**CHAPTER 1**

**INTRODUCTION**

The ticket automation is playing a huge role in maintaining a successful support operation and is helping to revolutionize the industry.

The days of the “gatekeeper” that directed customer support issues to the right people are over. Now, with customer support software and ticket automation, triggers can be created so if a ticket is submitted via email or a customer portal it goes to a specific agent or group. For example, a trigger where the subject line contains the word “how” (i.e. how do I add a user) could automatically route all emails to the training group.

And it also **Remind customers to follow up, Close tickets automatically, Get alerts for urgent tickets, Weekends and holidays**

In this project we are trying to perform few of the basic ticket automation operations via uipath.

**1.1 PROBLEM DEFINITION**

* Eliminate manual intervention in ticket creation, so raise a ticket based on the complaint mail. A database used for storing and administering all types of data required for efficient and accurate functioning of the bot.
* If the details are incomplete (e.g. customer id is missing), send a mail to customer asking for missing details and Link the subsequent responses from the customer to the original ticket.
* Recognize the bounced mails and initiate appropriate action via sending auto response to template-based mails

**1.2 OBJECTIVE OF THE PROJECT**

The objective of this project is to contribute to the solution of the problem of eliminating manual intervention in ticket creation, so raise a ticket based on the complaint mail. The main objectives of the project are as follows:

* **Uipath** : To develop a bot in order to fetch the mail from the mail server.
* **Python** : To develop a python module in order to assign a priority based on analysing the key words
* **Database:** To develop a database were all the relevant information about questions, answers, keywords, logs and feedback will be stored.
* **Algorithm:** To develop a keyword matching algorithm using NLTK in order to identify the keywords and tokenize.

**1.3 SIGNIFICANCE OF THE PROJECT**

* Effective in both Installation and Working with the Queues.
* Previously solved solutions are stored in a separate Database.
* When same queries arise by the other users, then it match to the saved results.
* This makes the Ticketing System Faster Comparatively.
* Applies certain algorithms (Sequential / Priority) to process the queries even more faster.
* Splits entire queries by word tokenization.
* Missing Details and Fraud are identified by NLP, then Updates the information to existing tickets accordingly.

**1.4 OUTLINE OF THE PROJECT**

* Get the complaint mails from the users .
* Segregate the mail, understands the user queries and assign the ticket to the particular complain mail
* If the queries already solved, then replay with the existing solution and close the ticket.
* If the problem is not solved then perform the analysis on the mail and raise the ticket to appropriate person.

**CHAPTER 2**

**LITERATURE REVIEW**

Management and maintenance of IT infrastructure resources such as hardware, software and network is an integral part of software development and maintenance projects. Service management ensures that the tickets submitted by users, i.e. software developers, are serviced within the agreed resolution times. Failure to meet those times induces penalty on the service provider. To prevent a spurious penalty on the service provider, non-working hours such as waiting for user inputs are not included in the measured resolution time, that is, a service level clock pauses its timing. Nevertheless, the user interactions slow down the resolution process, that is, add to user experienced resolution time and degrade user experience. Therefore, this work is motivated by the need to analyze and reduce user input requests in tickets’ life cycle. To address this problem, we analyze user input requests and investigate their impact on user experienced resolution time. We distinguish between input requests of two types: real, seeking information from the user to process the ticket and tactical, when no information is asked but the user input request is raised merely to pause the service level clock. Next, we propose a system that preempts a user at the time of ticket submission to provide additional information that the analyst, a person responsible for servicing the ticket, is likely to ask, thus reducing real user input requests. Further, we propose a detection system to identify tactical user input requests. To evaluate the approach, we conducted a case study in a large global IT company. We observed that around 57% of the tickets have user input requests in the life cycle, causing user experienced resolution time to be almost twice as long as the measured service resolution time. The proposed preemptive system preempts the information needs with an average accuracy of 94–99% across five cross validations while traditional approaches such as logistic regression and naive Bayes have accuracy in the range of 50–60%. The detection system identifies around 15% of the total user input requests as tactical. Therefore, the proposed solution can efficiently bring down the number of user input requests and, hence, improve the user-experienced resolution time.