Low Level Design

Campus Placement Prediction System

|  |  |
| --- | --- |
| Written By | Kunal Dagar |
| Document Version | 0.1 |
| Last Revised Date | 17 – Oct -2023 |

**Document Control**

### Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 17 – Oct -  2023 | Kunal Dagar | Introduction , Architecture defined, Unit test cases were added |
| 0.2 |  |  |  |
| 0.3 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Approval Status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

1. [**Introduction**](#_heading=h.gjdgxs) 1
   1. [**What is Low-Level design document?**](#_heading=h.30j0zll) 1
   2. [**Scope**](#_heading=h.1fob9te) 1
2. [**Architecture**](#_heading=h.3znysh7) 2
3. [**Architecture Description**](#_heading=h.2et92p0) 3
   1. [**Data Description**](#_heading=h.tyjcwt) 3
   2. [**Data Transformation**](#_heading=h.3dy6vkm) 3
   3. [**Pipelines**](#_heading=h.1t3h5sf) 3
   4. [**Model Selection and Hyper paramter tuning**](#_heading=h.4d34og8) 3
4. [**Model Building**](#_heading=h.26in1rg) 4

**4.1.** **[Data from User](#_heading=h.lnxbz9)** 4

**4.2.** [**Data Validation**](#_heading=h.35nkun2) 4

**4.3.** [**Data Transformation**](#_heading=h.1ksv4uv) 4

**4.4.**  [**Model Call**](#_heading=h.44sinio) 4

**4.5.** [**Status Prediction**](#_heading=h.2jxsxqh) 4

**4.6.** [**Deployment**](#_heading=h.z337ya) 4

1. [**Unit Test Cases**](#_heading=h.1y810tw) 5

# Introduction

## What is Low-Level design document?

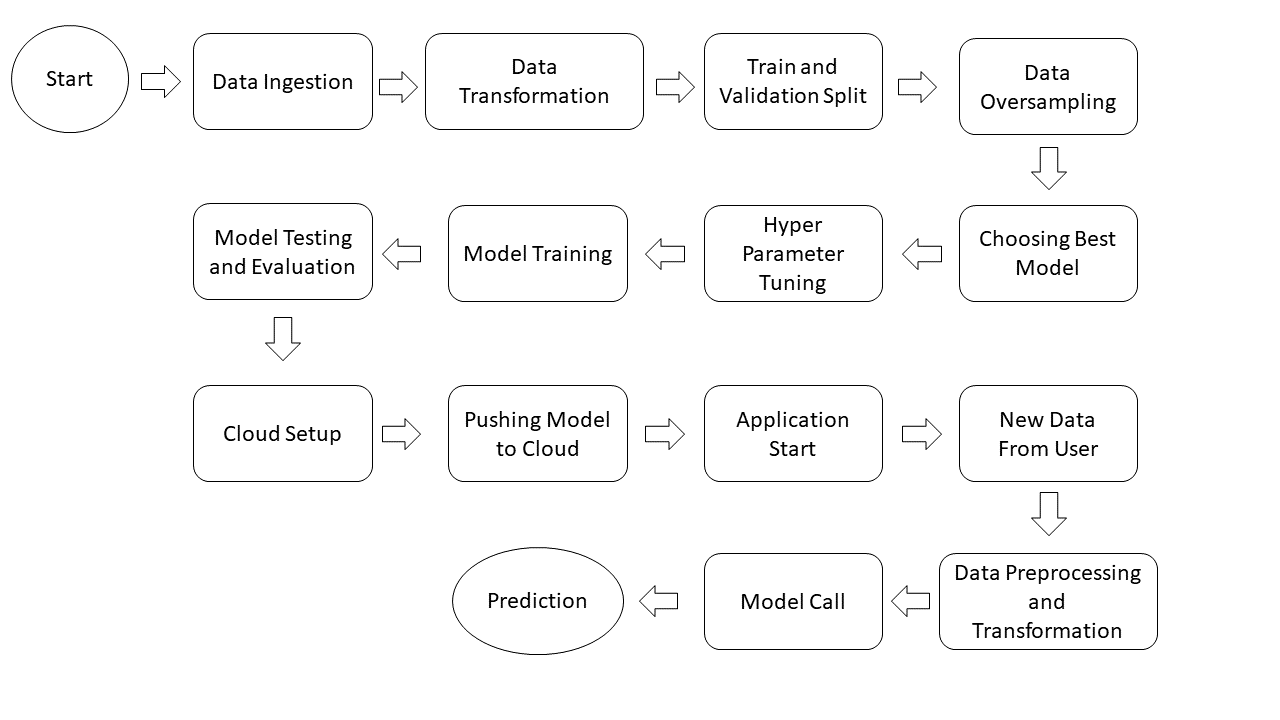
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

## Scope

Low-level design (LLD) is a component-level design process that follows a step-by-

step [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work

# Architecture



# Architecture Description

## Data Description

The Campus Recruitment dataset of Kaggle contains 215 records with 15 classes, the target class is status, we goal is to predict that based on certain parameters a student will be going to placed or not.

## Data Preprocessing

Data Pre-processing steps we could use are Null value handling, stop words removal, punctuation removal, Imbalanced data set handling, Handling columns with standard deviation zero or below a threshold, etc.

.

## Pipelines

With Pipelines we try to automate the process of data preprocessing such handling missing values or encoding the categorical features all we automated so that preprocessing can become more easy.

## Selecting Best Model and Hyper-parameter tuning

From a given list of models we try to find the best suited model for our data SVM in this case and try to do hyperparameter tuning in order to improve its accuracy and performance.

## Model Building

After selecting the best model we will be doing hyper parameter tuning in order to increase it’s accuracy using GridSearchCV that will provide us with the best parameters that we can use in our model to boost it’s efficiency.

## Data from User

Here we will collect data from the user like ssc board, hsc board, percentage, test percentages, MBA marks, specialization, work experience etc. through the web app.

## Data Validation

Here Data Validation will be done, given by the user.

## Data Transformation

Given data will be transformed as per the desired format.

## Model Call

Model will be applied on after transforming the data and results will be prediction.

## Status prediction

After applying the model the results will be predicted as placed or not placed.

## Deployment

We will be deploying the model to Render.

# Unit Test Cases

|  |  |  |
| --- | --- | --- |
| **Test Case Description** | **Pre-Requisite** | **Expected Result** |
| Verify whether the Application URL is  accessible to the user | 1. Application URL  should be defined | Application URL should be  accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | 1. Application URL is accessible 2. Application is deployed | The Application should load completely for the user when the URL is accessed |
| Verify whether the User is able to sign  up in the application | 1. Application is  accessible | The User should be able to sign up  in the application |
| Verify whether user is able to successfully login to the application | 1. Application is accessible 2. User is signed up to the application | User should be able to successfully login to the application |
| Verify whether user is able to see input fields on logging in | 1. Application is accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be able to see input fields on logging in |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should get Submit button to submit the inputs |
| Verify whether user is presented with recommended results on clicking  submit | 1. Application is accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be presented with recommended results on clicking  submit |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | The recommended results should be in accordance to the selections user made |
| Verify whether user has options to filter the recommended results as well | 1. Application is accessible 2. User is signed up | User should have options to filter the recommended results as well |