Low Level Design

Mushroom Classification System

|  |  |
| --- | --- |
| Written By | Vishal Kumar Mahato, Sumit Kumar |
| Document Version | 0.2 |
| Last Revised Date | 06-April-2024 |

**Document Control**

**Change Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 06-April-2024 | Vishal Kumar Mahato | Introduction |
| 0.2 | 07-April-2024 | Sumit Kumar | Architecture, Architecture Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
|  |  |  |  |

**Approval Status:**

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

Contents

[**1.** **Introduction** 1](#_Toc15828)

[**1.1.** **What is Low-Level design document?** 1](#_Toc15829)

[**1.2.** **Scope** 1](#_Toc15830)

[**2.** **Architecture** 2](#_Toc15831)

[**3.** **Architecture Description** 3](#_Toc15832)

[**3.1.** **Data Description** 3](#_Toc15833)

[**3.3.** 3](#_Toc15835)

[**3.4.** 3](#_Toc15836)

[**3.5.** 3](#_Toc15837)

[**3.6.** **Data Pre-processing** 3](#_Toc15838)

[**3.7.** 3](#_Toc15839)

[**3.10. Model Building** 4](#_Toc15840)

[**3.11. Data from User** 4](#_Toc15841)

[**3.12. Data Validation** 4](#_Toc15842)

[**3.13.**  4](#_Toc15843)

[**3.14.** 4](#_Toc15844)

[**3.15.** 4](#_Toc15845)

[**3.16.** 4](#_Toc15846)

[**3.17. Deployment** 4](#_Toc15847)

[**4.** **Unit Test Cases** 5](#_Toc15848)

# 1. Introduction

## 1.1.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 Mushroom Classification System. LLD describes the class diagrams with the methods and relations between classes and program specification. 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

Data Preprocessing

Data Transformation

Data for Recommendation

Start

Application Start

Pushing app to cloud

Cloud Setup

Model Building

# 

End

Data Validation

Data From User

# 3. Architecture Description

## 3.1.Data Description

The mushroom dataset contains information about various mushroom species, particularly focusing on features that help determine whether a mushroom is edible or poisonous. The dataset is organized into a single CSV (Comma-Separated Values) file. I also provide the feature names and their corresponding possible values as described in the provided mapping JSON file.

## 3.2.Data Pre-processing

Perform data cleaning and preprocessing tasks such as handling missing values, encoding categorical variables, and scaling numerical features if necessary.

## 3.3. Model Building

We will find the best model for classification. For each Classification Model, 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.

## 3.4. Data from User

The user is expected to submit mushroom characteristics data, adhering to predefined formats and preprocessing guidelines. Upon submission, the system will provide binary classification predictions indicating whether the mushrooms are edible or poisonous.

## 3.5. Data Validation

Here Data Validation will be done, given by the user

## 3.6. Deployment

We will be deploying the model to AWS.

# 

# 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 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 |
|  | to the application 3. User is logged in to the application |  |
| Verify whether KPIs modify as per the user inputs for the user's health | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | KPIs should modify as per the user inputs for the user's health |
| Verify whether the KPIs indicate details of the suggested recipe | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | The KPIs should indicate details of the suggested recipe |