Cloud Counselage

GIFT A CAREER FOUNDATION

INTERNSHIP PROGRAM 2023

SOFTWARE DESIGN SPECIFICATION

Machine Learning

EduPredict: Graduation & Placement Prediction

Created By:	Atharv Musale	**Approved By:**	Sarang Kadakia
Created On:	31-07-2024	**Approved On:**	25-07-2024

INDEX


Cloud Counselage

1 PURPOSE

This document is created based on the requirement specification document. The purpose of this Software Design Specification (SDS) Document is to break down the project into components to describe in detail what the purpose of each component is and how it will be implemented. The SDS will also serve as a tool for verification and validation of the final product.

2 PROJECT SCOPE

The scope of the project includes building a predictive model using historical student data to forecast graduation year and placement status. The system will help academic institutions assess the likelihood of student placement based on various factors such as academic performance, skillset, and extracurricular activities. This predictive model uses tools such as Python, pandas, and scikit-learn for data processing and model training.

3 SYSTEM OVERVIEW

The system consists of multiple components, including data collection, data preprocessing, feature extraction, model training, and model evaluation. The main subsystems include:

- **Data Collection Subsystem**: Retrieves historical data of students.
- **Preprocessing Subsystem**: Cleans and formats the data.
- **Machine Learning Model Subsystem**: Trains and tests various models.
- **Evaluation Subsystem**: Assesses the model's performance.

4 DESIGN CONSIDERATIONS

This section describes requirements, assumptions and dependencies to be addressed to devise a complete design solution.

4.1 Requirements

The functional requirements for this project involve implementing various product features and functions that allow users to achieve the desired task of predicting students' graduation year and placement status. The system must be equipped to handle data inputs, process that data through machine learning algorithms, and generate accurate predictions based on predefined criteria. The development team will use Jupyter Notebook for interactive code development, as well as Python libraries such as pandas, NumPy, and scikit-learn to facilitate data analysis and machine learning. Additionally, Matplotlib will be utilized for data visualization purposes, ensuring that trends and patterns in the data are clearly represented. Microsoft Excel will be employed for data storage and manipulation, enabling easy access to large volumes of data and its subsequent transformation into formats suitable for machine learning models.

4.2 Assumptions

Several assumptions have been made for the successful execution of this project. First, it is assumed that the student data provided is accurate, complete, and representative of the target population, which is crucial for training robust and reliable models. It is also assumed that the machine learning techniques and models selected for the project, such as decision trees, and random forests, will be appropriate for the datasets and objectives outlined. These assumptions help frame the project's approach and guide the selection of tools, techniques, and methodologies throughout the development process.

4.3 Dependencies

- Python libraries such as pandas, scikit-learn, and matplotlib.
- Access to the historical student data.

5 SYSTEM ARCHITECTURE

The software system architecture refers to the logical organization of a distributed system into software components. It defines how components of a software system are assembled, their relationship and communication between them. It serves as a blueprint for software application and development basis for developer team. An effective architecture serves as the conceptual glue that holds every phase of the project together for all of its stakeholders, enabling agility, time and cost savings, and early identification of design risks.

The Software architecture:

- Defines structure of a system
- Defines behaviour of a system
- Defines component relationship
- Defines communication structure
- Balances stakeholder’s needs
- Influences team structure
- Focuses on significant elements
- Captures early design decisions

Below some important characteristics which are commonly considered are explained.

Operational Architecture Characteristics:

- Availability
- Performance
- Reliability
- Low fault tolerance
- Scalability

Structural Architecture Characteristics:

- Configurability
- Extensibility
- Supportability
- Portability
- Maintainability

Cross-Cutting Architecture Characteristics:

- Accessibility
- Security
- Usability
- Privacy
- Feasibility

5.1 Architectural Strategies

The architectural strategy for this project revolves around the use of a modular, layered approach to ensure scalability, maintainability, and flexibility. The project is built using the following key components:

- Data Layer: This includes all data input sources such as Microsoft Excel spreadsheets containing student records and their relevant attributes. Data preprocessing is performed using pandas and NumPy, allowing for the cleaning, transformation, and preparation of the data for analysis.
- Machine Learning Layer: At this layer, different machine learning models such as logistic regression, decision trees, and random forests will be applied using scikit-learn. This layer will handle the core predictive tasks, learning patterns from the data and making predictions regarding graduation year and placement outcomes.
- Visualization Layer: This layer will leverage matplotlib to visualize key insights, such as correlations between features and the likelihood of successful placement. The interactive nature of matplotlib allows users to explore the data dynamically and uncover trends.
- User Interface: The user interface will be developed in Jupyter Notebook, providing an interactive environment where the development and analysis are presented in a clear and accessible format for stakeholders and developers alike.

5.2 Structure & Relationships

The project will follow a modular structure, where each component is treated as an independent module that interacts with the others in a sequential manner.

- Data Source: The Excel datasets act as the foundation, feeding into the data preprocessing module.

- Data Preprocessing: This module cleans, transforms, and structures the data. Relationships between variables are explored using statistical analysis, and features are selected or engineered to improve model performance.
- Model Layer: Various machine learning models are trained on the preprocessed data. These models have a direct relationship with the data layer, as they depend on the input features provided by that module.
- Visualization Layer: The results generated by the machine learning models are passed to the visualization module, where matplotlib is used to create interactive plots and graphs that depict the relationships between student features and their predicted graduation/placement outcomes.
- Feedback Loop: Based on the results of the model and visualizations, refinements in the data preprocessing or model selection may be made to improve accuracy and ensure reliable predictions.

6 DETAILED DESCRIPTION OF COMPONENTS

For detailed description of the components, please refer **Appendix A – Detailed Description of Components**

The below template will be used to specify the details of all the components

Table 1: Detailed Design Specification Template

Identification	EduPredict: Graduation & Placement Prediction
Type	Machine Learning-based Predictive System
Purpose	The primary purpose of EduPredict is to analyze historical student data and predict two key outcomes: their graduation year and placement status in their respective fields. The system provides valuable insights that can help educational institutions and students make informed decisions.
Subordinates	The system will have subordinate components such as the Data Preprocessing Module, Model Training Module, and Visualization Module, each handling specific parts of the overall process. The coordination between these modules ensures a seamless flow from data ingestion to prediction and result visualization.
Dependencies	Software Dependencies: Jupyter Notebook, Python, pandas, NumPy, scikit-learn, Matplotlib, and Microsoft Excel. External Data: The accuracy of predictions depends on the availability and quality of student data provided by the institution. Human Resources: Data engineers for data preprocessing, machine learning engineers for model training and optimization, and UI/UX experts for the visualization interface.
Interfaces	Data Input Interface: Utilizes Microsoft Excel for uploading student records. Model Interface: The machine learning models interface with the preprocessed data to perform predictive tasks. Visualization Interface: matplotlib is used for generating visual insights and presenting data trends to the user in an interactive manner.
Resources	Hardware: A local machine or cloud infrastructure with enough processing power to handle machine learning tasks

	Software: Python programming environment with necessary libraries
Processing	Data Preprocessing: Cleaning and transforming student data from Excel sheets using pandas and NumPy. Model Training: Training machine learning models like decision trees, logistic regression, and random forests using scikit-learn to predict graduation year and placement status. Prediction: Based on student features, the system predicts the likely graduation year and placement status. Visualization**:** Results are visualized using interactive graphs to provide insights into the model's predictions.
Data	Input Data: Student records stored in Excel files. Processed Data: Cleaned and feature-engineered data fed into machine learning models. Prediction Data: Outputs of machine learning models predicting the likelihood of student graduation and placement. Visualization Data: Visual outputs for trend analysis and insights derived from prediction results.

7 INTEGRATIONS

EduPredict project is built around interaction through Jupyter Notebook. Users input student data through CSV files or Excel spreadsheets, which are then processed using Python libraries like Pandas. The system guides users through model training and evaluation steps, displaying model performance metrics and predictions directly in the notebook. Data visualizations, generated with tools like Matplotlib and Plotly, provide insights into the predictions and model accuracy. The notebook environment also includes error handling to manage incorrect or incomplete data entries, ensuring users can easily interact with the system and receive predictions for graduation year and placement status.

8 APPENDICES

8.1 Appendix A – Detailed Description of Components

Identification	Data Preprocessing
Type	Module
Purpose	To clean and format data for the model.
Subordinates	Missing value imputation, feature scaling, etc.
Dependencies	Python libraries
Interfaces	Data preprocessing interface that allows for transformations and cleaning.
Resources	pandas, numpy
Processing	Code is executed to transform raw data into a usable form, dealing with missing values, etc.
Data	Student demographic, academic, and placement data for input.

Identification	Model Training
Type	Module
Purpose	To train and fit models with the student data.
Subordinates	Decision Tree, Random Forest
Dependencies	scikit-learn
Interfaces	Interface for training the machine learning models.
Resources	scikit-learn
Processing	The machine learning models are trained using a train-test split of the data. Hyperparameters are tuned as required.
Data	Data transformed from preprocessing is used to create and fit the models.

Identification	Model Evaluation
Type	Module
Purpose	To evaluate the accuracy and performance of the trained models.
Subordinates	Model evaluation metrics
Dependencies	Matplotlib, scikit-learn
Interfaces	Interface to display model evaluation metrics such as accuracy, precision, recall, and F1-score.
Resources	Matplotlib, numpy
Processing	Performance of models is measured through various evaluation metrics. Accuracy thresholds are determined.
Data	Results from model predictions and test data used for performance evaluation.