ARCHITECTURE

SPAM-HAM CLASSIFIER

|  |  |
| --- | --- |
| Written By | Suraj Joshi |
| Document Version | 1.0 |
| Last Revised Data | 20 Jan 2022 |

# 1 Document Version Control:

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1.0 | 20/01/2022 | Suraj Joshi |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1 Document Version Control: 2](#_Toc92905343)

[Table of Contents 3](#_Toc92905344)

[1.Introduction 4](#_Toc92905345)

[1.1 What is Low-Level design document? 4](#_Toc92905346)

[1.2. Scope 4](#_Toc92905347)

[2. Architecture 5](#_Toc92905348)

[3. Dataset 5](#_Toc92905349)

[3.1 Dataset Overview 5](#_Toc92905350)

[4. Logging 6](#_Toc92905351)

[5. Database 6](#_Toc92905352)

[6. Deployment 6](#_Toc92905353)

[7. Proposed Solution 6](#_Toc92905354)

[8. Model Training/Validation workflow 7](#_Toc92905355)

[9. User I/O workflow 7](#_Toc92905356)

[4 Test Cases 8](#_Toc92905357)

# 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 flight fare estimation 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.

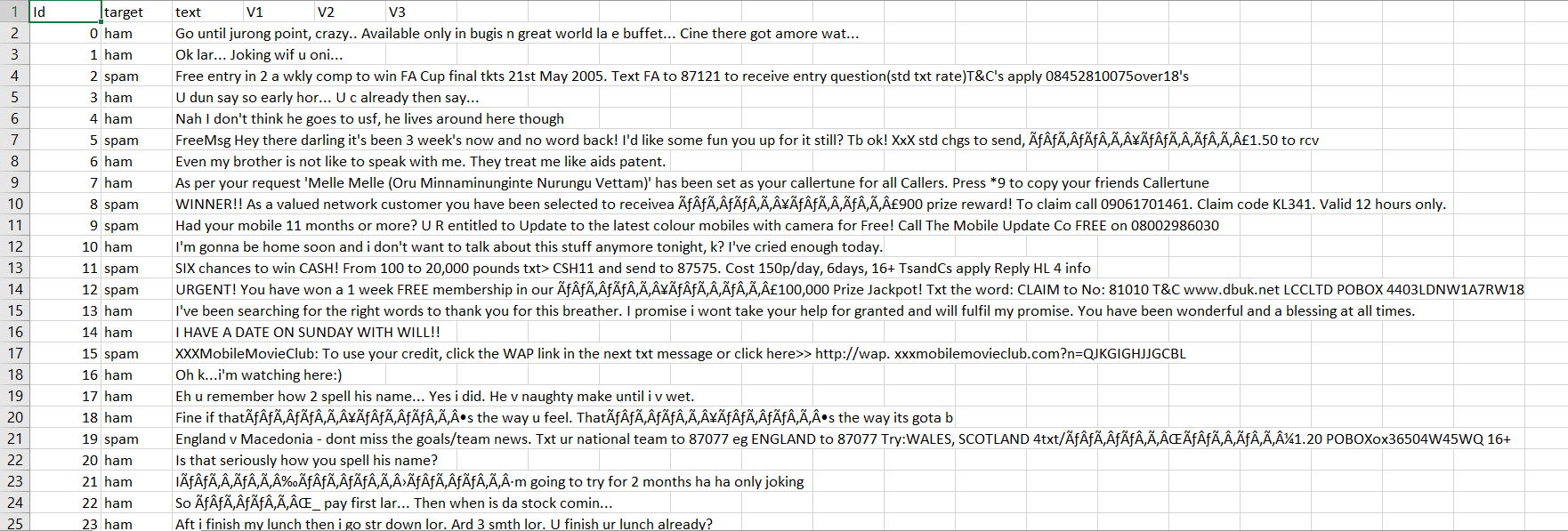
## 1.2. Scope

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

## 3.1 Dataset Overview

The training dataset consists of 5 columns and every column datatype is string except the first one that is Id.

Testing Data is just a message.

# 4. Logging

Logging is very important to keep track of the activities performed by our application. I have used logging module to do so. All the logs either it is train or test, both will be present inside All\_logs folder. Logging helps us in debugging process also so it is mandatory to do.

# 5. Database

The database I am using is Cassandra db. System needs to store every request into the database and we need to store it in such a way that it is easy to retrain the model as well.

The system stores each and every data given by the user or received on request to the database.

# 6. Deployment

Deployment is done in AWS and it’s a production server.



Link: http://ec2-3-140-241-66.us-east-2.compute.amazonaws.com:5000/

# 7. Proposed Solution

Solution is very simple here. I am going to build a simple ML model which will be able to classify the message type based on the message words. Doing some research I got to kno w that Multinomial Naïve Bayes is the best one.

# 8. Model Training/Validation workflow

# 9. User I/O workflow

# 4 Test Cases

Test cases are given below

|  |  |  |
| --- | --- | --- |
| **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 Response time of url from backend model. | 1. Application is accessible | The latency and accessibility of application is very faster we got in heroku service. |
| Verify whether user is giving standard input. | 1. Handeled test cases at backends. | User should be able to see successfully valid results. |
| Verify whether user is able to see input fields on logging in | 1. Application is accessible 2. 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 logged in to the application | User should be able to edit all input fields |
| Verify whether user gets Predict button to submit the message. | 1. Application is accessible 2. User is logged in to the application | User should get predict button to submit the message. |
| Verify whether user is presented with recommended results on clicking predict button. | 1. Application is accessible 2. User is logged in to the application | User should be presented with recommended results on clicking predict. |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is logged in   to the application and database | The recommended results should be in accordance to the selections user made |