(3) {
  margin-bottom: 0;
}
.language-text {
  text-transform: uppercase;
  font-size: 11px
}
.languages-per {
  font-size: 15px;
  font-weight: 600;
}
.lines {
  display: flex;
  flex-direction: column;
  justify-content: center;
}
.line {
  display: block;
  width: 90px;
  height: 2px;
  background-color: #5c5c5c;
  margin-top: 10px;
  margin-bottom: 10px;
}
.line:nth-child(2) {
  width: 100px;
  margin-left: 20px;
}
.resume {
  padding: 25px 30px;
  flex-basis: 63%;
  background-color: #fff;
```

```
}
.resume-wrap {
  padding: 36px 56px;
  border: 1px solid rgba(168, 168, 168, 0.44);
  min-height: 100%;
}
.logo {
  display: flex;
  justify-content: center;
  margin-bottom: 38px;
}
.logo-img {
  width: 90px;
  height: 90px;
  border: 1px solid #39383a;
  border-radius: 50%;
  display: flex;
  justify-content: center;
  align-items: center;
}
.logo-lines {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  margin-left: 17px;
  margin-right: 17px;
}
.logo-line {
  width: 43px;
  height: 2px;
  background-color: #39383a;
  margin-top: 10px;
  margin-bottom: 10px;
}
.logo-lines_left .logo-line:nth-child(2) {
  margin-right: 20px;
  width: 55px;
}
```

```
.logo-lines_right .logo-line:nth-child(2) {
  margin-left: 20px;
  width: 55px;
}
.about {
  padding-bottom: 30px;
  border-bottom: 1px solid #e0e0e0;
  text-align: center;
  margin-bottom: 40px;
}
.name {
  font-size: 16px;
  text-transform: uppercase;
  letter-spacing: 10.75px;
  margin-bottom: 10px;
}
.position {
  display: inline-block;
  font-size: 9px;
  text-transform: uppercase;
  color: #808080;
  margin-bottom: 30px;
}
.about-address {
  font-size: 13px;
  margin-bottom: 15px;
  font-family: Roboto;
}
.about-contacts {
  font-size: 8px;
}
.about-contacts__link {
  text-decoration: none;
  color: #777777;
}
.heading_dark {
  font-size: 16px;
  font-weight: 400;
  border-bottom: 1px solid #e0e0e0;
```

```
  margin-bottom: 37px;
}
.list {
  list-style: none;
  padding-left: 0;
}
.list-item {
  position: relative;
  padding-left: 40px;
  padding-bottom: 30px;
  margin-bottom: 30px;
  border-bottom: 2px dashed #ececec;
}
.list-item:before {
  content: '';
  position: absolute;
  left: 0;
  top: 3px;
  width: 9px;
  height: 9px;
  border-radius: 50%;
  background-color: #000;
}
.list-item__title {
  font-size: 11px;
  text-transform: uppercase;
  margin-bottom: 5px;
}.list-item__date {
  font-size: 10px;
  text-transform: uppercase;
}
.list-item__text {
  font-size: 10px;
  color: #777;
}
.list-item_non-border {
  border: none;
}
.heading_skills {
```

```
  margin-bottom: 48px;
}
.skills-list {
  list-style-type: none;
  padding-left: 0;
}
.skills-list__item {
  margin-bottom: 30px;
  text-transform: uppercase;
  font-size: 11px;
  display: flex;
  justify-content: space-between;
}
.level {
  width: 70%;
  height: 8px;
  border: 1px solid #39383a;
  position: relative;
}
.level:before {
  content: '';
  position: absolute;
  left: 0;
  top: 0;
  height: 100%;
  background-color: #898989;
}
.level-80:before {
  width: 80%;
}
.level-90:before {
  width: 90%;
}
.level-50:before {
  width: 50%;
}

@media (max-width: 1024px) {
  .container {
```

```
    width: 90%;
  }
}

@media (max-width: 992px) {
  .container {
    flex-direction: column;
    width: 70%;
  }
  .profile-photo {
    width: 200px;
    height: 200px;
    border: 3px solid #fff;
    margin: auto;
    margin-top: 40px;
  }
  .profile {
    position: relative;
  }
  .profile:before {
    content: '';
    width: 100%;
    height: 150px;
    background-color: #03A9F4;
    display: block;
    position: absolute;
  }
  .profile-photo {
    position: relative;
    z-index: 0;
  }
  .lines {
    display: none;
  }
}
@media (max-width: 768px) {
  .container {
    width: 80%;
  }
```

```
  .resume {
    padding: 10px;
  }
  .resume-wrap {
    padding-left: 20px;
    padding-right: 20px;
  }
  .list-item__title {
    font-size: 14px;
  }
  .list-item__date {
    font-size: 12px;
  }
  .list-item__text {
    font-size: 12px;
    line-height: 1.4;
  }
  .about-contacts__link {
    display: block;
    font-size: 13px;
  }
}
@media (max-width: 567px) {
  .logo-img {
    width: 70px;
    height: 70px;
  }
  .logo-lines {
    margin-left: 8px;
    margin-right: 8px;
  }
}
@media (max-width: 480px) {
  .logo {
    display: none;
  }
  .container {
    min-width: 320px;
  }
```

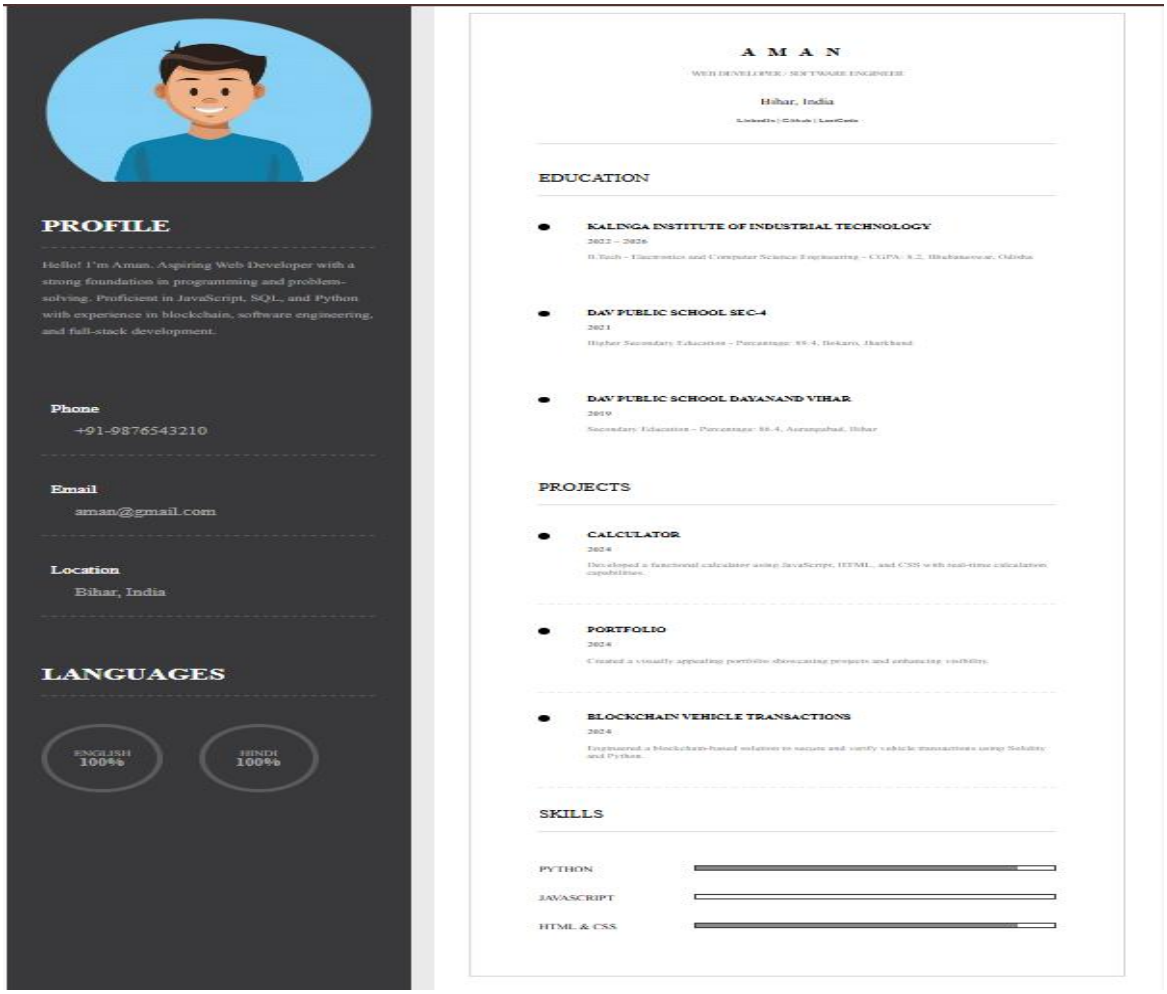
```
  .name {
    letter-spacing: normal;
  }
  .level {
    width: 50%;
  }

}
    </style>
</head>
<body>
    <div class="container">
        <div class="profile">
          <div class="profile-photo"></div>
          <div class="profile-info">
            <h2 class="heading heading-light">
              Profile
            </h2>
            <p class="profile-text">
              Hello! I’m Aman. Aspiring Web Developer with a strong foundation in programming and problem-solving.
              Proficient in JavaScript, SQL, and Python with experience in blockchain, software engineering, and full-stack development.
            </p>
            <div class="contacts">
              <div class="contacts-item">
                <h3 class="contacts-title">
                  <i class="fas fa-phone-volume"></i>
                  Phone
                </h3>
                <a href="tel:+919876543210" class="contacts-text">+91-9876543210</a>
              </div>
              <div class="contacts-item">
                <h3 class="contacts-title">
                  <i class="fas fa-envelope"></i>
                  Email
                </h3>
                <a href="mailto:aman@example.com" class="contacts-text">aman@gmail.com</a>
              </div>
              <div class="contacts-item">
```

```
      <h3 class="contacts-title">
        <i class="fas fa-map-marker-alt"></i>
        Location
      </h3>
      <address class="contacts-text">
        Bihar, India
      </address>
    </div>
  </div>
  <h2 class="heading heading-light">Languages</h2>
  <div class="languages">
    <div class="language">
      <span class="language-text">English</span>
      <strong class="languages-per">100%</strong>
    </div>
    <div class="language">
      <span class="language-text">Hindi</span>
      <strong class="languages-per">100%</strong>
    </div>
  </div>
 </div>
</div>
<div class="resume">
 <div class="resume-wrap">
  <div class="about">
    <h1 class="name">Aman</h1>
    <span class="position">Web Developer / Software Engineer</span>
    <address class="about-address">Bihar, India</address>
    <div class="about-contacts">
      <a class="about-contacts__link" href="https://www.linkedin.com">
        <b>LinkedIn</b>
      </a> |
      <a class="about-contacts__link" href="https://github.com/">
        <b>Github</b>
      </a> |
      <a class="about-contacts__link" href="https://leetcode.com/">
        <b>LeetCode</b>
      </a>
    </div>
```

```
        </div>
        <div class="education">
          <h2 class="heading heading_dark">
            Education
          </h2>
          <ul class="list">
            <li class="list-item list-item_non-border">
              <h4 class="list-item__title">Kalinga Institute of Industrial Technology</h4>
              <span class="list-item__date">2022 – 2026</span>
              <p class="list-item__text">B.Tech - Electronics and Computer Science Engineering - CGPA: 8.2, Bhubaneswar, Odisha</p>
            </li>
            <li class="list-item list-item_non-border">
              <h4 class="list-item__title">DAV Public School Sec-4</h4>
              <span class="list-item__date">2021</span>
              <p class="list-item__text">Higher Secondary Education - Percentage: 89.4, Bokaro, Jharkhand</p>
            </li>
            <li class="list-item list-item_non-border">
              <h4 class="list-item__title">DAV Public School Dayanand Vihar</h4>
              <span class="list-item__date">2019</span>
              <p class="list-item__text">Secondary Education - Percentage: 86.4, Aurangabad, Bihar</p>
            </li>
          </ul>
        </div>
        <div class="projects">
          <h2 class="heading heading_dark">
            Projects
          </h2>
          <ul class="list">
            <li class="list-item">
              <h4 class="list-item__title">Calculator</h4>
              <span class="list-item__date">2024</span>
              <p class="list-item__text">Developed a functional calculator using JavaScript, HTML, and CSS with real-time calculation capabilities.</p>
            </li>
            <li class="list-item">
              <h4 class="list-item__title">Portfolio</h4>
              <span class="list-item__date">2024</span>
              <p class="list-item__text">Created a visually appealing portfolio showcasing projects and enhancing visibility.</p>
```

```
            </li>
            <li class="list-item">
              <h4 class="list-item__title">Blockchain Vehicle Transactions</h4>
              <span class="list-item__date">2024</span>
              <p class="list-item__text">Engineered a blockchain-based solution to secure and verify vehicle transactions using Solidity and Python.</p>
            </li>
          </ul>
        </div>
        <div class="skills">
          <h2 class="heading heading_dark heading_skills">
            Skills
          </h2>
          <ul class="skills-list">
            <li class="skills-list__item">
              Python
              <div class="level level-90"></div>
            </li>
            <li class="skills-list__item">
              JavaScript
              <div class="level level-85"></div>
            </li>
            <li class="skills-list__item">
              HTML & CSS
              <div class="level level-90"></div>
            </li>
          </ul>
        </div>
      </div>
    </div>
  </div>

</body>
</html>
```

4. Results/Output:- Entire Screen Shot including Date & Time



PROFILE

Hello! I'm Aman. Aspiring Web Developer with a strong foundation in programming and problem-solving. Proficient in JavaScript, SQL, and Python with experience in blockchain, software engineering, and full-stack development.

Phone
+91-9876543210

Email
aman@gmail.com

Location
Bihar, India

LANGUAGES

ENGLISH 100%

HINDI 100%

A M A N

Bihar, India

EDUCATION

- KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
- DAV PUBLIC SCHOOL SEC-4
- DAV PUBLIC SCHOOL DAYANAND VIHAR

PROJECTS

- CALCULATOR
- PORTFOLIO
- BLOCKCHAIN VEHICLE TRANSACTIONS

SKILLS

PYTHON

JAVASCRIPT

HTML & CSS

5. Remarks:-

Signature of the Student

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)

Experiment Number	2
Experiment Title	To classify images as cats or dogs using machine learning models.
Date of Experiment	14/01/2025
Date of Submission	21/01/1025

1.Objective:-To classify images as cats or dogs using machine learning models.

2. Procedure:-

Download the Dataset:

1. Obtain a dataset of cat and dog images, e.g., from Kaggle.
2. Unzip the dataset if necessary.

Organize the Dataset:

1.Place cat images in the `cats` folder and dog images in the `dogs` folder.

Verify the Path in Code:

1. Ensure the `data_dir` variable in the script matches the dataset path.
   Example: If the dataset is located in `C:/Users/KIIT/Desktop/AD/Lab2/data/train`, set `data_dir` accordingly.

Add Error Handling:

1. Update the script to check if the required folders (`cats` and `dogs`) exist.
2. Raise an appropriate error message if they do not.

Check File Formats:

1. Ensure all images in the `cats` and `dogs` folders are valid image files (e.g., `.jpg`, `.png`).

Verify Dataset Access:

1. Use a simple script to print the number of files in each folder to ensure the data is correctly placed and accessible.

Run the Model Training Script:

1. Execute the training script (`train_models.py`) to preprocess the data and train models.

Save Trained Models:

1. Ensure the trained models are saved (e.g., `svm_model.pkl`, `cnn_model.h5`) in the specified location.

Start Flask Backend:

1. Run the Flask app to serve the trained models and handle predictions.

Test the Frontend:

- Upload an image via the frontend UI and select a model for prediction.
- Verify that the output correctly identifies the uploaded image as a cat or a dog.

Code -

Training of model-

```
import os
import pickle
import cv2
import numpy as np
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
from keras.api.models import Sequential
from keras.api.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Load dataset
def load_data(data_dir):
    X, y = [], []
    for label, class_dir in enumerate(['cats', 'dogs']):
        class_path = os.path.join(data_dir, class_dir)
        for img_name in os.listdir(class_path):
            img_path = os.path.join(class_path, img_name)
            img = cv2.imread(img_path, cv2.IMREAD_COLOR)
            img = cv2.resize(img, (64, 64))  # Resize images
            X.append(img.flatten())  # Flatten image
            y.append(label)  # 0 for cat, 1 for dog
    return np.array(X), np.array(y)
```

```
# Train SVM
def train_svm(X, y):
    model = SVC(kernel='linear', probability=True)
    model.fit(X, y)
    with open('backend/models/svm_model.pkl', 'wb') as f:
        pickle.dump(model, f)
```

```
# Train Random Forest
def train_random_forest(X, y):
    model = RandomForestClassifier(n_estimators=100)
    model.fit(X, y)
    with open('backend/models/random_forest.pkl', 'wb') as f:
        pickle.dump(model, f)
```

```
# Train Logistic Regression
def train_logistic_regression(X, y):
    model = LogisticRegression()
    model.fit(X, y)
    with open('backend/models/logistic_regression.pkl', 'wb') as f:
        pickle.dump(model, f)
```

```
# Train K-Means
def train_kmeans(X):
    model = KMeans(n_clusters=2)
    model.fit(X)
    with open('backend/models/kmeans_model.pkl', 'wb') as f:
```

```
        pickle.dump(model, f)
```

```
# Train CNN
def train_cnn(X, y):
    X = X.reshape(-1, 64, 64, 3) / 255.0  # Normalize and reshape
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(X, y, epochs=10, batch_size=32, validation_split=0.2)
    model.save('backend/models/cnn_model.h5')
```

```
if __name__ == '__main__':
    data_dir = 'data/train'
    X, y = load_data(data_dir)
    train_svm(X, y)
    train_random_forest(X, y)
    train_logistic_regression(X, y)
    train_kmeans(X)
    train_cnn(X, y)
    print("All models trained and saved.")
```

App.py

```
from flask import Flask, request, render_template, jsonify
import os
import pickle
import cv2
import numpy as np
from keras.api.models import load_model

app = Flask(__name__)
UPLOAD_FOLDER = 'backend/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
# Load models
MODELS = {
    'svm': pickle.load(open('backend/models/svm_model.pkl', 'rb')),
    'random_forest': pickle.load(open('backend/models/random_forest.pkl', 'rb')),
    'logistic_regression': pickle.load(open('backend/models/logistic_regression.pkl', 'rb')),
    'kmeans': pickle.load(open('backend/models/kmeans_model.pkl', 'rb')),
    'cnn': load_model('backend/models/cnn_model.h5')
}
```

```
@app.route('/')
def index():
    return render_template('index.html')
```

```
@app.route('/predict', methods=['POST'])
def predict():
    if 'image' not in request.files:
        return jsonify({'error': 'No file uploaded'}), 400
```

```
    file = request.files['image']
    model_name = request.form['model']
```

```
    if file and model_name in MODELS:
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(filepath)
```

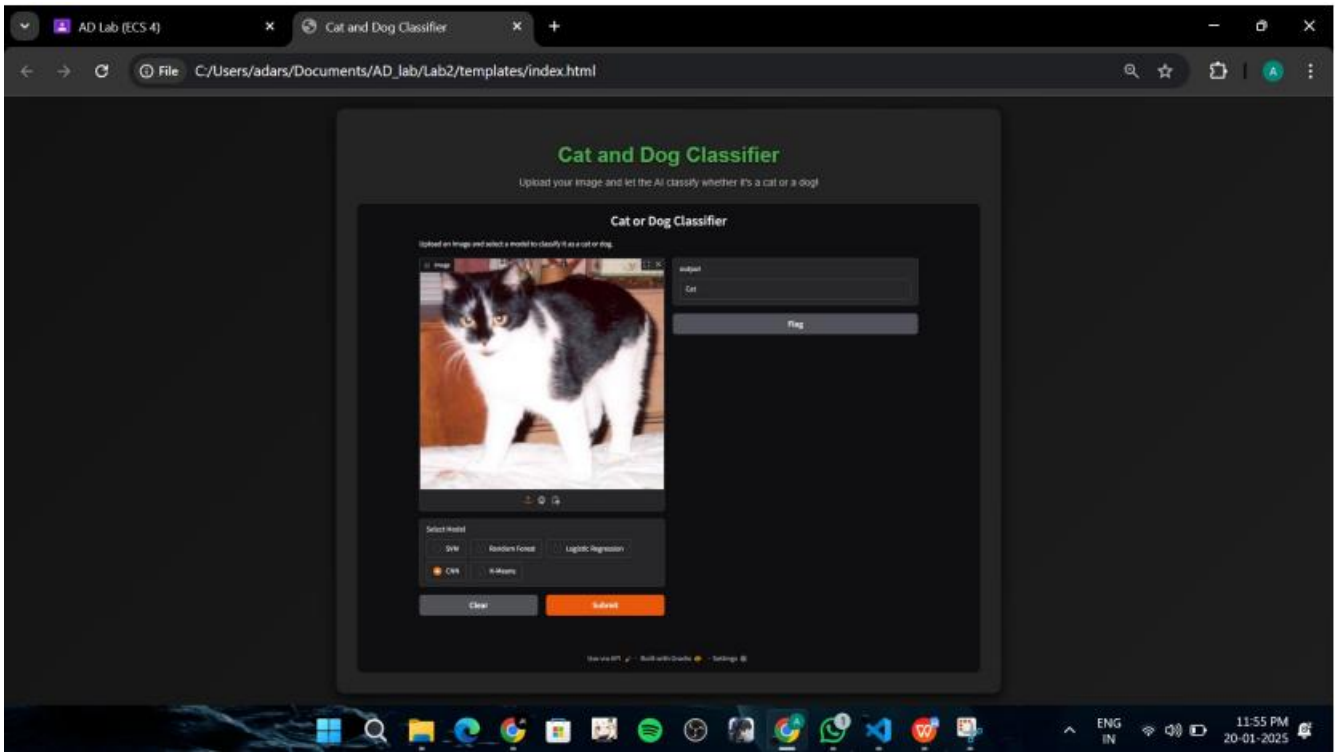
```
        img = cv2.imread(filepath)
        img = cv2.resize(img, (64, 64)).flatten() / 255.0
        img = np.array([img])
```

```
        model = MODELS[model_name]
        if model_name == 'cnn':
            img = img.reshape(-1, 64, 64, 3)
            prediction = model.predict(img)
            label = 'Cat' if prediction[0] < 0.5 else 'Dog'
        elif model_name == 'kmeans':
            cluster = model.predict(img)
            label = 'Cat' if cluster[0] == 0 else 'Dog'
        else:
            prediction = model.predict(img)
            label = 'Cat' if prediction[0] < 0.5 else 'Dog'
```

```
        return jsonify({'prediction': label})
    return jsonify({'error': 'Invalid request'}), 400
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

Results/Output:-



6. Remarks:-

Signature of the Student

(Name of the Student)

Signature of the Lab Coordinator

(Name of the Coordinator)

Experiment Number	3
Experiment Title	Regression Analysis for Stock Prediction
Date of Experiment	
Date of Submission	

1. Objective:-

2. Procedure:-

Procedure:

1. Collect historical stock price data.
2. Preprocess the data for analysis (missing data, scaling, splitting into train/test)
3. . Implement Linear Regression to predict future stock prices.
4. Design and train an LSTM model for time-series prediction.
5. Compare the accuracy of both models.
6. Create a Flask backend for model predictions.
7. Build a frontend to visualize predictions using charts and graphs.

3. Code:-

Linear Regression-

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import os

def linear_regression_model():
    # File path for stock data
    file_path = 'C:\\Users\\KIIT\\Desktop\\AD\\Lab3\\data\\stock_data.csv'

    # Check if the file exists
    if not os.path.exists(file_path):
        raise FileNotFoundError(f"File not found: {file_path}")

    # Load data
    data = pd.read_csv(file_path)

    # Ensure necessary columns exist
    required_columns = ['Open', 'High', 'Low', 'Close']
    for col in required_columns:
        if col not in data.columns:
            raise ValueError(f"Missing required column: {col}")

    # Feature selection (X) and target variable (y)
    X = data[['Open', 'High', 'Low']]  # Using Open, High, and Low as features
    y = data['Close']  # Using Close price as the target variable

    # Train-test split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
    # Model initialization
    model = LinearRegression()

    # Training the model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)

    # Evaluation metrics
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    accuracy = r2 * 100  # Accuracy as a percentage

    # Visualization
    plt.figure(figsize=(10, 6))
    plt.scatter(range(len(y_test)), y_test, color="blue", label="Actual Values", alpha=0.6)
    plt.scatter(range(len(y_pred)), y_pred, color="red", label="Predicted Values", alpha=0.6)
    plt.title("Actual vs Predicted Values")
    plt.xlabel("Data Points")
    plt.ylabel("Close Price")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig('output_visualization.png')  # Save the graph as an image
    plt.show()

    # Return results
    results = {
        "message": "Linear regression model executed successfully.",
        "model_coefficients": model.coef_.tolist(),  # Coefficients of the features
        "model_intercept": model.intercept_,  # Intercept of the regression line
        "mean_squared_error": mse,
        "r2_score": r2,
        "accuracy": accuracy,
        "sample_predictions": {
            "actual": y_test.tolist()[:5],  # First 5 actual values
            "predicted": y_pred.tolist()[:5]  # First 5 predicted values
        }
    }

    return results

if __name__ == '__main__':
    try:
        results = linear_regression_model()
        print("Linear Regression Results:")
        print(f"Mean Squared Error: {results['mean_squared_error']}")
        print(f"R2 Score: {results['r2_score']}")
        print(f"Accuracy: {results['accuracy']}%")
    except FileNotFoundError as e:
        print(e)
    except ValueError as e:
        print(e)
```

LSTM Model-

```
import pandas as pd
import numpy as np
```

```
from keras.api.models import Sequential
from keras.api.layers import Dense, LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

def create_dataset(data, look_back=1):
    X, Y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:(i + look_back), 0])
        Y.append(data[i + look_back, 0])
    return np.array(X), np.array(Y)

def lstm_model():
    # Load the dataset
    data = pd.read_csv('C:\\Users\\KIIT\\Desktop\\AD\\Lab3\\data\\stock_data.csv')
    data['Date'] = pd.to_datetime(data['Date'])
    data.set_index('Date', inplace=True)

    # Normalize the Close prices
    scaler = MinMaxScaler()
    data['Close'] = scaler.fit_transform(data['Close'].values.reshape(-1, 1))

    # Split data into training and testing
    train_size = int(len(data) * 0.8)
    train_data = data[:train_size]
    test_data = data[train_size:]

    # Prepare data for LSTM
    look_back = 10
    train_scaled = train_data['Close'].values.reshape(-1, 1)
    test_scaled = test_data['Close'].values.reshape(-1, 1)

    X_train, y_train = create_dataset(train_scaled, look_back)
    X_test, y_test = create_dataset(test_scaled, look_back)

    X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
    X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(50, return_sequences=True, input_shape=(look_back, 1)))
    model.add(LSTM(50))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mean_squared_error')

    # Train the model
    history = model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)

    # Make predictions
    predictions = model.predict(X_test)
    predictions = scaler.inverse_transform(predictions)
    y_test = scaler.inverse_transform(y_test.reshape(-1, 1))

    # Calculate metrics
    mse = mean_squared_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)

    # Visualization: Actual vs Predicted Close Prices
```

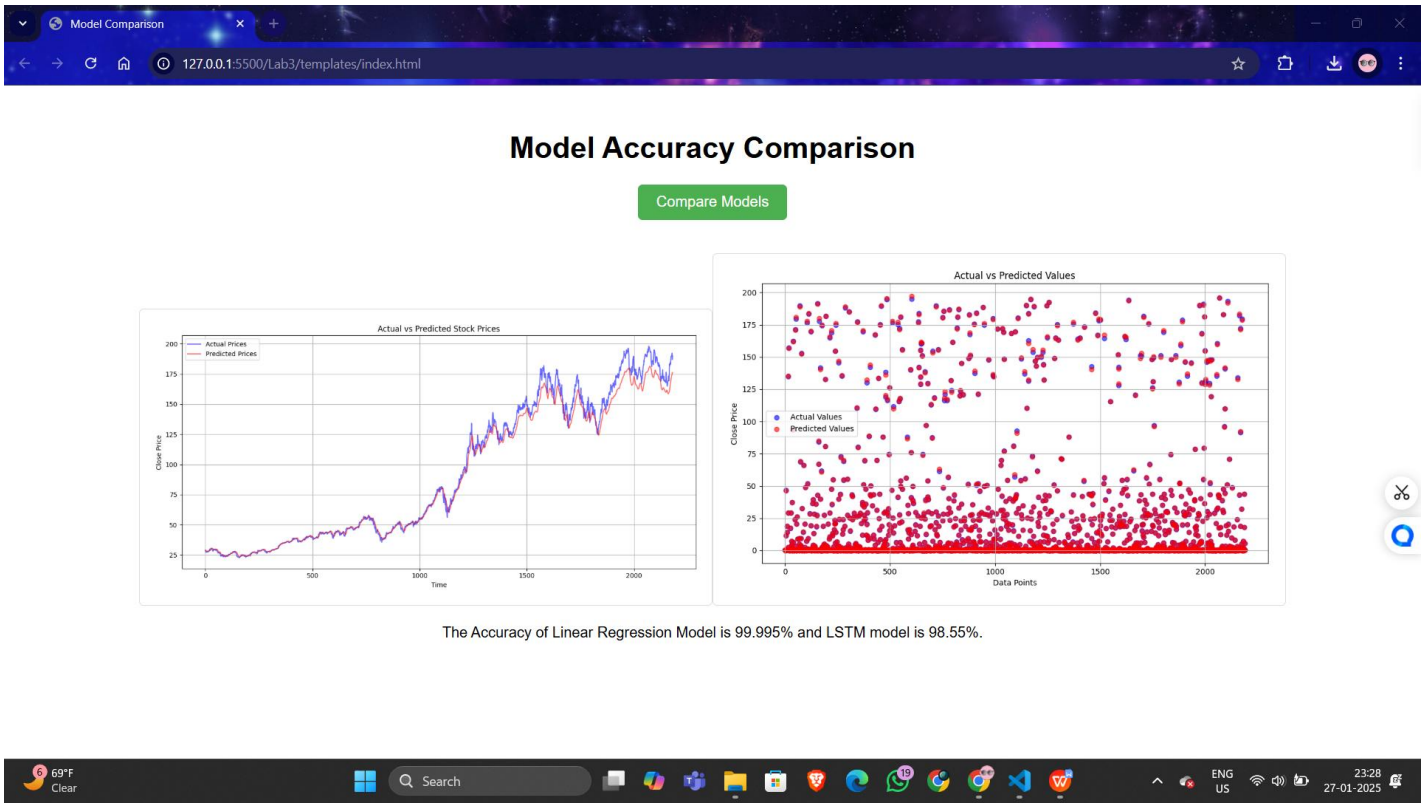
```
    plt.figure(figsize=(12, 6))
    plt.plot(y_test, label="Actual Prices", color='blue', alpha=0.6)
    plt.plot(predictions, label="Predicted Prices", color='red', alpha=0.6)
    plt.title("Actual vs Predicted Stock Prices")
    plt.xlabel("Time")
    plt.ylabel("Close Price")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig("actual_vs_predicted.png")  # Save the graph as an image
    plt.show()

    # Visualization: Training Loss
    plt.figure(figsize=(10, 4))
    plt.plot(history.history['loss'], label="Training Loss", color='green')
    plt.title("Model Training Loss Over Epochs")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.savefig("training_loss.png")  # Save the graph as an image
    plt.show()

    return {
        "MSE": mse,
        "R2": r2,
        "Predictions": predictions.flatten().tolist()
    }

if __name__ == '__main__':
    results = lstm_model()
    print("LSTM Results:")
    print(f"Mean Squared Error: {results['MSE']}")
    print(f"R2 Score: {results['R2']}")
```

APP-

```
from flask import Flask, render_template, jsonify, send_from_directory
from flask_cors import CORS
import os
import matplotlib
import matplotlib.pyplot as plt
from linear_regression import linear_regression_model
from lstm_model import lstm_model
import traceback

app = Flask(__name__)
CORS(app)

def create_accuracy_chart(lr_accuracy, lstm_accuracy):
    try:
        # Data for the chart
        models = ['Linear Regression', 'LSTM']
        accuracies = [lr_accuracy, lstm_accuracy]

        # Create a bar chart
        plt.figure(figsize=(6, 4))
        plt.bar(models, accuracies, color=['blue', 'green'])
```

```
        plt.title('Model Accuracy Comparison')
        plt.xlabel('Models')
        plt.ylabel('Accuracy (%)')
        plt.ylim(0, 100)

        # Save the chart
        chart_path = os.path.join('static', 'graphs', 'accuracy_comparison.png')
        # Ensure the directory exists
        os.makedirs(os.path.dirname(chart_path), exist_ok=True)
        plt.savefig(chart_path)
        plt.close()

        return chart_path
    except Exception as e:
        print(f"Error generating chart: {e}")
        return None

# Route to serve the frontend
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/compare-models', methods=['GET'])
def compare_models():
    try:
            lr_results = linear_regression_model()
        lstm_results = lstm_model()

        # Calculate accuracy percentages
        lr_accuracy = lr_results["accuracy"] * 100  # Assuming accuracy is already in percentage form
        lstm_accuracy = lstm_results["R2"] * 100  # Assuming R2 is converted to percentage form

        # Generate the comparison chart
        chart_path = create_accuracy_chart(lr_accuracy, lstm_accuracy)

        if not chart_path:
            return jsonify({"error": "Error generating comparison chart"}), 500

        # Respond with the chart's URL
        response = {
            "chart_url": f"/static/graphs/accuracy_comparison.png"
        }
        return jsonify(response)

    except Exception as e:
        print(f"Error: {e}")
        print(traceback.format_exc())  # Log full traceback
        return jsonify({"error": str(e)}), 500

# Route to serve static files (like graphs)
@app.route('/static/<path:filename>', methods=['GET'])
def serve_static(filename):
    static_dir = os.path.join(os.getcwd(), 'static')
    return send_from_directory(static_dir, filename)

if __name__ == '__main__':
    os.makedirs('static/graphs', exist_ok=True)
    app.run(debug=True)
```

4. Results/Output:-



5. Remarks:-

Signature of the Lab Coordinator

Prof. Bhragav Appasani
