# Project Report: Placement Predictor Web Application

## 1. Introduction

### 1.1. Project Overview

The "Placement Predictor" is a web-based application designed to predict the likelihood of a student being placed in a job based on various academic and personal parameters. The application leverages machine learning models to make predictions and provides users with insights based on their input data.

### 1.2. Purpose

The purpose of this project is to develop a user-friendly web application that utilizes machine learning models to predict job placement probabilities. This tool aims to assist students in understanding their placement prospects based on their academic and personal data.

### 1.3. Target Audience

- **Students:** Individuals seeking insights into their job placement chances based on their academic performance.
- **Career Counselors:** Professionals providing guidance to students regarding their career prospects.

## 2. Objectives

- Develop a web application using Flask that integrates machine learning models.
- Provide a user-friendly interface for students to input their data and receive predictions.
- Ensure the application is accessible and responsive across different devices.
- Deploy the application on a cloud platform for public access.

## 3. Scope

### 3.1. Functional Requirements

- **Input Form:** Users must be able to input data related to their academic performance and personal information.
- **Prediction Models:** The application should utilize Random Forest, SVM, and Logistic Regression models to generate predictions.
- **Result Display:** Display the prediction results clearly and concisely for each model.
- **User Interaction:** The application must handle user inputs, perform predictions, and display results without errors.

### 3.2. Non-Functional Requirements

- **Performance:** The application should provide predictions within a reasonable time frame.

- **Security:** Ensure user data is handled securely and the application is protected against common web vulnerabilities.
- **Usability:** The interface should be intuitive and easy to navigate for users with varying levels of technical expertise.

# 4. Methodology

## 4.1. Data Collection and Preprocessing

- **Data Source:** Utilize historical data provided in `train.csv` for training the machine learning models.
- **Preprocessing:** Clean the data, handle missing values, and convert categorical features into numerical format using one-hot encoding.

## 4.2. Model Development

- **Model Selection:** Train and evaluate multiple machine learning models including Random Forest, SVM, and Logistic Regression.
- **Hyperparameter Tuning:** Optimize model performance using techniques such as Grid Search.

## 4.3. Web Application Development

- **Backend:** Develop the application using Flask to handle user requests and integrate the machine learning models.
- **Frontend:** Create an intuitive user interface using HTML, CSS, and JavaScript for form inputs and result display.

## 4.4. Deployment

- **Deployment Platform:** Deploy the web application on Render.com for accessibility over the internet.
- **Testing:** Conduct thorough testing to ensure the application functions correctly and meets all requirements.

# 5. Architecture

## 5.1. System Architecture

The architecture of the Placement Predictor web application consists of three main components:

1. **Frontend (User Interface):**
    - **Technologies:** HTML, CSS, JavaScript
    - **Function:** Collect user inputs and display prediction results.
2. **Backend (Server-side Processing):**
    - **Technologies:** Flask (Python web framework)
    - **Function:** Handle user requests, process data, invoke machine learning models, and return results.

3. **Machine Learning Models:**
    - **Technologies:** Scikit-learn (Python library for machine learning)
    - **Function:** Generate predictions based on user inputs using trained models.

### 5.2. Deployment Architecture

- **Render.com:**
    - **Service:** Hosting the Flask application.
    - **Function:** Manage web hosting, handle HTTP requests, and provide a URL for public access.

## 6. Wireframe

The wireframe for the Placement Predictor web application includes:

- **Header:** Displays the application title "Placement Predictor".
- **Form Section:** Contains input fields for user data.
- **Submit Button:** Triggers the prediction process.
- **Results Section:** Displays the prediction results from different models.
- **Footer:** Includes optional links for additional information.

*Refer to the visual wireframe for detailed layout.*

## 7. Implementation Details

### 7.1. Frontend Development

- **HTML:** Structure of the form and results display.
- **CSS:** Styling to enhance the appearance and ensure responsiveness.
- **JavaScript:** Handle form submission and interact with the Flask backend.

### 7.2. Backend Development

- **Flask:** Create routes to handle form submissions, invoke prediction models, and return results.
- **API Integration:** Endpoint `/predict` to process user data and return prediction results.

### 7.3. Machine Learning Models

- **Model Training:** Train models using historical data and evaluate their performance.
- **Model Integration:** Load and use the trained models within the Flask application.

# 8. Deployment

## 8.1. Deployment Steps

1. **Prepare Codebase:** Ensure the Flask application and model files are ready for deployment.
2. **Create Render Account:** Sign up and set up a new web service on Render.com.
3. **Deploy Application:** Upload code to Render.com and configure environment settings.
4. **Verify Deployment:** Test the application to ensure it functions correctly and is accessible online.

## 8.2. Maintenance

- **Monitoring:** Regularly monitor application performance and address any issues.
- **Updates:** Apply updates to the application as needed, including model retraining and feature enhancements.

# 9. Conclusion

The Placement Predictor web application provides a valuable tool for students to assess their job placement prospects based on their academic performance. By integrating machine learning models with a user-friendly web interface, the application offers actionable insights and supports users in their career planning efforts.

# 10. Future Enhancements

- **Model Improvements:** Continuously improve model accuracy with updated data and advanced techniques.
- **Feature Additions:** Add new features based on user feedback and requirements.
- **User Experience:** Enhance the user interface and experience based on usability testing.