High-Level Document: Placement Predictor Web Application

1. Project Overview

Title: Placement Predictor

Description: The Placement Predictor is a web application that leverages machine learning models to predict the placement status of candidates based on their educational and professional details. It provides predictions using Random Forest, SVM, and Logistic Regression models.

2. Objectives

- To provide a user-friendly interface for candidates to input their details.
- To use machine learning models to predict placement outcomes based on the input data.
- To deliver real-time predictions and results via a web application.

3. Features

- **User Input Form:** A form for candidates to input personal and academic details.
- **Model Predictions:** Predictions from three different models (Random Forest, SVM, and Logistic Regression).
- **Result Display:** Display the prediction results on the web page.
- **Responsive Design:** Modern and responsive design to ensure compatibility across devices.

4. Architecture

4.1. System Architecture

- **Frontend:** HTML, CSS, and JavaScript for user interface.
- **Backend:** Flask web framework to handle form submissions, run predictions, and serve the application.
- **Machine Learning Models:** Trained models for Random Forest, SVM, and Logistic Regression.
- **Deployment Platform:** Render.com for hosting and deployment.

4.2. Components

- **Flask Application (app.py):** Manages routing, handles form submissions, and integrates with ML models.
- **Templates (HTML Files):** User interface components and forms.
- **Static Files (CSS & JS):** Styling and client-side scripting.
- **Model Files:** Serialized machine learning models saved as `.pkl` files.

5. Deployment Strategy

5.1. Preparation

1. **Source Code Repository:** Store code in a Git repository (GitHub/GitLab/Bitbucket).
2. **Configuration Files:**
    - `requirements.txt`: Lists dependencies.
    - `Procfile`: Defines how to run the Flask application.

5.2. Deployment on Render.com

1. **Create a Render Account:** Sign up or log in to Render.com.
2. **Create New Web Service:**
    - Connect to the Git repository.
    - Configure the service with build and start commands.
    - Set environment variables if needed.
3. **Deploy:** Render automatically handles the build and deployment process.

5.3. Post-Deployment

- **Testing:** Validate the deployment by accessing the provided URL.
- **Monitoring:** Use Render's logs and monitoring tools to ensure smooth operation.
- **Scaling:** Adjust resources based on traffic and usage.

6. Security

- **Data Protection:** Ensure that user data is securely handled and not exposed.
- **Model Security:** Protect the ML model files and ensure they are not accessible to unauthorized users.

7. Future Enhancements

- **Model Improvement:** Continuously retrain and improve models based on new data.
- **Additional Features:** Implement additional features such as user authentication, detailed analytics, or integration with other platforms.

8. Dependencies

- **Flask:** Web framework for Python.
- **Scikit-learn:** Machine learning library.
- **Gunicorn:** WSGI HTTP server for UNIX.
- **Render.com:** Platform for deployment and hosting.

9. Development and Testing

- **Local Development:** Set up a local development environment with Flask and necessary dependencies.
- **Testing:** Conduct unit tests, integration tests, and user acceptance tests to ensure functionality.

10. Documentation

- **User Guide:** Instructions for using the web application.
- **Developer Guide:** Documentation for maintaining and extending the application.

11. References

- **Flask Documentation:** Flask Docs
- **Scikit-learn Documentation:** Scikit-learn Docs

- **Render.com Documentation:** Render Docs
