**High Level Design Document**

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

**Problem Statement:**

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

Currently, queries received to iNeuron team get resolved through Skype which is not efficient as queries need to be assigned manually to iNeuron team members. So, in this project, we will be automating the query resolution system by creating a model that automatically detects the query type, provide the resolution itself if it's a frequently asked query and assign it to the correct team if it’s not able to provide the resolution.

**Objective:**

Objective of the document is to provide a brief overview of the technical architecture of the model of the chatbot which will be built to automate the query resolution system.

# **Prerequisites**

This Documents assumes familiarity with Below approaches:

Machine Learning Technique:

Like K Nearest Neighbor, Naive Bayes, Logistic Regression, SVM, Random Forest, gradient Boosting Decision tree, Model Calibration

Deep Learning & NLP Techniques:

Like Convolution Neural Networks, LSTM, Descriptive Statistics, Underfitting, Overfitting, Probability, Text Processing and TensorFlow and Keras Data Frames, Transfer Learning approaches and many libraries, etc.

AI Platform:

Applications Deployment and GIT Repository

## **Data Collection**

The Organizational Skype Tool was the primary source for our project data related to the queries. The data was collected for the resolved queries.

**Customers Dataset** – iNueron\_dataset.csv

This dataset has information about the queries asked to iNeuron team from skype support. Use it to train our chatbot using ML and NLP(Transfer Learning) techniques.

List of features identified as follows —

ID - Serial Number

Display\_Name - Customer Name

Content - Query

Intent - Specified Query like-Our query related to ML/DL/CV/NLP

Original\_arrival\_time - ID columns represents when we received data(Timing)

**Flow Diagram – Explanation**

Following are the key components of a conversational chatbot architecture:

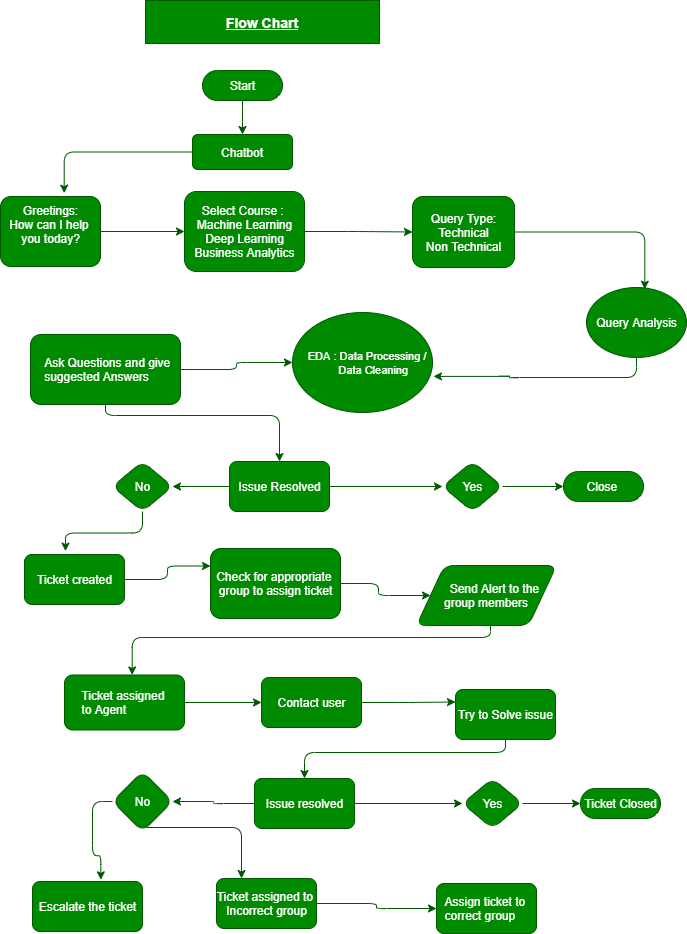
* BOT Service
* NLP Engine
* Machine learning
* Database
* **Bot Service** – After receiving input from the user Bot service will do the initial process. Both services mainly contain context tracking and conversation manager. Conversation Manager helps you recognize moments when you can take action to improve the customer experience. Within the Conversation Manager, Context Services helps you to recognize the moment and the Business Rules help you to take action. Contextual awareness refers to knowing who the customer is, what they want, and where they are in this process.
* **NLP Engine** - NLP Engine is the core component that interprets what users say at any given time and converts the language to structured inputs that the system can further process. Since the chatbot is domain specific, it must support so many features. NLP engine contains advanced machine learning algorithms to identify the user’s intent and further matches them to the list of available intents the bot supports.

**NLP Engine further has two components:**

* **Intent Classifier:** Intent classifier takes the user's input, identifies its meaning and relates back to one of the intents that the chatbot supports.
* **Entity Extractor**: Entity extractor is what extracts key information from the user’s query.
* **Machine Learning** - This is the key component in answering users’ frequently asked questions. Q & A system interprets the question and responds with relevant answers from the knowledge base. It has the following components
* **Manual Training:** Manual training involves the domain expert creating the list of frequently asked users queries and maps its answers. This helps the bot quickly identify the answers to the most important questions.
* **Automated Training**: Automated training involves submitting the company’s documents like policy documents and other Q&A type of documents to the bot and asking it to train itself. The engine comes up with a list of questions and answers from these documents. The bot then can answer with confidence
* **Database** – Database is used to store previous queries and training our model to address similar queries in future.

If queries cannot be addressed by Bot, then those queries will be forwarded to the technical team.

## **HLD:**



## **Data Mining**

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

Different Data Mining techniques:

# **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. Below are the ways to Remove the Null value and Duplicate Data from the Data frame.

After Reading all the dataset we have to first check whether it contains a null value or not.

After handling all the Null values, we have to look for Duplicate Data as part of Data cleaning.

For good performance of the model, we need a reasonable quantity of data with sound quality. The data collected for training should be able to capture the variances of the population and should represent the population best way possible.

# **Exploratory Data Analysis**

After data Cleaning and mining all the data set in the final set. Let’s Start Exploratory Data Analysis.

First we have a feature review Score ranging from 1 to 5. so we posed this problem as a multi classification task.

As a part of Exploratory Data Analysis, we can perform a univariate Analysis on Categorical Feature.

## 

## 

## 

## 

## 

## **Technical Stack**

The technical landscape of the chatbot system is shown below:



## 

## **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 Chatbot 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 through a DB/csv file upload. There are two parts of this application:

**Front-end (designed using Streamlit/HTML/CSS)** - Streamlit is an open-source Python library that makes it easy to build beautiful custom web-apps for machine learning and data science.

**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. A web application can be a commercial website, a blog, e-commerce system, or an application that generates predictions from data provided in real-time using trained models. Before we publish the application on Google Cloud platforms to test the web app locally. Open Shell Prompt and navigate to the folder where ‘app.py’ is saved on your computer and run the python file python app.py. Once executed, copy the URL into a browser, and it should open a web application hosted on your local machine (localhost:8000). Try entering test values to see if the recommendation system function is working.

* **Step3 — Create a Docker file**

A container is a type of software that packages up an application and all its dependencies so the application runs reliably from one computing environment to another. Docker is a company that provides software (also called Docker) that allows users to build, run and manage containers while Docker’s containers are the most common to containerize our application for deployment. We need a Docker image that becomes a container at runtime. A Docker image is created using a Docker file. A Docker file is just a file with a set of instructions. The Docker file is case-sensitive and must be in the project folder with the other project files.

* **Step4 —Deploy a ML pipeline on GKE/Cloud (AWS/GCP/Azure):**

Google Kubernetes is a powerful open-source system developed by Google back in 2014, for running and managing containerized applications across a cluster of applications.

1. Sign-in to the GCP console and go to Manage Resources and then click on Create New Project followed by importing the Project Code.
2. Set Environment Variable
3. Build the Docker image of the application followed by authenticating to the Container Registry.
4. Create Cluster
5. Deploy Application -To deploy and manage applications on a GKE cluster, you must communicate with the Kubernetes cluster management system
6. Expose the application to the internet.
7. Check Service

## **Conclusion**

An automated ticket/query assignment chatbot system that will help the organization in addressing queries quickly and can optimize their business in a better form.