Campus Placement Prediction

Revision Number: 1.0
Last date of revision: 30/07/2023

Document Version Control

Date Issued	Version	Description	Author
30/07/2023	1	Initial HLD — V1.0	Dev Reshamiya

Contents

Abstract

The Placement of students is one of the most important objective of an educational institution. Reputation and yearly admissions of an institution invariably depend on the placements it provides it students with. That is why all the institutions, arduously, strive to strengthen their placement department so as to improve their institution on a whole. Any assistance in this particular area will have a positive impact on an institution's ability to place its students. This will always be helpful to both the students, as well as the institution.

The main goal is to predict whether the student will be recruited in campus placements or not based on the available factors in the dataset.

By analyzing historical placement data, this project identifies key factors that contribute to successful placements and those that hinder the process. Machine learning algorithms are employed to predict the likelihood of a student being placed based on various attributes such as academic performance, extracurricular activities, and skillset. These predictive models assist placement officers in identifying students who may require additional support and personalized guidance.

1 Introduction

1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

- Present all of the design aspects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project
- List and describe the non-functional attributes like:
    - Security
    - Reliability
    - Maintainability
    - Portability
    - Reusability
    - Application compatibility
    - Resource utilization
    - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3 Definitions

Term	*Description*
UGV	Unmanned Ground Vehicle
Database	Collection of all the information monitored by this system
IDE	Integrated Development Environment
AWS	Amazon Web Services

2 General Description

2.1 Product Perspective

Our product is a cutting-edge AI-powered campus placement platform that revolutionizes the traditional placement process, providing an efficient and data-driven solution for students and recruiters to connect and achieve successful outcomes..

2.2 Problem statement

1 Develop a predictive model to determine whether a student will be placed in a campus placement program based on their academic and other relevant information.
2 Build a salary prediction model to estimate the expected salary for students who have been successfully placed.
3 Create a user-friendly web interface that allows students and recruiters to access the predictive models easily and obtain placement-related insights.

2.3 PROPOSED SOLUTION

The proposed solution aims to develop a comprehensive campus placement prediction system that utilizes machine learning algorithms to forecast a student's likelihood of getting placed and estimate their expected salary upon successful placement. The system will be designed with Python's Flask framework for the backend and ReactJS for the frontend, ensuring an efficient and interactive user interface. Preprocessing steps like outlier removal, feature engineering, and standard scaling will be employed to enhance model performance. The trained logistic regression model will predict placement status, while the linear regression model will forecast salary. The web application will offer a simple form-based user interface, allowing students to input their academic and personal details. Upon submission, the models will process the data and display the results, providing valuable insights for both students and recruiters.

2.4 FURTHER IMPROVEMENTS

1. **Expand Data Collection:** Gather more comprehensive and up-to-date data to enhance the model's accuracy. Include additional relevant features such as extracurricular activities, internships, and industry-specific certifications.
2. **Hyperparameter Tuning:** Fine-tune the hyperparameters of the chosen models to optimize their performance and increase prediction accuracy.
3. **Feature Importance Analysis:** Conduct feature importance analysis to identify the most influential factors affecting placement and salary predictions. This insight can help students focus on key areas for improvement.
4. **Implement Cross-Validation:** Utilize cross-validation techniques like K-Fold Cross-Validation to evaluate the model's performance on different subsets of the data and mitigate overfitting issues.
5. **Real-Time Updates:** Introduce a feature to update the model periodically with fresh data to ensure its relevance and accuracy over time.

2.5 Technical Requirements

This document outlines the technical requirements for a campus placement system. The goal is to create a platform that predicts whether a student will be placed in a company and also predicts their expected salary. The system will require the following technical features:

1. Data Preprocessing: The system should be able to preprocess the input data, including handling missing values, encoding categorical variables, and scaling numerical features.
2. Machine Learning Models: The system will include machine learning models, such as logistic regression for placement prediction and linear regression for salary prediction. The models should be trained on historical placement data.
3. Web Interface: The system will have a user-friendly web interface to collect input from students, such as their academic details and work experience.
4. Model Integration: The system will integrate the trained machine learning models to make real-time predictions based on user inputs.
5. Data Visualization: The system will provide visualizations of the prediction results, such as pie charts to show the probability of placement and salary range.
6. Backend Server: The system will require a backend server to handle user requests, process data, and serve prediction results.
7. Database: The system will store historical placement data and user inputs in a database for future reference and analysis.
8. Security: The system should implement security measures to protect user data and ensure privacy.
9. Scalability: The system should be designed to handle a large number of users and data to support campus-wide usage.
10. Deployment: The system should be deployable on a web server or cloud platform to make it accessible to students and placement coordinators.

2.6 Data Requirements

The data requirements for the campus placement system are as follows:

1. Student Data: The system will need data about individual students, including their academic performance, such as SSC and HSC percentages, degree percentage, and MBA percentage. Other relevant information, such as gender, work experience, and specialization, will also be required.
2. Placement Data: Historical placement data from previous years will be necessary to train the machine learning models. This data will include information about which students were placed in companies, along with their respective salaries.
3. Categorical Data: Data related to categorical variables, such as board of education (SSC and HSC), degree specialization, and work experience (yes/no), will be needed for encoding and processing.
4. Numerical Data: Numerical data, such as academic percentages, work experience duration, and expected salary, will be required for training the models and making predictions.
5. Clean and Preprocessed Data: The data should be preprocessed and cleaned to handle missing values, outliers, and ensure consistency for accurate model training and predictions.
6. Real-Time Input Data: The system will require real-time data from users when they fill out the web interface with their academic and personal details for placement prediction.
7. Placement Offers: Data about job offers and salary packages from various companies will be helpful to analyze trends and provide better salary predictions.

2.7 Tools used

The tools used in the campus placement project are as follows:

1. Python: Python is the primary programming language used for data analysis, data preprocessing, and building machine learning models.
2. Flask: Flask is a lightweight and flexible web framework used for developing the backend of the web application.
3. ReactJS: ReactJS is a JavaScript library used for building the frontend of the web application, providing a user-friendly interface for users to input their details.
4. Chart.js: Chart.js is a JavaScript library used for creating interactive and visually appealing charts and graphs to visualize the model predictions and accuracy metrics.
5. Pandas: Pandas is a Python library used for data manipulation, handling data in the form of DataFrames, and preprocessing the dataset.
6. NumPy: NumPy is a Python library used for numerical computations and mathematical operations on arrays.
7. Scikit-learn: Scikit-learn is a popular Python library used for machine learning tasks, including building classification and regression models.
8. Joblib: Joblib is used for saving and loading machine learning models.
9. Matplotlib: Matplotlib is a Python library used for creating various types of plots and visualizations.

These tools are essential for the development and implementation of the campus placement project, enabling data analysis, model training, web development, and providing valuable insights to students and placement coordinators.

2.8 Constraints

1. Data Availability: The success of the project relies on the availability of high-quality and relevant data related to student profiles, academic performance, and placement outcomes. Limited or inaccurate data could impact the model's accuracy and effectiveness.
2. Model Accuracy: The machine learning models used for predicting placement status and salary are based on historical data and are subject to certain limitations. The accuracy of the models might not be 100%, and there could be cases where predictions may not align with actual outcomes.
3. Ethical Considerations: The project involves handling sensitive information about students, such as academic performance and personal details. Ensuring data privacy and complying with ethical guidelines are critical aspects that need to be strictly followed.
4. Deployment and Hosting: Deploying the web application requires hosting the backend server and the frontend on suitable platforms. This could incur costs, and ensuring smooth hosting and maintenance are important considerations.
5. User Engagement: Encouraging students to use the platform and provide accurate information is essential for the success of the project. Ensuring a user-friendly interface and clear communication of the benefits are crucial to drive user engagement.
6. Scaling and Performance: As the number of users and data increases, the web application should be capable of handling the load efficiently. Ensuring scalability and optimizing the performance of the application are important factors to consider.

2.9 Assumptions

3 **Data Completeness:** It is assumed that the provided data for students' profiles, academic performance, and placement outcomes are complete and do not contain significant missing values. Any missing data is assumed to be minimal and can be imputed or handled appropriately during data preprocessing.

4 **Data Reliability:** The assumption is that the data collected and used for training the machine learning models are accurate and reliable. Any inconsistencies or errors in the data are assumed to be minimal and do not significantly affect the model's performance.

5 **Generalization of Models:** The machine learning models used for predicting placement status and salary are assumed to generalize well to unseen data. It is assumed that the models will perform effectively on new student data, not just on the training data used during model development.

6 **Equal Opportunity:** The project assumes that the machine learning models used for prediction do not introduce any bias or discrimination against any specific group of students. Equal opportunity for all students is considered during model development.

7 **Stable Placement Criteria:** The project assumes that the placement criteria and processes remain relatively stable during the implementation of the solution. Any major changes in the placement criteria during the project timeline could affect the model's accuracy and relevance.

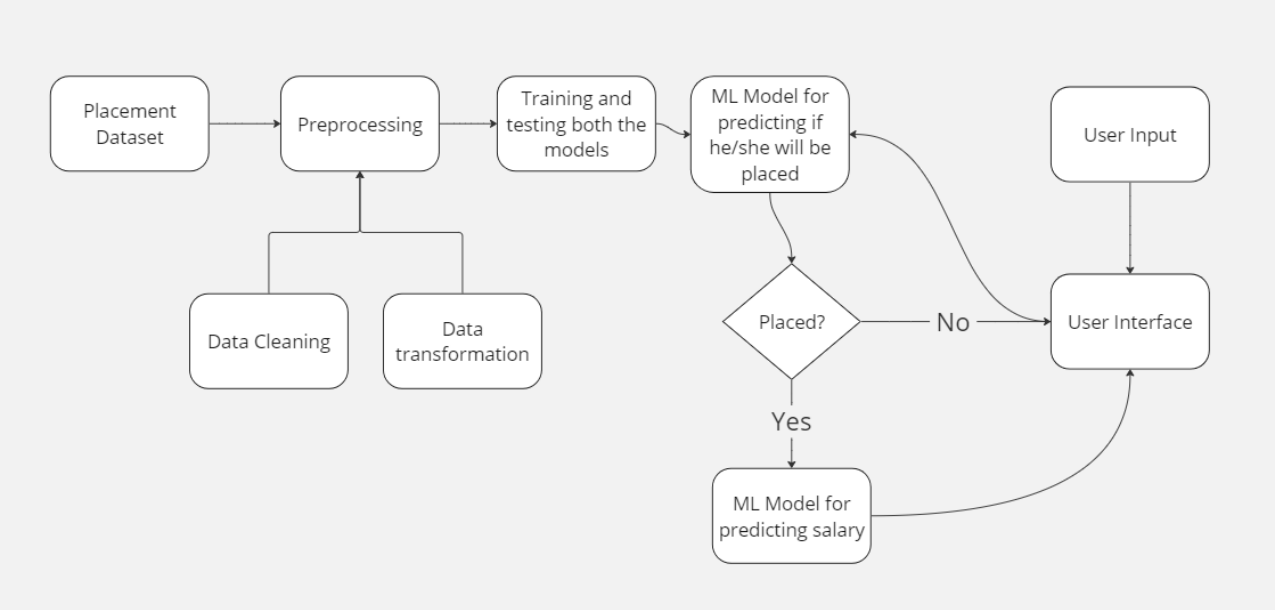
8 **Active User Participation:** It is assumed that students will actively participate in using the web application and provide accurate information during form filling. User engagement and cooperation are crucial for the success of the project.

3 Design Details

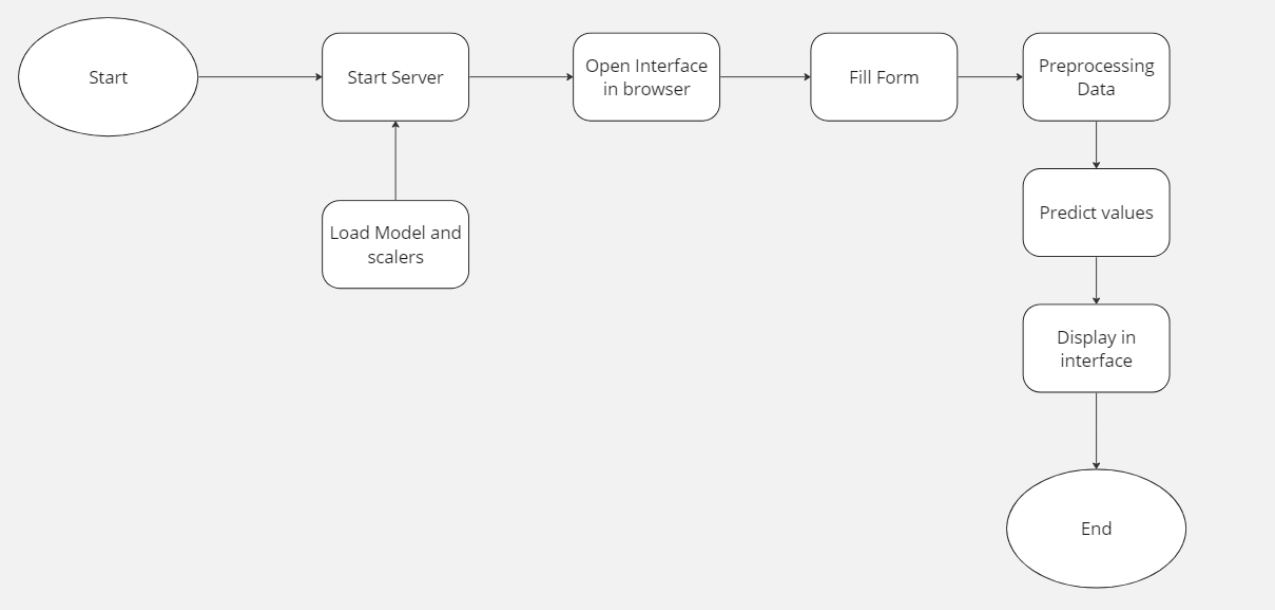
3.1 Process Flow

For identifying the different types of anomalies, we will use a deep learning base model. Below is the process flow diagram is as shown below.

3.1.1 Proposed methodology



3.1.2 Deployment Process



3.2 Event log

The system should log every event so that the user will know what process is running internally.

Initial Step-By-Step Description:

1. User Interaction: User opens the web application in their browser and accesses the landing page.
2. Form Submission: User fills in their personal and academic details in the provided form fields (e.g., gender, 10th and 12th percentage, specialization, etc.).
3. Prediction Request: User clicks the "Submit" button to request predictions for their placement status and expected salary.
4. Data Preprocessing: The app processes the user input data, handling missing values, and standardizing the format for model compatibility.
5. Prediction: The app feeds the preprocessed data into the logistic regression model to predict the placement status (placed or not placed) and the linear regression model to predict the expected salary.
6. Model Response: The predicted results are sent back from the server to the web application.
7. Result Display: The web application displays the prediction results on the user interface

3.3 Error Handling

1. Form Validation: The app checks for mandatory fields and valid input formats (e.g., numeric values within a certain range, valid email address). If any input is missing or incorrect, appropriate error messages are displayed to guide the user.
2. • Server Errors: If there is an issue with the server, such as unavailability or internal server errors, the app displays a user-friendly error message indicating the problem and advises the user to try again later.
3. • Model Compatibility: The app ensures that the user's input data is in the correct format and compatible with the prediction models. If the data format is not valid for predictions, the app will notify the user and request the necessary corrections.
4. • Data Preprocessing Errors: During data preprocessing, if there are any issues with handling missing data or scaling, the app provides informative error messages to assist the user in understanding and resolving the problem.
5. • Unhandled Exceptions: The app includes robust exception handling to catch any unexpected errors or exceptions that may occur during its execution. These errors are logged in the server's event log for later analysis and debugging.

3.4 Performance

Prediction Speed: The app aims to provide quick predictions for placement status and expected salary based on user input. The prediction models are optimized for fast execution, ensuring minimal latency in displaying results.

3.5 Reusability

The code written and the components used should have the ability to be reused with no problems.

3.6 Application Compatibility

The different components for this project will be using Python as an interface between them. Each component will have its own task to perform, and it is the job of the Python to ensure proper transfer of information.

3.7 Resource Utilization

When any task is performed, it will likely use all the processing power available until that function is finished.

3.8 KPIs (Key Performance Indicators)

Key Performance Indicators (KPIs) for the Campus Placement Project App:

1. Prediction Accuracy: The accuracy of the placement status and salary predictions made by the app's machine learning models is a crucial KPI. It measures how well the app's models are performing in correctly predicting whether a candidate will be placed and estimating their expected salary.
2. Response Time: The time taken by the app to respond to user actions, such as form submission and displaying prediction results, is an essential KPI. Lower response times indicate better app performance and a more enjoyable user experience.
3. User Engagement: User engagement metrics, such as the number of form submissions and user interactions with the app, help measure how actively users are using the app and how interested they are in the prediction results.
4. Error Rate: The percentage of errors encountered during the app's operation is an important KPI. Lower error rates indicate better app stability and reliability.
5. Scalability: The app's ability to handle an increasing number of users and requests without a significant degradation in performance is a crucial KPI, especially during peak periods.

4 Conclusion

In conclusion, the Campus Placement Project App aims to provide valuable insights and predictions for students seeking placements. By utilizing machine learning models for predicting placement status and estimated salaries, the app offers personalized guidance to users. The app's user-friendly interface ensures ease of access and smooth navigation. Through the use of technologies such as ReactJS for the frontend and Flask for the backend, the app achieves seamless communication and data processing. While the app has shown promising results in predicting placement outcomes, there are opportunities for further improvements, such as incorporating more data sources and advanced machine learning algorithms. Overall, the app demonstrates the potential to assist students in making informed career decisions and enhancing their chances of securing successful placements. With continued monitoring of performance metrics and user feedback, the Campus Placement Project App can continue to evolve and become a valuable resource for students and placement stakeholders