Low-Level Document: Placement Predictor Web Application

1. Project Structure

1.1. Directory Layout

```
placement_predictor/
│
├── app.py              # Flask application entry point
├── models/
│   ├── random_forest_model.pkl  # Random Forest model
│   ├── svm_model.pkl           # SVM model
│   └── logistic_regression_model.pkl  # Logistic Regression model
├── static/
│   └── style.css        # CSS file for styling
├── templates/
│   └── index.html       # Main HTML template
├── requirements.txt    # Python dependencies
├── Procfile            # Deployment instructions
└── runtime.txt          # Python version (optional)
```

A low-level document for your Placement Predictor project provides detailed information about the system components, code structure, and implementation specifics. It serves as a comprehensive guide for developers and technical teams who need to understand the inner workings of the project.

Low-Level Document: Placement Predictor Web Application

1. Project Structure

1.1. Directory Layout

```
graphql
Copy code
placement_predictor/
│
├── app.py               # Flask application entry point
├── models/
│   ├── random_forest_model.pkl  # Random Forest model
│   ├── svm_model.pkl           # SVM model
│   └── logistic_regression_model.pkl  # Logistic Regression model
├── static/
│   └── style.css        # CSS file for styling
├── templates/
│   └── index.html       # Main HTML template
├── requirements.txt    # Python dependencies
├── Procfile            # Deployment instructions
└── runtime.txt          # Python version (optional)
```

2. Flask Application (app.py)

2.1. Code Overview

```
from flask import Flask, request, jsonify, render_template
import joblib
import numpy as np
import pandas as pd

app = Flask(__name__)

# Load models
rf_model = joblib.load('models/random_forest_model.pkl')
svm_model = joblib.load('models/svm_model.pkl')
log_reg_model = joblib.load('models/logistic_regression_model.pkl')

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

@app.route('/predict', methods=['POST'])
def predict():
    data = request.json
    # Convert data to DataFrame and ensure correct format
    input_df = pd.DataFrame([data])

    # Predict with models
    rf_prediction = rf_model.predict(input_df)[0]
    svm_prediction = svm_model.predict(input_df)[0]
    log_reg_prediction = log_reg_model.predict(input_df)[0]

    # Map prediction to labels
    prediction_labels = {0: 'Not Placed', 1: 'Placed'}
    result = {
        'Random Forest': prediction_labels[rf_prediction],
        'SVM': prediction_labels[svm_prediction],
        'Logistic Regression': prediction_labels[log_reg_prediction]
    }

    return jsonify(result)

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

2.2. Explanation

- **Imports:** Necessary libraries are imported including Flask for web application, joblib for model loading, and pandas for data manipulation.
- **Model Loading:** Models are loaded from the `models` directory.
- **Routes:**
    - `/`: Renders the HTML form for user input.
    - `/predict`: Receives input data, performs predictions using the loaded models, and returns the results as JSON.

3. HTML Template (index.html)

3.1. Code Overview

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>ML Model Prediction</title>
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
   <div class="container">
      <h1>Placement Predictor</h1>
      <form id="predictionForm">
         <!-- Input fields -->
         <label for="gender">Gender:</label>
         <select id="gender" name="gender">
            <option value="0">Male</option>
            <option value="1">Female</option>
         </select>

         <label for="ssc_p">SSC Percentage:</label>
         <input type="number" id="ssc_p" name="ssc_p" step="0.01" required>

         <label for="hsc_p">HSC Percentage:</label>
         <input type="number" id="hsc_p" name="hsc_p" step="0.01" required>

         <label for="degree_p">Degree Percentage:</label>
         <input type="number" id="degree_p" name="degree_p" step="0.01" required>

         <label for="etest_p">E-Test Percentage:</label>
         <input type="number" id="etest_p" name="etest_p" step="0.01" required>

         <label for="mba_p">MBA Percentage:</label>
         <input type="number" id="mba_p" name="mba_p" step="0.01" required>

         <label for="ssc_b">SSC Board:</label>
         <select id="ssc_b" name="ssc_b">
            <option value="0">Central</option>
            <option value="1">Others</option>
         </select>

         <label for="hsc_b">HSC Board:</label>
         <select id="hsc_b" name="hsc_b">
            <option value="0">Central</option>
            <option value="1">Others</option>
         </select>

         <label for="hsc_s">HSC Stream:</label>
         <select id="hsc_s" name="hsc_s">
            <option value="0">Arts</option>
```

```
            <option value="1">Commerce</option>
            <option value="2">Science</option>
        </select>

        <label for="degree_t">Degree Type:</label>
        <select id="degree_t" name="degree_t">
            <option value="0">Comm&Mgmt</option>
            <option value="1">Others</option>
            <option value="2">Sci&Tech</option>
        </select>

        <label for="workex">Work Experience:</label>
        <select id="workex" name="workex">
            <option value="0">No</option>
            <option value="1">Yes</option>
        </select>

        <label for="specialisation">Specialisation:</label>
        <select id="specialisation" name="specialisation">
            <option value="0">Mkt&HR</option>
            <option value="1">Mkt&Fin</option>
        </select>

        <button class="btn" type="submit">Predict</button>
    </form>

    <div id="results" class="results">
        <h2>Results:</h2>
        <div id="rfResult">Random Forest: </div>
        <div id="svmResult">SVM: </div>
        <div id="logRegResult">Logistic Regression: </div>
    </div>
</div>

<script>
    document.getElementById('predictionForm').addEventListener('submit', async (event) => {
        event.preventDefault();

        const formData = new FormData(event.target);
        const data = Object.fromEntries(formData);

        const response = await fetch('/predict', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(data)
        });

        const result = await response.json();
        document.getElementById('rfResult').textContent = `Random Forest: ${result['Random
Forest']}`;
```

```
            document.getElementById('svmResult').textContent = `SVM: ${result['SVM']}`;
            document.getElementById('logRegResult').textContent = `Logistic Regression: $
{result['Logistic Regression']}`;
        });
    </script>
</body>
</html>
```

3.2. Explanation

- **Form Elements:** Input fields for user data including percentages, categorical choices, etc.
- **JavaScript:** Handles form submission, sends data to the `/predict` endpoint, and updates the results section with predictions.

4. CSS File (style.css)

4.1. Code Overview

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
}

.container {
    width: 90%;
    max-width: 800px;
    margin: 50px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    text-align: center;
    color: #333;
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-top: 10px;
```

```
    font-weight: bold;
}

input, select {
    margin-top: 5px;
    padding: 10px;
    font-size: 16px;
    border-radius: 5px;
    border: 1px solid #ccc;
    background-color: #fff;
}

input[type="number"] {
    height: 40px;
}

button {
    margin-top: 20px;
    padding: 10px;
    font-size: 18px;
    color: #fff;
    background-color: #007BFF;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

.results {
    margin-top: 30px;
}

.results div {
    margin-bottom: 10px;
    font-size: 18px;
    color: #333;
}
```

4.2. Explanation

- **General Styling:** Provides a clean, modern look with a light background and rounded container.
- **Form Styling:** Consistent styling for form elements with increased height for input fields and a button with hover effect.
- **Results Section:** Clear display for prediction results.

5. Deployment Configuration

5.1. `requirements.txt`

Flask==2.0.3
pandas==1.4.2
scikit-learn==1.1.1
joblib==1.2.0

5.2. `Procfile`

```
web: gunicorn app:app
```

5.3. `runtime.txt` (optional)

```
python-3.9.12
```

6. Testing and Validation

6.1. Unit Testing

- **Function Tests:** Test each function in `app.py` to ensure they handle input and output correctly.
- **Integration Tests:** Test end-to-end form submissions and predictions.

6.2. Browser Testing

- Test the web application in multiple browsers and devices to ensure responsiveness and compatibility.

6.3. Model Accuracy

- Evaluate the performance of machine learning models using test datasets and update models as needed.

7. Security Considerations

- **Input Validation:** Ensure that all user inputs are validated and sanitized to prevent security issues.
- **Secure Deployment:** Use HTTPS and secure configurations for deployment on Render.com.

8. Documentation

- **Code Comments:** Include comments in code to explain functionality and logic.
- **Developer Documentation:** Provide detailed explanations for complex code sections and configurations.
