Low Level Design

**Insurance Prediction**

|  |  |
| --- | --- |
| Written By | Tejas Bhosale |
| Document Version | 0.1 |
| Last Revised Date | 23 – Aug -2024 |

**Document Control**

### **Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 19 – May -  2021 | Ankur Das | Introduction & Architecture defined |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

1. **Introduction**

**1.1 What is Low-Level design document?**

**1.2 Scope**

1. **Architecture**
2. **Architecture Description**

**3.1 Data Description**

**3.2 Data Cleaning**

**3.3 Data Transformation**

**3.4 Data Insertion into Database**

**3.5 Export Data from Database**

**3.6 Data Pre-processing**

**3.7 Data Clustering**

**3.8 Model Building**

**3.9 Data from User**

**3.10 Data Validation**

**3.11 User Data Inserting into Database**

**3.12 Data Clustering**

**3.13 Model Call for Specific Cluster**

**3.14 Recipe Recommendation & Saving Output in Database**

**3.15 Deployment**

1. **Unit Test Cases**

# 1. Introduction

## 1.1 What is Low-Level design document?

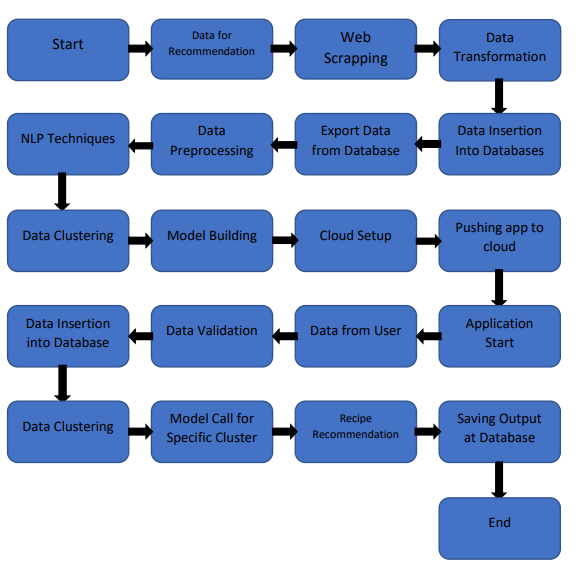
The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Insurance Prediction. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly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

1.2 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.

# 2. Architecture



# 3. Architecture Description

## 3.1 Data Description

The goal is to give people an estimate of how much they need based on  
their individual health situation. After that, customers can work with any health  
insurance carrier and its plans and perks while keeping the projected cost from our  
study in mind. This can assist a person in concentrating on the health side of an  
insurance policy rather than the ineffective part.

## 3.2. Data Cleaning

In this, we cleaned our data as per requirements. In this, we handled null values with mean and median also removes duplicates.

## 3.3 Data Transformation

In the Transformation Process, we will convert our original dataset into our labelled encoding. And will merge it with the Scrapped dataset.

## 3.4 Data Insertion into Database

* Database Creation and connection - Create a database with name passed. If the database is already created, open the connection to the database.
* Table creation in the database.
* Insertion of files in the table.

## 3.5 Export Data from Database

## Data Export from Database - The data in a stored database is exported as a CSV file to be used for Data Pre-processing and Model Training.

3.6 Data Pre-processing

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

## 3.7 Data Clustering

Ensembles techniques is used to create clusters in the pre-processed data. Whichever Algorithm perform well that will be used for model building.

## 3.8 Model Building

## After clusters are created, we will find the best model for each cluster. For each cluster, algorithms will be passed with the best parameters derived from Grid-Search. We will calculate the AUC scores for models and select the model with the best score. Similarly, the models will be selected for each cluster. All the models for every cluster will be saved for use in Recommendation.

## 3.9 Data from User

Here we will collect data from user such as age, bmi, children, smoking, gender, peer pressure.

## 3.10 Data Validation

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

3.11 User Data Inserting into Database

Collecting the data from the user and storing it into the database. The database can be either MYSQL or MongoDB.

3.12 Data Clustering

The model created during training will be loaded, and clusters for the user data will be predicted.

3.13 Model Call for Specific Cluster

Based on the cluster number, the respective model will be loaded and will be used to predict/Recommend the data for that cluster.

## 3.14 Recipe Recommendation & Saving Output in Database

After calling model Recipe/Output will be recommended, this output will be saved in Database and it will be used to show the same Output if other users provide the same data.

## 3.15 Deployment

We will be deploying the model to AWS. This is a workflow diagram for the Recipe Recommendation.

# 4. 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  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. |