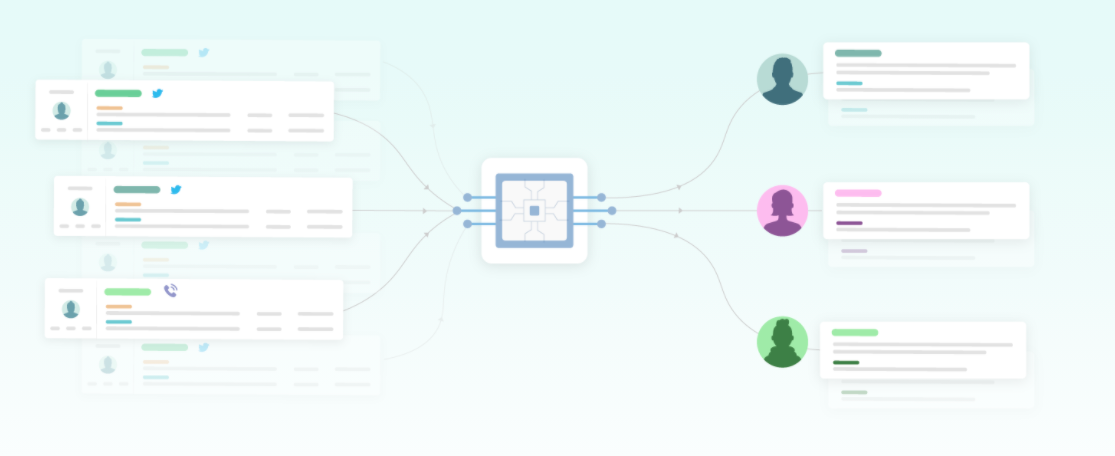
**Low Level Design Document**

**VER - 1.0**

**Automatically assign tickets in CRM tools to particular team.**



**Prepared by: Manpreet Singh**

**Abstract:**

Customers/users contact companies via multiple channels, from social media platforms and review sites to email and live chats – any time of the day, wherever they are. With the growing number of mobile users, customers can easily access these services, adding to the volume of tickets generated. IT tickets are the generalized term used to refer to a record of work performed (or needing to be performed) by an organization to operate the company’s technology environment, fix issues, and resolve user requests. These tickets can be raised through the web, mobile app, emails, calls, or even in customer care centers.

When an issue or support ticket drops into the helpdesk, first it needs to be processed, provide resolution at first instance with the help of chatbot or assigned a tag or category so that these tickets are then routed to the right agent for resolution, according to the department that would be a perfect fit for the label of that ticket. Hence, the ticket needs to be correctly labeled, so that there is no waste of time and resources in routing the ticket to the right agent, only adding to the resolution time of that ticket.

1. **Introduction**

**1.1 Need for Low-Level Design Document**

The purpose of this document is to present a detailed description of the Chatbot system and explain the purpose and features of the system, the interfaces of the system, scope of the system, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

In any industry, Incident Management plays an important role in delivering quality support to customers. An incident ticket is created by various groups of people within the organization to resolve an issue as quickly as possible based on its severity. Whenever an incident is created, it reaches the Service desk team and then it gets assigned to the respective teams to work on the incident.

The Service Desk team/Support team will perform basic analysis on the user's requirement, identify the issue based on given descriptions and assign it to the respective teams.

The manual assignment of these incidents might have below disadvantages:

More resource usage and expenses.

Human errors - Incidents get assigned to the wrong assignment groups

Delay in assigning the tickets,

High resolution time.

Delay in other productive tasks

If this ticket assignment is done only for the incident which needs human intervention and rest of the queries/incident if resolved then and there itself then it can be more cost-effective, less resolution time and the support team can focus on other productive tasks.

**This project shall be delivered in two phases:**

Phase 1: All the functionalities with PyPi packages.

Phase2: Integration of UI to all the functionalities.

**1.2 Scope**

Chatbot will provide first hand resolution to maximum queries and will use AI-based classifier model to assign the tickets to right functional groups by analyzing the given description if resolution is not available and needs human intervention.

Chatbot will be integrated with web application to answer the queries and solve the issues which are recurring and commonly faced by most of the users using previously solved queries data collected over period of time from the users.

**1.3 Constraints**

There are certain general queries for which the answer can’t be provided by chatbot and direct human intervention is required. So, ticket will be assigned to functional groups directly.

**1.4 Risks**

Document specific risks that have been identified or that should be considered.

**1.5 Out of Scope**

Delineate specific activities, capabilities, and items that are out of scope for the project.

1. **Technical solution design**

**2.1 Pre-requisites:**

**To develop chatbot model**-Python, NLP Libraries (like Spacy, NLTK), TensorFlow 2.0/ Keras, Hugging Face

**To develop web application-** HTML, CSS, Java Script

**2.2 Dataset**

Dataset is provided by ineuron.ai and available on share drive with folder name final\_data

<https://drive.google.com/drive/folders/1JLwcdDGYxEkPX0xXKzh5DgFGsUh0Rkjq>

**2.3 Technical Stack**

DB- Mongodb

Language - Python

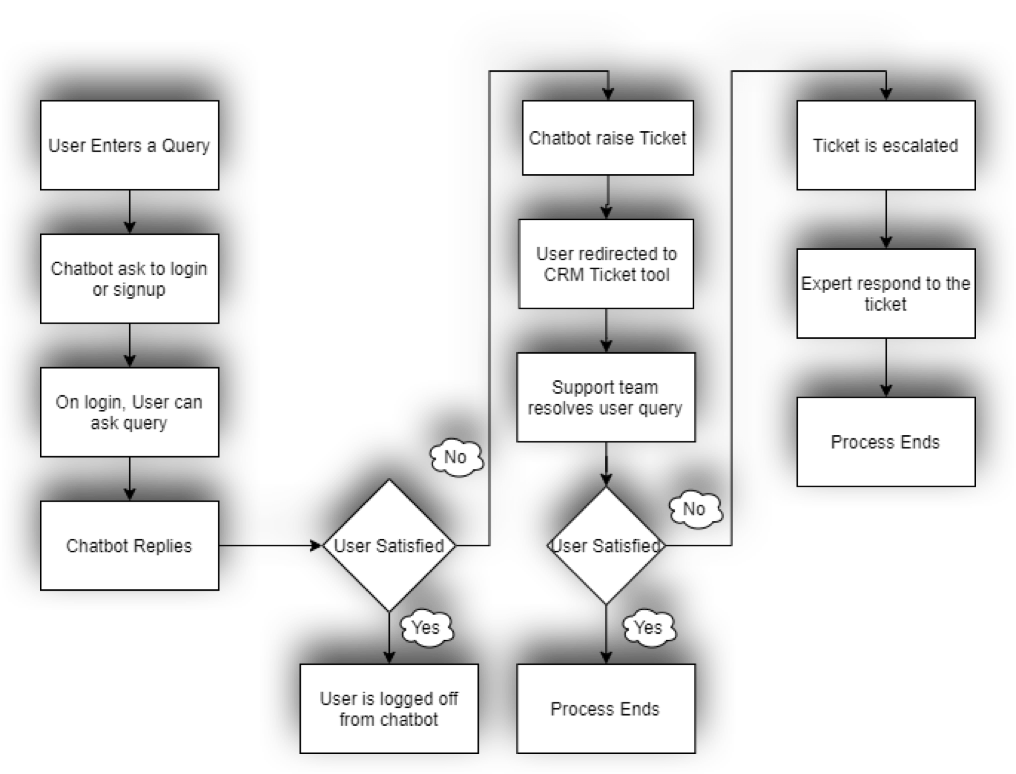
Core model - Distillbert

Data - Inhouse data

Frameworks - Flask, Django

Cloud - AWS

**2.4 Flow Diagram**



**3. Data Transformation and pre-processing steps**



**3.1 Data Mining**

Data mining is the process of extracting useful information from large amounts of data.

Different Data Mining techniques:

All the datasets are were append using MS Excel, POWER BI and Python but we chose to go with python to remove manual task

**3.2 Data Cleaning**

When we are dealing with a lot of data, we have to look for Null Value, Duplicate value present in the data frame.

There are 174 files with content and intent data, we followed below process:

* Merge all files into one file – This can be achieved by Manually or in automatic way, However, we preferred to use automatic process by using python script which merged all files in to one is quick time.
* Once Null Value and duplicate value are removed, we have to remove all the columns which are not required for building the model.
* We will be left with only columns after performing cleaning activity namely “Content” and “Intent”

Dividing complete dataset into two sections:

1) Technical Queries (ML, DL, BA, Other)

2) Non-Technical queries

Further dividing "Technical Queries" section into four sub-sections:

1) Machine Learning

2) Deep Learning

3) Business Analytics

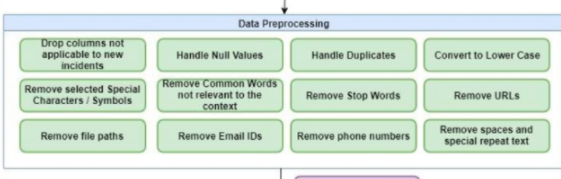
4) Other (Queries which are common between ML, DL, BA and even other generic queries)

By adding Category column in the dataset, we will have data as below

38429 rows and 3 columns

**3.3 EDA**

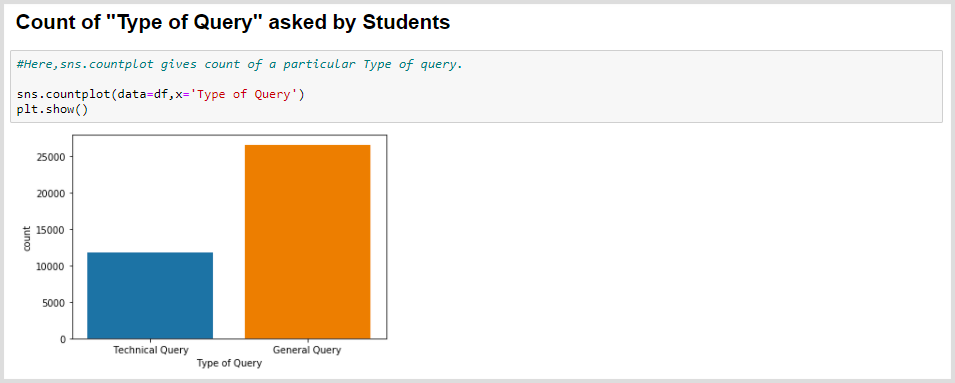
**3.3.1** Data Pre-processing



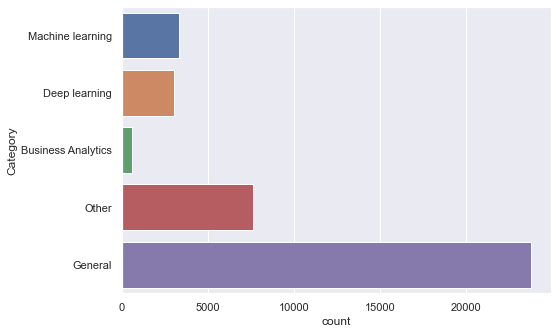
Total Unique Intent- 177

Data frame shape **-** 38429 rows × 3 columns

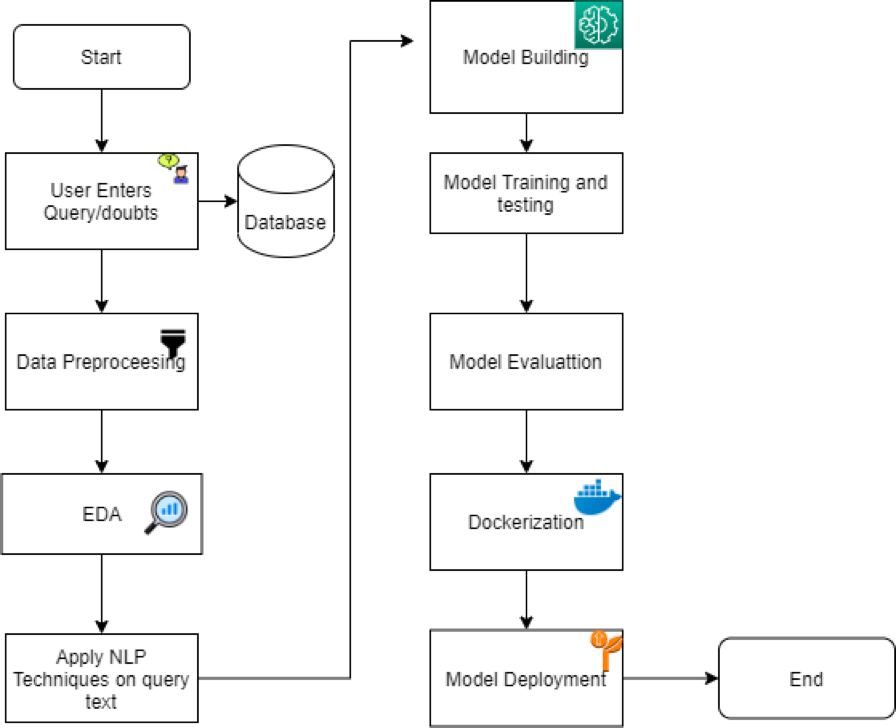
# **3.3.2** Count plot for Type of Query



# **3.3.3** Count plot for all type of Intents Category

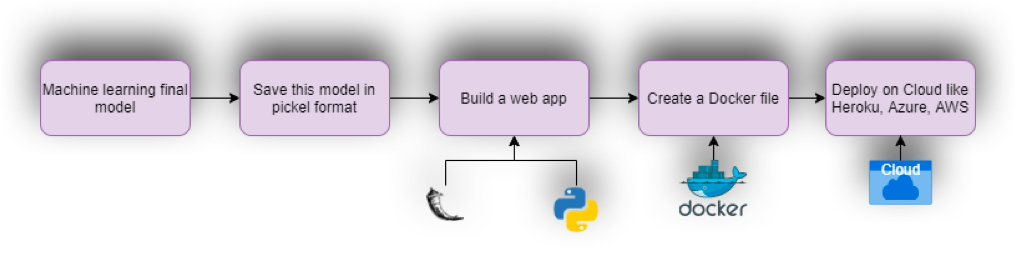


1. **Technical process flow**



1. **Deployment strategy**

The deployment of machine learning models is the process of making models available in production where web applications, enterprise software and APIs can consume the trained model by providing new data points and generating results/outcomes for business Decisions.



# **Step1 — Model building**

The machine learning model which is finalized is saved as a pickle/Joblib file.

**Step2 — Building Web Application**

Now that our machine learning pipeline and model are ready, we will start building a web application that can connect to them and generate outcomes on streaming data in real-time. This application will support ‘Online’ as well as ‘Batch’ processing’s through a DB. There are two parts of this application:

Front-end (designed using HTML, CSS, JS) – Basic web development frameworks like this will be used to build beautiful custom web-apps for machine learning.

Back-end (developed using Flask in Python)-

The back-end of a web application is developed using a Flask framework. It is a framework that allows you to build web applications. Flask runs on a server. This can be in the environment of the client or a different server depending on the client’s requirements. When running python app.py it first loads the created pickle file. Once this is loaded you can start making predictions.

**Step3 — Create a Docker file**

[**Docker**](https://www.docker.com/why-docker) is one of the most popular, open-source container technologies that allows you to build, run, test, and deploy distributed applications

There are three parts in docker containerization:

Container: This describes a virtual environment that bundles the application code with all the binaries and libraries that are required to run an application. Since the container includes all of its dependencies, you don’t have to install anything on the host operating system, keeping it separate and pristine.

Container orchestration:

This refers to coordinating behaviors for containers and between containers, such as scheduling, resource management, and load balancing. In complex or dynamic ecosystems, teams will use an orchestration engine to control, manage, and automate this activity.