ASHESI

ASHESI UNIVERSITY COLLEGE

A CENTRALIZED REPOSITORY TO ENHANCE CUSTOMER SERVICE USING DATA MINING AND MACHINE LEARNING

APPLIED PROJECT

B.Sc. Computer Science

Agatha Adjoa Maison

April 2016

ASHESI UNIVERSITY COLLEGE

A Centralized Repository to Enhance Customer Service using Data Mining and Machine Learning

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

Agatha Adjoa Maison

April 2016

DECLARATION

I hereby declare that this applied project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

…………………………………………………………………………………………

Candidate's Name:

…………………………………………………………………………………………

Date: …………………………………………………………………………………………

I hereby declare that preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

…………………………………………………………………………………………

Supervisor's Name:

…………………………………………………………………………………………

Date:

…………………………………………………………………………………………

ACKNOWLEDGEMENT

I would like to express my profound gratitude to all who assisted me in diverse ways to complete this project.

ABSTRACT

In the quest to reduce customer churn rate and retain existing customers, organizations have resorted to investing fortunes in their customer care services, which proves to be a relatively cheaper means of staying in business. In this regard, this project sought to explore a less costly way of providing quality customer care services to an organization's clients in order to keep them satisfied. As it stands today, there is a growing number of digital customer care owing to the fact more clients have increased their online interactions. Organizations with customer satisfaction as priority have invested in these e-care services to better serve their customers. The gap identified however is the lack of a centralized repository to store and track all concerns raised by customers. To bridge this gap, a prototype of a trouble ticketing system was developed to allow clients to issue trouble tickets whenever faced with a difficulty. This system in addition integrates and monitors company systems using Nagios IT Infrastructure Monitoring, conducts sentiment analysis with Datumbox Machine Learning Framework, analyzes and generate reports on the efficiency of the organization in dealing with their customers. Since the world today revolves around making sense out of previous occurrences to forecast the future and prepare adequately towards it, this system finds patterns in the errors received in order to make predictions on which category of services provided by the company is likely experience more error logs using the Amazon Machine Learning Web Service.

TABLE OF CONTENT

List of Figures

List of Tables

Chapter 1 : Introduction

1.1 Background

The most traditional means customers follow to resolve issues they face while using a service or product by an organization is by calling, sending emails or accessing live chat dialog boxes on the organization's website. The cumbersome processes a customer has to follow; long voice prompts, long waiting/holding time involved in getting connected to a personnel and the unreliability or unavailability of some customer service call centers and online chat services at the crucial time of need could be very unnerving.

The conventional wisdom which defines the customer as king is fizzling out because most companies are failing to keep their eyes on the ball – putting customers first. Businesses are now in a constant quest to outdo competitors in providing unique services but this only results in increasing similarity in physical products and services they offer. In failing to give priority to customer needs and with undifferentiated products on the market, the likelihood of customer attrition is high (Barwise & Meehan, 2004).

According to Abiw-Abaidoo (2011), MTN, a telecommunication company in Ghana, recorded a 7.2% customer churn which was attributed to poor customer service provision in a study conducted in Kumasi, Ghana in October 2011. Also, a survey conducted by the American Express in 2011 recorded that, about 78% of customers bail out on transactions or decline to make intended purchases because of poor customer service experiences (American Express, 2011). Not rendering the necessary attention to the needs of customers in a competitive market increases the percentage of customer attrition in a company which leads to a significant loss of revenue.

Today, most organizations have resonated with the fact that it is ideal to focus more on customer retention than to scout for prospects owing to research that proves that there is a 60 – 70% probability of selling to an already existing customer as opposed to a 5 – 20% probability of selling to a new prospect (Hull, 2013). In light of this, most organization are heavily investing in their customer care services to enable them reduce the rate of attrition. For instance, MTN invested about 274 million Ghana cedis and 311 million cedis in 2013 and 2014 respectively in their quest to provide better services to their clients. They also introduced the 'Customer Loop Feedback', which randomly samples customers after they have been served by customer service agents to generate feedback with respect to the quality of the service they received. This project, was embarked on to provide immediate feedback to the customer care team for continuous improvement of their services to customers (Lumor, 2014). This attempt placed their company above its competitors and enabled them sweep away four awards at the 2015 Ghana Telecom Awards (News Ghana, 2015).

1.2 Problem

The rising surge of the use of the internet has created an avenue for customers to voice out their frustrations. These new communication channels include but are not limited to the use of Twitter, Facebook, Whatsapp, Instagram and the traditional phone calls and emails. A lot more customers are going online to find solutions to their problems rather than rely on the traditional means (Ban, Gbahoué, & Schneider, 2013). In this vein, most organizations have adopted the use of these digitized channels to meet the needs of their customers faster. Information received through these channels are of high importance to these organizations as the customers themselves. However, data received from these channels are in isolation and little insight can be drawn from them as a unit. There is no centralized repository to store and track all data received through these channels. Bringing together all customer concerns and complaints could be a great way to draw meaningful deductions and analysis to aid better decision making with customer satisfaction as a driving force. However, with the absence of a centralized repository arriving at such insights is a challenge.

1.3 Objective

The goal of this project is to build a generic but customizable trouble ticketing system that integrates customer complaints from the available communication channels (social media) while making this data accessible for efficient analysis in order to find patterns and possibly make accurate predictions for the management of the subscribed organization. This will improve customer service to a large extent, grant customers a higher satisfaction, improve customer experience with the use of services, help organizations maintain their customers at a reduced cost and positively brand these companies as proactive, rather than reactive. The system will act as a rich self-service tool due to the knowledge base feature that provides customers easy access to self-help solutions to problems as well as answer frequently asked questions. This will shorten or completely eliminate the time it takes customers to reach customer care personnel in order to resolve a problem. For issues that need technical assistance, a trouble ticket will be issued to the right personnel who would work on the issue at his/her earliest convenience in order to satisfy customer needs. This system will also allow the integration of existing company systems for monitoring purposes. This feature could also generate tickets as and when these company systems develop faults that need to be resolved. After series of data collection and issue tracking, the system would analyze data received using machine learning techniques to find patterns or trends in the kind of data received as well as make predictions on when next the problem could be encountered. This will enable the timely detection of both external and internal factors that contribute to the problem.

1.4 Related Work

This subsection puts this proposed system into perspective with other related products already on the market. This proposed trouble ticketing system seeks to mine information from integrated and monitored company systems as well as the web portal which allows customers report their frustrations or faults. What sets this system apart from other existing solutions like Zendesk, Kayako, Nanorep, ServiceDesk Plus is its ability to integrate with other company systems in order to issue tickets to draw attention to faults. Its ability to source data from social media and record the general perception customers have of the company, send periodic notifications to support team personnel when a ticket’s deadline is overdue may be common to other existing solutions. However, its feature to find trends, learn and analyze data to make predictions of possible errors/faults gives a clearer picture of some technical issues that may have affected the smooth delivery of services. This feature is a major distinguishing factor that may not be common in these existing solutions altogether.

Chapter 2 : Requirements

2.1 Overview

The purpose of this section is to outline the requirements for the generic but customizable trouble ticketing system that will help meet the needs of clients in the shortest possible time as well as empower organizations to work proactively towards the needs of their clients. This document will give a detailed description of the functionalities and all intended features of this system.

This section contains a general overview of the background for and requirements of this TTS. Section 2.2 gives insight into the requirement gathering process, whereas section 2.3 give the scope of the project. Section 2.4 describes the system into details covering areas such as product/software perspective, software functions, operating environment, design and implementation constraints and assumptions and dependencies. Section 2.5 touches on the system specifications which contains software requirements with high levels of detail to inform any developer, designer or tester adequately for the implementation of the requirements of this system. Section 2.6, however, captures all external interface requirements. This section looks at the various user, hardware, software and communication interfaces necessary for the development of this system. Lastly, section 2.7 deals with the Non-functional requirements of this software.

2.2 Requirement Gathering

For requirement elicitation and gathering, interviews and observations were the main methods used. The resources explored were stakeholders of the system (companies and individual clients), existing trouble ticketing systems and data mining and machine learning techniques. Information sort after from these resources include:

- How do companies resolve customer related issues?
- How do customers send complaints to their various service providers?
- How long does it take these companies to address customer problems?
- Why is it necessary to track these reported problems?
- How are issues tracked?
- What are the various techniques of data mining and machine learning available?
- Which tools are available and relevant to this project?
- What kind of data will this project be dealing with?
- What does proactive customer service entail?

2.2.1 Requirement Analysis

Analyzing input from the stakeholders and observations showed that:

- Putting in place a process that will alert the company of a potential problem before the customer gets to know about it is very helpful and thus must be considered as part of the system.
- Reaching out to customers when there is a system failure that will affect their activities is very helpful. This will help reduce the amount of traffic on communication channels of the company.
- Staying in constant communication with a customer when a ticket is being resolved is necessary.
- Engaging various communication channels to aid in eliciting the views and problems of customers goes a long way to satisfy customers.

- Tracking the occurrence of a particular error and making accurate predictions of when they are likely to occur gives the company a head start to providing proactive services.
- Customers don’t mind self-help services to solve problems they face.

2.3 Scope

This Trouble Ticketing Software is a web portal that allows customers of a company to log errors/complaint/faults they find in using a service provided by the given company. The system also integrates with the given company’s existing systems in order to proactively detect and fix faults as and when they occur. Lastly, this system will study and find trends or patterns in the frequency and arrival of errors or faults in order to make predictions.

This software product will:

- **Issue tickets**: This will allow customers and other integrated company systems to submit tickets to company personnel for review and resolving. Customers will do this through a web portal whereas company systems will be continuously probed for errors or downtimes.
- **Send notifications on ticket arrival**: Based on the configurations made by the company’s administrator, there will be periodic reminders of tickets that need to be attended to. These notifications will be sent out to the assigned personnel or department after the set timer goes off.
- **Conduct sentiment analysis**: The system will mine/extract the objective feelings of its customers online to allow management make better decisions.

- **Find trends and make predictions**: Here, the system will study the data (errors or complaint) received, find trends and make predictions as to what and when the next set of errors should be expected. This may draw managements attention to an overlooked or unforeseen fault that may have occurred in their system.
- **Generate reports**: This will allow the management of the organization to have an overview of how well their customers are being served. The system will give an account on the efficiency of customer service team as well how customers perceive the services of the company.

This software product will not:

- **Resolve tickets generated**: As it is, this software will not resolve tickets submitted by customers or monitored company systems.
- **Keep track/manage customer accounts**: This software will not keep records of the lifecycle of customers as a typical customer relationship management system does. As such, customers will not be required to create user accounts to use the service.

This software will be used by companies who have high interest in providing quality products and services to clients by regarding customer feedback and complaints and proactively working on system downtimes before it reaches the customer. This system will enable the companies track the in and out flows of tickets submitted and how urgently they were treated by company's personnel. This system will give the administrator an idea of how efficient his personnel are. Also, the company will have a fair idea of the sentiments of its clients through the feeds received through social media.

2.4 Overall Product Description

This section provides an overview for the requirements outlined in this document to facilitate understanding of the requirements discussed in section 2.5 of this document.

2.4.1 Product Perspective

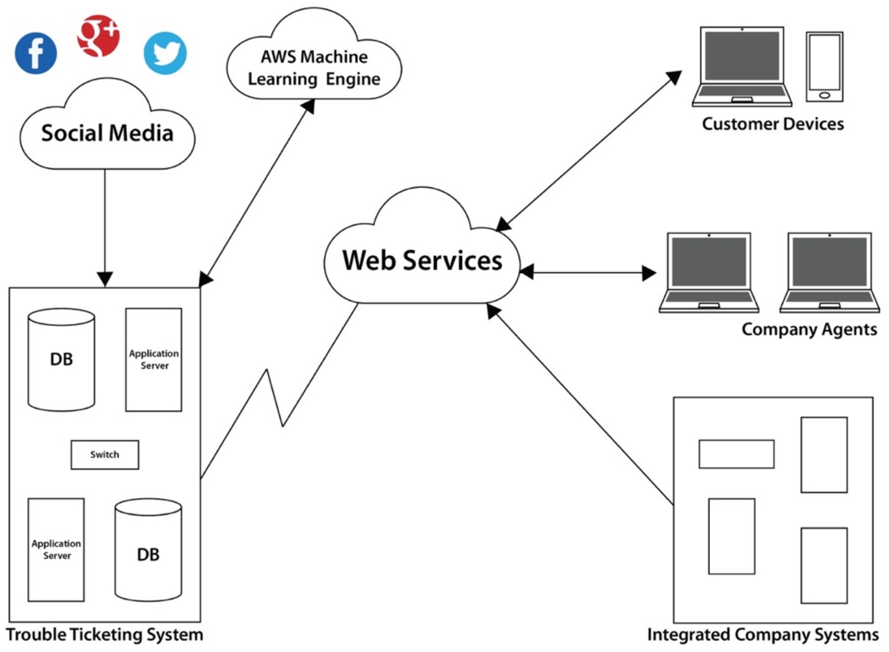


Figure 2.1 An overview of the trouble ticketing system

This software product is a self-contained, independent web-based application. The server-client model makes use of central web and database servers which host data relevant to signed up organizations, their support team personnel and customers. This group of clients (administrator, personnel, customers) establish connection from their various devices through the Internet to the server. The server stores and manages all data and processes relevant for the smooth operation of this application. It relays the output of processes to clients after each request.

Figure 2.1 illustrates a general overview of the software and the major players that make it work. The scope of the software will be classified into 2 – the system and the users. The system side comprises database and application servers which interact with social media and Amazon Machine Learning Web Service on a higher level. Through the web application, users – company systems, company agents, administrators and company clients can access the services of this software.

2.4.2 Product Functions

The functions of this software include:

- **System Configuration**: The administrator of an organization will be required to configure the system based on the needs and preferences of the company. Configuration will cover selected fields for tickets, company details and preferences, knowledge base configuration and addition, type of reports to generate, categories, departments and the members that make up the support team.
- **Ticket Generation**: Once configurations are complete, customers of an organization will be able to access the ticket form of the company they intend to submit to. Customers can then submit tickets to specific departments of the company. Company agents can also submit tickets on behalf of customers who phone in. These tickets are generated with a unique id for clear and easy identification.
- **Ticket Management**: On the company personnel side, submitted tickets are attended to. Here, tickets could be (re) assigned to another personnel for resolution. Tickets can be sorted based on given parameters, archived, have statuses changed i.e. open, in progress or closed. Personnel will receive frequent notifications on unresolved tickets to draw attention for immediate resolution.

- **Data Analysis and Predictions**: Data received over a number of days will be learned by the Amazon machine learning engine to find patterns in order to make predictions as to which errors/faults to expect either from the monitored systems or from customers.
- **Sentiment Analysis**: The system will conduct sentiment analysis to report the general objective feelings of the company’s customers on a particular product or service.
- **Report Generation**: The system will generate graphical reports to cover the general output of the company with respect to ticket resolution and customer satisfaction.

2.4.3 Operating Environment

This system is a web-based application that will be hosted on a web server. The enabling operating environment that will facilitate the smooth running of the software is a MySQL database server (version 15.1 Distribution 5.5.44-MariaDB), Apache web server (version 2.4.6) on a Linux operating system. It can be viewed by any web browser on any device.

2.4.4 Design and Implementation Constraints

Some companies may have regulations limiting access to their company systems because data privacy is important. This would make it challenging or even impossible to automate the creation of tickets when these systems develop faults.

The service is built to work for various companies across sectors. However, company systems may not support the interfaces that this system will provide. For example, a company may have systems that only supports serial connections, without an internet connection, etc. Such a system will be unable to work with this trouble ticketing system.

Management of the various companies must be ready to believe in the efficiency of system for them to adopt it as part of their work process. There will be no end to this system should it not be used.

2.4.5 Assumptions and Dependencies

This section lists the factors that affect the requirements of this system.

2.4.5.1 Assumptions

For the smooth running of the system, it is assumed that:

- Support team personnel are always available to respond to tickets.
- Support team members take prompt action when a ticket is issued.
- Support team members are highly trained technically to handle all anomalies that are reported.
- The company administrator correctly enters the steps taken to resolve a problem in order to build a credible knowledge base.
- All users of this systems are able to use simple web application.
- There is an uninterrupted supply of internet.
- Company systems can easily be integrated into the system.
- Browser versions used support HTML5.

2.4.5.2 Dependencies

- **Time Dependency:** Due to a rigid time constraint, only the essential features of this project will be given higher priority. These include submitting of tickets, manual resolution of tickets by an agent, sentiment analysis and error prediction. The remaining lower priority constraint can be implemented at a later date. Such features include usability improvement, mobile application for convenience of operations, automated ticket assignment and natural language processing.

- **Email and SMS Notifications:** The system will depend on emailing solutions from a third party system to send notifications to clients and company personnel based on the configuration selected by the company. SMS solutions for customers and personnel who may not be accessible via email at a particular time will be available also.
- **Machine Learning Engine:** This trouble ticketing system will depend on a trained model from the Amazon Machine Learning Web Service to make predictions on faults and errors that are likely to be encountered by an organization.
- **Sentiment Analysis Engine**: As part of adding value to the software, a sentiment analysis tool will be depended on to mine the objective sentiments of the clients of an organization on social media predominantly Facebook and Twitter.

2.5 User Classes and Characteristics

The identified user classes are company clients, company personnel and the company administrator. Their general characteristics relevant to this system are as follows:

- These users must be literate.
- They must be able to read and write in order to effectively communicate their views (complaints, solutions) to the other party involved in the interaction.
- They should be conversant with the use of a simple web applications.

- Company personnel must be tech-savvy and competent.
- Company Administrator should understand technical terms that will be used in the generated reports.

2.5.1 Scenarios

This section gives narratives of how the various actors will like to interact with the system and what kind of activities that the system is required to support.

2.5.1.1 Client User Class

Customer A uses a service from company B for her daily activities at work. She encounters an error while using the service which needs to be addressed at the earliest possible time. She first visits the web portal of this company, checks out the knowledge base to see if there has been a posting of her problem with a solution. Not finding her exact problem, she decides to issue a trouble ticket to the company to get an agent working on her issue. She is presented with a ticket form that she fills out with the relevant details of her problem. The system auto-replies her submission and generates a ticket number for her to track the progress of her ticket. In a few minutes, she receives a notification from a company agent via email or SMS. She enters the ticket id of her submission is presented with all the details of her ticket including any responses she has received from the agent in charge of her ticket as well as her ticket status. She may respond to the message received from the agent on this same platform.

2.5.1.2 Personnel User Class

Mr. C, is part of the support team for company B. He receives notifications whenever new tickets are submitted by company clients. He can view all tickets that have been submitted to the company. He creates tickets on behalf of customers who phone in. This helps him track the conversation. Mr. C views details of a selected ticket, changes ticket statuses from ‘in progress’ to ‘closed’ depending on how much work has been done on it. He is able to respond to conversations started on a ticket. He is able to send his responses to the client via email or SMS. He can also log the summary of calls to tickets that he created on a client’s behalf. Other management options for a ticket include sorting which may be by priority, deadlines, department, assignments are made available to aid Mr. C through his duties. He also has a database of archived tickets that he can fall back on when he needs to reference them. Owing to the vast number of tickets he deals with, he usually makes searches to help him find tickets faster. In the event that he cannot work effectively on a ticket, he reassigns it to another support team member.

2.5.1.3 Administrator User Class

Mr. X, the administrator of company B needs an effective way to keep his customers satisfied. He comes across a great tool which would help him achieve this dream. He first registers/subscribes/signs up the company on the system. He then sets up the configuration his company will like to use based on available options on the system. He is able to manage his support team members, the type of ticket configuration that his company will like to use, content of the knowledge base, departments and categories available in his firm and furnishes the system with other required details. He personalizes what kind of output he will like to receive from the system in terms of generated reports. As an administrator, he can view and analyze the generated report in order to make informed decision.

2.5.2 Use Case Diagram

This section shows the user interactions with the system and the different relationships that exist between users and the system as well as the different use cases they are involved in.

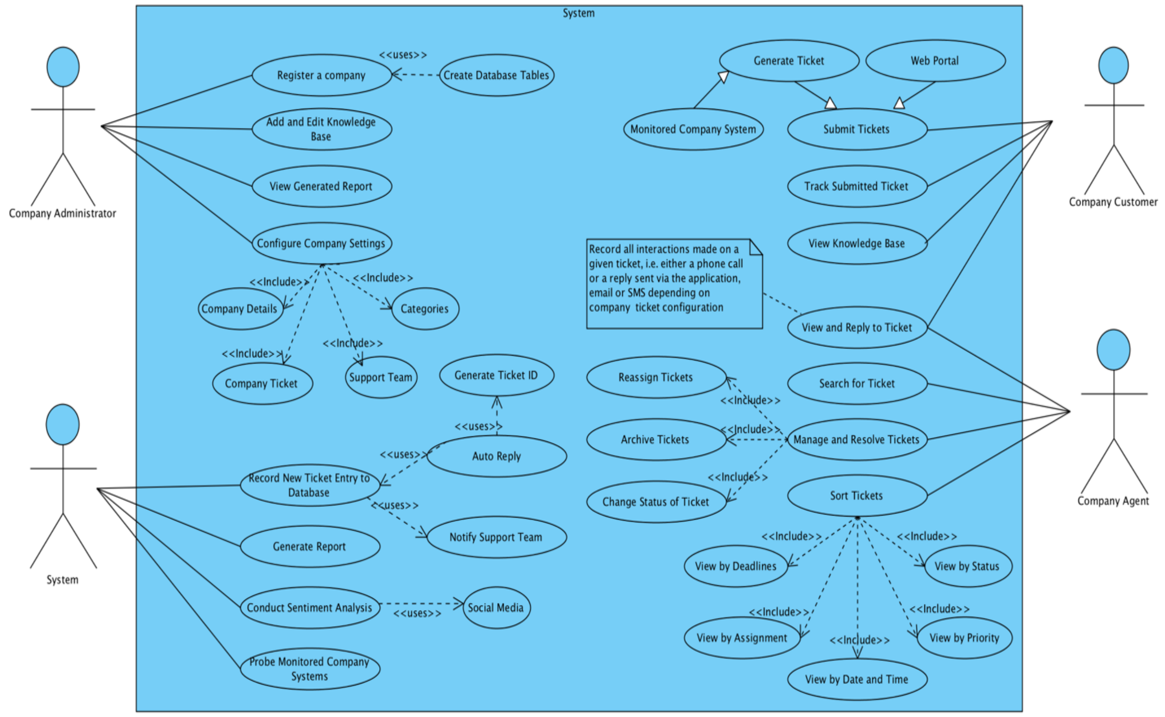


Figure 2.2 Use case diagram showing the requirements of actors of the system

- **Company Administrator**: He is in charge of registering and setting up a company. He initializes all the required tables needed for the company's operations as he configures the system. As part of his requirements, he manages the content of the knowledge base, ticket configuration/outlook and support team management.
- **Company Customer**: He is able to issue a ticket, track the progress of a submitted ticket and view the content of the knowledge base made available by

company administrator. He can also interact with a company agent through the system.

- **Company Agent**: He manages all the details of a submitted ticket to facilitate its resolution. Management includes sorting tickets, changing ticket status and interacting with a customer to clarify a submitted ticket.
- **System**: The system is in charge of conducting sentiment analysis, system probing, report generation of the data stored in the database. It also sends notifications to the both client and agent reporting ticket summaries and other details.

2.6 Specific Requirements

This section contains all the software requirements of this system that adequately informs designers to design a suitable system to satisfy these requirements. These functional requirements are organized according to the user classes of this system.

2.6.1 Client Requirements

This section gives a detailed breakdown of the client's requirements for this system

2.6.1.1 Description and Priority

This feature handles everything a client of a company can do with this system. It is of high priority to the system because it may be the first point of contact for a client who needs assistance. This feature greatly feeds data into the system for resolution and analysis since the system is centered around it to a large extent.

2.6.1.2 Functional Requirements

Table 2.1 Client requirement to submit ticket

REQ-Cl-1: A client should be able to submit a Ticket

Description	This requirement enables a client to issue a ticket to a company's support team.
Input	(Configurable by company's preference). Sender's name, email, phone number, subject, complaint, file, location, category, department.
Source	A form filled by a client.
Output	Alert user to enter valid data should wrong inputs be entered. Success or failure of submission. On success, send generated ticket id and summary to sender.
Destination	An alert in the Client's view; A notification to the Company agent.
Action	To complete this action, a client must fill a form with the inputs above given as fields. Click on a button to submit. An alert is then sent to the company agent's view.
Pre-condition	The client must have a problem/concern/challenge with the use of a service or product.
Post-condition	A ticket id is generated and sent to the client who submitted the ticket through an auto-reply message. Members of the selected department receive a notification of the arrival of a new ticket. The submitted ticket is made accessible to company agents.

Table 2.2 Client requirement to view ticket details

REQ-Cl-2: A client should be able to view ticket details

Description	This requirement allows a client to view details of a ticket submitted using a given ticket id.
Input	Ticket id.
Source	Client fills a text field with the ticket id.
Output	A wrong ticket id entered must be flagged as an error. The user is prompted to enter a valid ticket id. A correct ticket id entry gives access to all the details of a submitted ticket. This is subject to the selected configuration of a company. This may include Sender's name, email, phone number, subject, complaint, file, location, status (new, in progress, closed), agent assigned, time, date, interactions, priority.
Destination	Client's view.
Action	A client enters a ticket id into an input field to query the progress of the ticket.
Pre-condition	The ticket being queried must exist i.e. the ticket id must be valid.
Post-condition	Display of ticket details to client.

Table 2.3 Client requirement to respond to ticket

REQ-Cl-3: A client can respond to a Ticket

Description	This requirement allows a client to respond/interact with a company agent concerning the needs of a ticket for clarity and better service delivery.
Input	Ticket id, A message (response).
Source	Client
Output	Entry of an invalid ticket id should send an alert indicating that the id is invalid. Successful submission of reply posts the reply in an interactions panel with a timestamp.
Destination	Ticket details page.
Action	The client may type in a message into a text field and click a button in order to add to the ongoing conversation concerning the resolution of a ticket.
Pre-condition	The ticket must already be created.
Post-condition	View the latest added response to a ticket's conversation

Table 2.4 Client requirement to view ticket interactions

REQ-Cl-4: A client should be able to view ticket interactions

Description	This requirement allows a client to view interactions with respect to a ticket.
Input	Ticket id
Source	Client enters the ticket id in a text field and submits.
Output	Display interactions (conversations) that has gone on concerning a ticket i.e. message with timestamp. Entry of an invalid ticket id should send an alert indicating that the id is invalid.
Destination	Client's view
Action	The client may view/read previous conversations
Pre-condition	The ticket must already be created with existing conversation.
Post-condition	View responses to a ticket's conversation

Table 2.5 Client requirement to view knowledge base

REQ-Cl-5: A client should be able to view Knowledge Base

Description	This requirement allows a client to view inputs added to a company's knowledge base.
Input	None
Source	None
Output	Successfully display inputs from company's knowledge base.
Destination	Client's view

Action	The client clicks a menu item to view recorded information from a company's knowledge base table. These are grouped based on categories of services the company has.
Pre-condition	The company must have some entries in their knowledge base database.
Post-condition	None

2.6.2 Company Personnel

This section gives a detailed breakdown of the company agent's requirements for this system

2.6.2.1 Description and Priority

This feature covers the requirements as per a company agent or support team personnel. It is of high priority as it goes to complement the requirements of the company's customer. Here, management of tickets will be focused on.

2.6.2.2 Functional Requirements

Table 2.6 Agent requirement to view all submitted tickets

REQ-CA-1: An agent should be able to view a list of all submitted tickets.

Description	This requirement allows a user to view details of a ticket submitted using a given ticket id.
Input	None
Source	None
Output	A list of all tickets that have been submitted. An alert to show that tickets are unavailable should that be the case.
Destination	Agent's view
Action	An agent is permitted to view a list of all ticket on start of the web application
Pre-condition	There must be a number of tickets already available in the system
Post-condition	None

Table 2.7 Agent requirement to view ticket details

REQ-CA-2: An agent should be able to view details of a ticket.

Description	This requirement allows a user to view details of a ticket submitted using a given ticket id.
Input	Click on summarized ticket details in the display all ticket view.
Source	Agent
Output	An agent is given access to all the details of a submitted ticket. This is subject to the selected configuration of a company. This may include Sender's name, email, phone number, subject, complaint, file, location, status (in progress, closed), agent assigned, time, date, interactions. This makes up the ticket details page.
Destination	Agent's view
Action	An agent clicks a row from the ticket table which has the summarized details of a ticket. This opens a ticket details page with all the controls to change the status of a ticket
Pre-condition	The ticket being queried must exist i.e. the ticket id must be valid.
Post-condition	None

Table 2.8 Agent requirement to sort tickets

REQ-CA-3: An agent should be able to sort tickets

Description	This requirement allows an agent to sort tickets by the following parameters: date, time, deadline, priority (emergency, high, normal, low), assignment, archived, department
Input	Based on parameter required, click of a menu item allows agent to sort
Source	Agent
Output	A list of tickets that satisfy the parameter clicked on. If there isn't any output for the selected parameter, display a message showing agent that the selected section is empty.
Destination	Agent's view
Action	An agent has to select a desired menu item by what parameter he wants the tickets to be sorted
Pre-condition	Tickets must exist
Post-condition	Display of a list of tickets that satisfy the parameter selected.

Table 2.9 Agent requirement to change ticket status

REQ-CA-4: An agent should be able to change ticket status

Description	An agent should be able to change the status of a ticket that he is working on. Status changes may include the progress level of the ticket, archived or not
Input	Click on a ticket item
Source	Agent
Output	State changes of a ticket

Destination	Agent's view
Action	While viewing a given ticket, an agent can change its progress state from in progress to closed by selecting from a radio input. Tickets can be archived by selecting the archive option from the display
Pre-condition	Tickets must exist.
Post-condition	The state change must be recorded against that ticket id.

Table 2.10 Agent requirement to search for a ticket

REQ-CA-5 An agent should be able to search for a ticket

Description	An agent should be able to search for a ticket by id or keywords
Input	Search text
Source	Agent.
Output	A list of rows that satisfy the search item. An alert indicating the success or failure of the search.
Destination	An alert in the agent's view.
Action	Type the search text into a search box and click go to execute the query.
Pre-condition	A search box must exist.
Post-condition	Display of all the tickets that meet the requirements of the search item

Table 2.11 Agent requirement to create ticket

REQ-CA-6 An agent should be able to create ticket

Description	This requirement of the user enables them to issue a ticket on behalf of a client who calls the support center
Input	(Configurable by company's preference). Sender's name, email, phone number, subject, complaint, file, location, category, department
Source	A form filled by the agent.
Output	Alert user to enter valid data should wrong inputs be entered. Success or failure of submission. On success, an auto-reply message to sender. Notification.
Destination	An alert in the agent's view; A notification to the Company agent.
Action	To complete this action, a client must fill a form with the fields given as the inputs above. Click on a button to submit. An alert is then sent to the company agent's view.
Pre-condition	A client must have a problem/concern/challenge with the use of a service or product and must have phoned in.
Post-condition	The submitted ticket is made accessible to company agents.

Table 2.12 Agent requirement to respond to tickets

REQ-CA-7 An agent should be able to reply to a ticket.

Description	This requirement allows an agent to respond to a ticket.
Input	A message (response)
Source	On the ticket details page, the message (response) to be sent is typed and submitted.
Output	Successful reply must be displayed on the screen as an interaction with a timestamp. Auto-reply of this message is sent to the client via email or SMS based on the company's configuration.
Destination	Agent's view
Action	The agent may type in a message into a text field and click a button in order to add to the ongoing conversation concerning the resolution of a ticket. In the same process, an SMS or email is sent to the client as a response.
Pre-condition	The ticket must already be created.
Post-condition	View the latest added response to a ticket's conversation

Table 2.13 Agent's requirement to view conversation on a ticket

REQ-CA-8 An agent should be able to view conversation threads of a ticket.

Description	This requirement allows an agent to view ongoing conversations on a ticket as part of ticket details.
Input	Click on ticket in table row
Source	Agent
Output	Display interactions (conversations) that has gone on concerning a ticket i.e. message with timestamp. Where there are no interactions, there will be nothing to view.
Destination	Agent's view
Action	The agent may view/read previous conversations
Pre-condition	The ticket must already be created with existing conversation.
Post-condition	View responses to a ticket's conversation

Table 2.14 Agent's requirement to reassign tickets

REQ-CA-9 An agent should be able to (re)assign tickets

Description	This requirement enables an agent to reassign a ticket to a colleague.
Input	Selection of new agent's name.
Source	Agent
Output	An alert to show if a ticket has been successfully or unsuccessfully reassigned to a different agent.
Destination	An alert in the Client's view; A notification to the Company agent

Action	To complete this action, an agent must select the new assignee's name and append the ticket to it.
Pre-condition	There must be a ticket for reassignment. There must be a free/unoccupied agent.
Post-condition	The ticket is reassigned to the new agent.

2.6.3 Company Administrator

This section gives a detailed breakdown of the company administrator's requirements for this system

2.6.3.1 Description and Priority

This feature covers the requirement of a company administrator. Being in charge of setting up a company, requirements under this feature is of high priority since the whole system runs based on the configurations this administrator makes.

2.6.3.2 Functional Requirements

Table 2.15 Administrator's to register company

REQ-A-1: An administrator should be able to register a company

Description	This requirement allows an administrator to get a company started on the system. This includes creating an account for the company.
Input	Company name, username, password
Source	Administrator
Output	Access to Administrator's dashboard if sign up is successful. Alerts to refill fields that were otherwise not filled or filled wrongly
Destination	Administrator's view
Action	The administrator should fill out the fields presented with the required input, then submit to be processed
Pre-condition	The registering company must exist.
Post-condition	Access to administrators dashboard.

Table 2.16 Administrator's to configure company details

REQ-A-2: An administrator should be able to configure company details

Description	This requirement allows a company's administrator to successfully set up a company on the system. Areas needing configuration

	include: company details, ticket details (which will be seen and used by clients), support team details.
Input	Company details: Name, Logo, Theme colour, url, contact details, monitored system. Ticket details: Name, email, phone number, subject, complaint, file, location, status, priority, department, category, set timer for periodic notifications. Support team details: Adding users /team members (names, email, phone numbers, assigned departments) Department: Name, overview, group email Category: name, overview
Source	Administrator
Output	Alert on successful or failed attempt to configuration at each level. Alert to prompt administrator if a field is left unfilled or has an invalid input.
Destination	Administrator's view
Action	For company details, make the necessary changes to the fields provided from the input. For ticket configuration, select those fields that apply or better address the needs of the company For support team, department and category fill out the necessary fields provided from the input.
Pre-condition	The company must already be signed up.
Post-condition	Fully set up company i.e. stored details of the registered company to get it started.

Table 2.17 Administrator's to manage knowledge base

REQ-A-3: An administrator must be able to manage content of knowledge base

Description	This requirement allows an administrator to create and manage content that form the knowledge base for customer self help services.
Input	Add to, View and Edit knowledge base uses the following input: category type, question, solution.
Source	Administrator
Output	Alert to show a successful or failed attempt to add, edit or view content of a knowledge base.
Destination	Administrator's view
Action	The administrator can add to the knowledge base by filling a form with the above inputs. An edit involves displaying the already existing input and giving permission to edit. A list of all the content made available can also be viewed by clicking a button to view.
Pre-condition	Content must exist
Post-condition	Permission to add, view and edit content of knowledge base

Table 2.18 Administrator's to manage support team

REQ-A-4 An administrator should be able to manage users on the support team.

Description	This allows and administrator to add, edit details of, or delete team members from the support team
Input	Name, phone number, email, department, username, password of team members.
Source	Administrator
Output	Alert to show a successful or failed attempt to add a user or edit content of a user.
Destination	Administrator's view
Action	Administrator adds team members by filling out a form with the inputs above. Editing team member's details involves displaying the already existing information in a form and permitting the administrator to edit. He deletes a team member by the click of a button. Viewing available team members displays a list of members on the support team.
Pre-condition	Team members must be have been added in order for the administrator to edit, view or delete
Post-condition	Information about team members will be created and available for edits.

2.6.4 System Requirement

2.6.4.1 Description and Priority

Requirements of the system user class is of medium priority. These requirements cater for the smart and proactive aspect of the software. Features that make up this requirement facilitate data mining and analysis, report generation, prediction, auto-reply feature and the sending of notifications to involved parties.

2.6.4.2 FUNCTIONAL REQUIEMENTS

Table 2.19 System's requirement to send notifications

Req-S-1: The system should be able to send out notifications to personnel and clients.

Description	This requirement of the system enables email or SMS notifications company personnel assigned to a ticket or a department if the ticket is new or overdue.

Input	Name of recipient, phone number, personal email, department email, message, ticket id
Source	System
Output	Alert to show a successful or failed attempt to send a notification (SMS or email). Submission of notification.
Destination	Company agent’s device
Action	The system receives a trigger after the configured time set by the administration elapses. The system sends the message with the id of the overdue ticket to the department in charge of that ticket.
Pre-condition	There must be an existing recipient with valid contact information.
Post-condition	The recipient personnel must have received the notification

Table 2.20 System's requirement to auto-reply

Req-S-2: The system should be able to send auto-reply messages to clients

Description	This requirement of the system enables email or SMS notifications to clients after the submission of a ticket.
Input	Name, phone number, email of recipient, message, ticket id
Source	System
Output	Alert to show a successful or failed attempt to send a notification (SMS or email). Submission of notification (email or SMS).
Destination	Client device
Action	Once the system receives a new entry of a ticket, it generates a ticket id. The ticket id is appended to the message indicating that the problem will be addressed soon. The message is sent to the client using either the submitted email address or phone number.
Pre-condition	There must be an existing recipient with valid contact information.
Post-condition	The recipient (client) must have received the notification

Table 2.21 System's requirement to conduct sentiment analysis

Req-S-3: The system should be able to conduct sentiment analysis

Description	This requirement to conduct sentiment analysis using twitter
Input	Brand name, social media account name
Source	Administrator
Output	Results received positive, negative, neutral sentiments.
Destination	Database, administrator’s view

Action	Click button to display results of analysis grouped by category
Pre-condition	The company must have a social media account and brand name
Post-condition	Display results of analysis

Table 2.22 System's requirement monitor company systems

Req-S-4: The system must be able to monitor company systems

Description	This requirement monitors and consistently probes connected company systems.
Input	Systems to be monitored
Source	Administrator
Output	Log of probing results
Destination	Database, administrator's view
Action	Click button to view logs from probing
Pre-condition	The company must have a social media account
Post-condition	Create ticket when there is a fault.

Table 2.23 System's requirement generate report

Req-S-5: The system must be able to generate a comprehensive report on the activities of the company

Description	This requirement to conduct sentiment analysis using twitter
Input	Previous tickets in the database
Source	System
Output	Graphs of result of analysis
Destination	Administrator's view
Action	Click button to generate and show results of analysis
Pre-condition	The company must have a social media account
Post-condition	Display of graphs generated

Table 2.24 System's requirement to make predictions

Req-S-6: The system must be able to find patterns and make predictions

Description	This requirement to conduct sentiment analysis using twitter
Input	Tickets from database
Source	System
Output	Predicted value of error
Destination	Amazon Web Service, Administrator's view
Action	Click button to analyze and view predictions

Pre-condition	The company must have a tickets available
Post-condition	Summary of prediction points.

2.7 External Interface Requirement

2.7.1 User Interface

To a large extent, the user interface of this system will give users a wide working space for operations. The client's screen will have a fixed menu bar at the top owing to the fact that they will not have a lot of options to deal with.

The agent and administrator view will both have a simple sidebar menu navigation with very descriptive titles and icons for each menu. These group of users will have a large content pane where all interactions with the user will occur.

All buttons will be visible with a clear description of their functions to clearly define their use. All input texts will alert the user when a wrong or invalid text is entered.

2.7.2 Software Interface

Software interfaces needed for the smooth running of this application. This system connects via RESTful APIs to the following services Nagios Core service, Amazon Machine Learning web service, Mailgun emailing service, SMSGH and the Datumbox Machine Learning Framework.

Nagios Core is an IT infrastructure monitoring which serves as a basic event scheduler, event processor and alert manager for monitored company systems. Output from this service which report the state of a monitored system will be stored in the database as strings.

Amazon Machine Learning receives input from the database in CSV format in order to find patterns to show predictions. It returns its results in JSON format into the system.

Datumbox Machine Learning framework connects to social media, i.e. Twitter fetch data, analyze the feed and return a JSON formatted result. The result shows the sentiments retrieved after the analysis.

SMS notification API from SMSGH connects and authenticates the system. The system sends messages as well as the recipient's phone number to this service as a string which is forwarded to the mobile phones. This service will only be used in the event that the recipient does not have access to email services.

Email service from Mailgun receives input from the system in an array and returns an HTTP GET method with the HTTP response and body of the email. This notification sent to the client and personnel assures the former that his submitted ticket is being worked on and alerts the latter of a new ticket submission.

2.7.3 Communication Interface

This system, being a web-based application will be created using PHP for server side transactions. This aids in the multiple calls to the database adds, retrieves and update tickets and company configuration information. The web server and client will make use of the HTTP POST, PUT and GET Request methods to interact with each other for data transmission purposes.

2.8 Non Functional Requirement

- **Security**: The system must make sure that all data sent and received cannot be intercepted. The system should allow for authentication of company administrators and personnel so that all changes made can be traced to an authorized figure.
- **Availability**: The system should always be up and running with no downtime. It must be accessible at anytime during the day to deliver defined services to users. The system should be lightweight in order to support a wide range of bandwidth.
- **Reliability**: The system should be trustworthy. It should be able to work as defined by this document.
- **Performance**: Interactions with this system must always be fast with little to no delay in processes. The response time of every action prompt. To boost performance there is very little communication between components. Queries made to the database are not complex, thus, use very little processing power and time to execute which enhances performance. Changing of screens as a client uses the system will be fast and accurate because there is little computation involved.
- **Software Quality Attributes**: The graphical user interface of this software is designed with a high level of usability as priority owing to the fact that the system will be used by clients of different age ranges. Menus will be well organized to enable a user correctly navigate the system with little room for errors. There will be constant feedback/notification after every action performed to guide user through the use of the system.

Chapter 3 : Architecture and Design

3.1 Overview

This software design document contains the system overview of this trouble ticketing software, system architecture, component design, Data design, Human interface design and the traceability requirement matrix

3.2 System Overview and Architecture

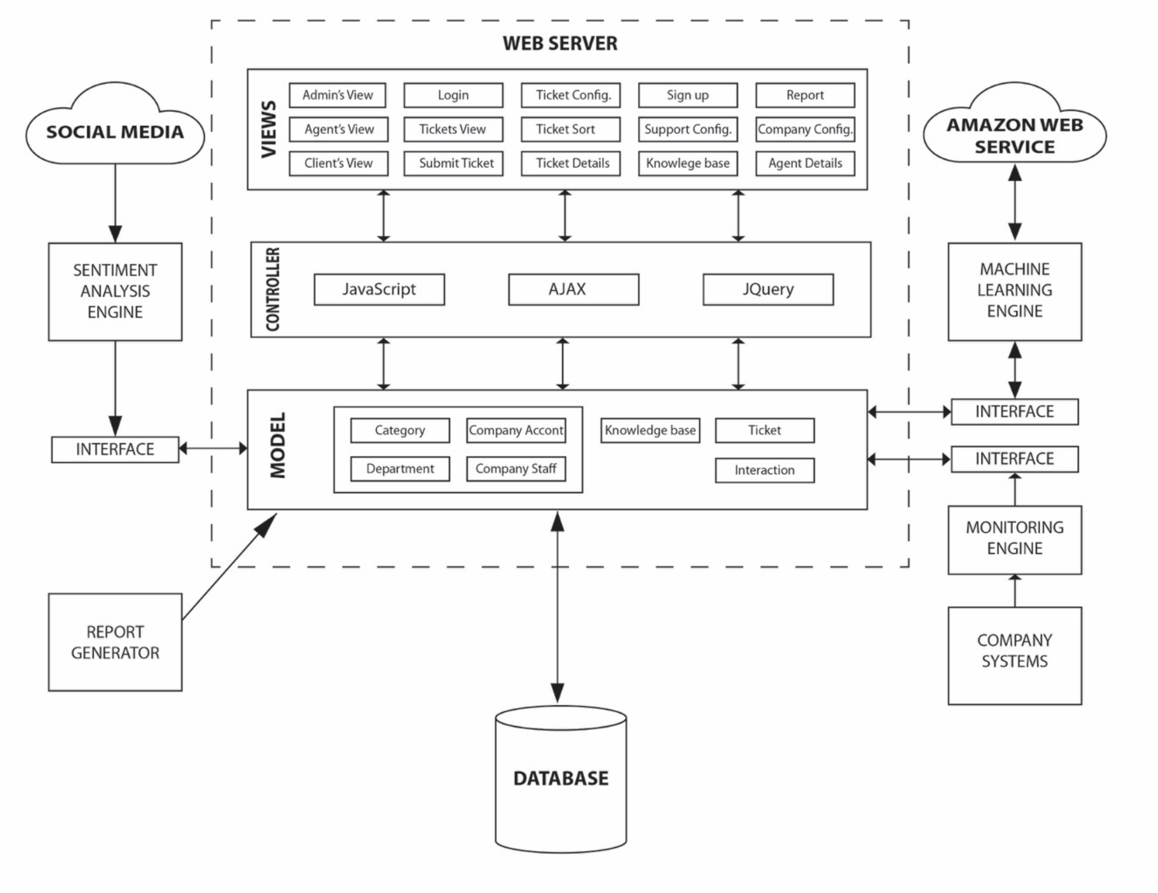


Figure 3.1 System overview of the trouble ticketing system

The architectural design of this system employs a combination of the Model-View-Controller and Repository pattern.

- MVC forms the basis of interaction management of the web-based section of this system. The system is structured into logical components that interact with each other. The Model component manages the system data and associated operations on that data. It serves as the interface between the database and controller as well as other components connected to the system. The View component defines and manages how the data retrieved from the database is presented to the various users. The Controller component manages user interaction by passing these interactions from the various Views to the Model and vice versa. This pattern was used because there are multiple ways to view and interact with data. It also supports presentation of the same data in different ways with changes made in one representation shown in all of them.

- The second part of this architecture looks at the system as having a repository pattern. This is because system makes use of large amount of data made accessible through a central repository. Data is fed into the system (shared database) through three main components – users, social media and company systems which is made accessible to the other components of the system – report generator and machine learning engine.

3.3 Data Design

Data will solely reside on the server side of the system and will be organized based on classes. These classes manage the following: tickets, company details, knowledge base, sign up, configurations, interactions, analysis. Owing to the fact that this software is heavily dependent on data, the classes that handle the data will be isolated and will be accessed through the MVC model as discussed above. These classes that form the interface facilitate client-server interactions will be implemented using PHP and a permanent storage for the data received will be a MySQL database. Data flow from the server to client side will will be by the JSON format since it is a lightweight data-interchange format.

3.3.1 Database Description

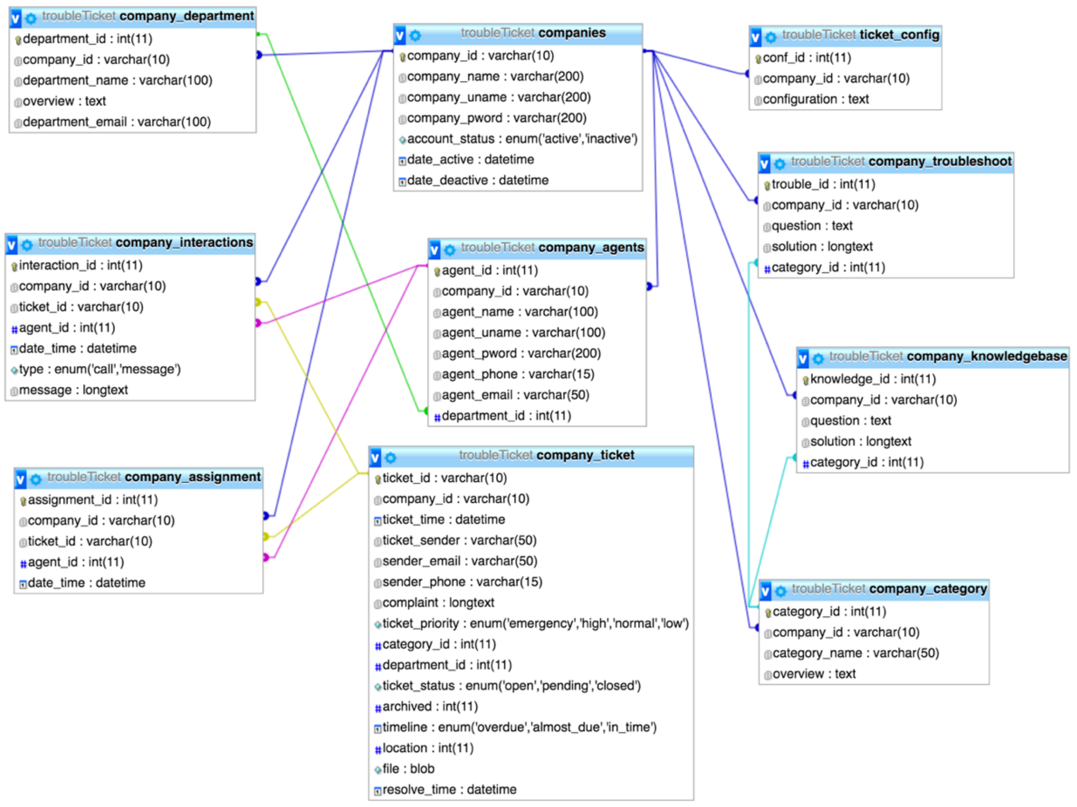


Figure 3.2 Database architecture for the system

3.4 Activity Diagram

Figure 3.3 shows the workflow of a new administrator. He starts off by registering the company to allow him access into the system. When this is successful, he is required to configure the system as per the preferences of the organization. Under system configuration, he manages the support team members and their assignment into various departments, he selects relevant fields to make up the organization's ticket and he also manages the content of the knowledge base.

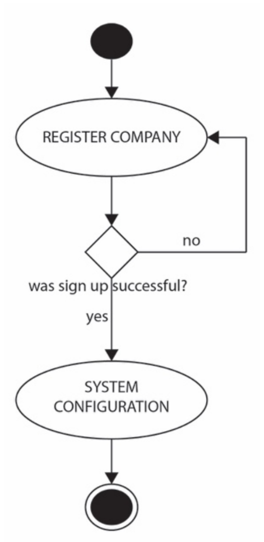


Figure 3.3 Activity diagram showing administrator's workflow

Figure 3.4 on the other hand shows the interaction that goes on between a client and an agent. This interaction deals with ticket generation and management. First, a client facing an issue searches the organizations knowledge base to find a solution to his problem, with no suitable results found, he issues a ticket. A company agent reviews the ticket and gets back the client for further clarification when necessary. The agent continues working the issue raised in the ticket till it is solved. Once solved, he closes the ticket.

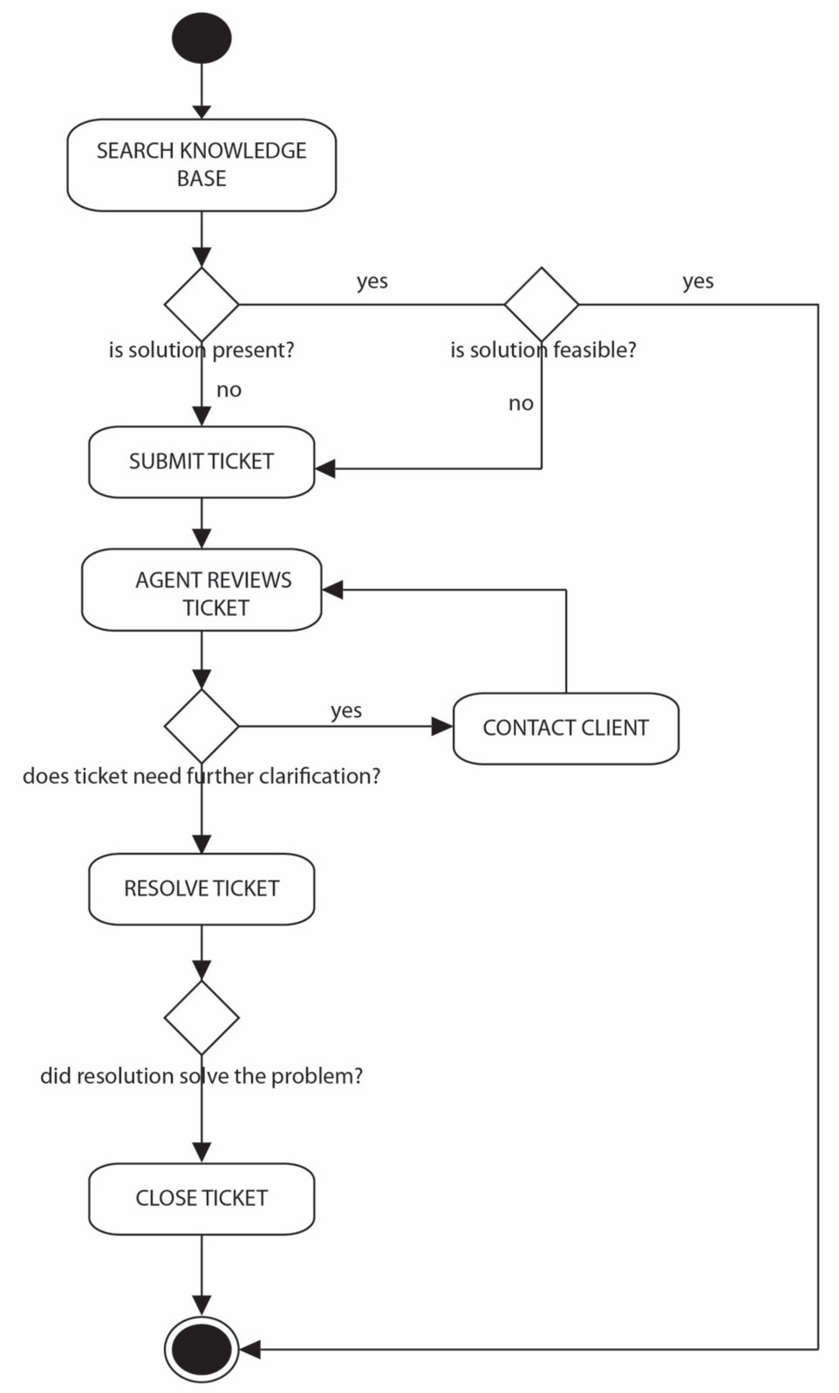


Figure 3.4 Activity diagram showing client and agent workflow

3.5 Human Interface Design

The agent's dashboard (figure 3.5) is the operating field of the signed in company agent. He is able to view all tickets that are assigned to the department he belongs to. From here, he can manage tickets, search through, archive tickets, sort tickets, interact with clients who issued tickets and finally resolve these tickets.

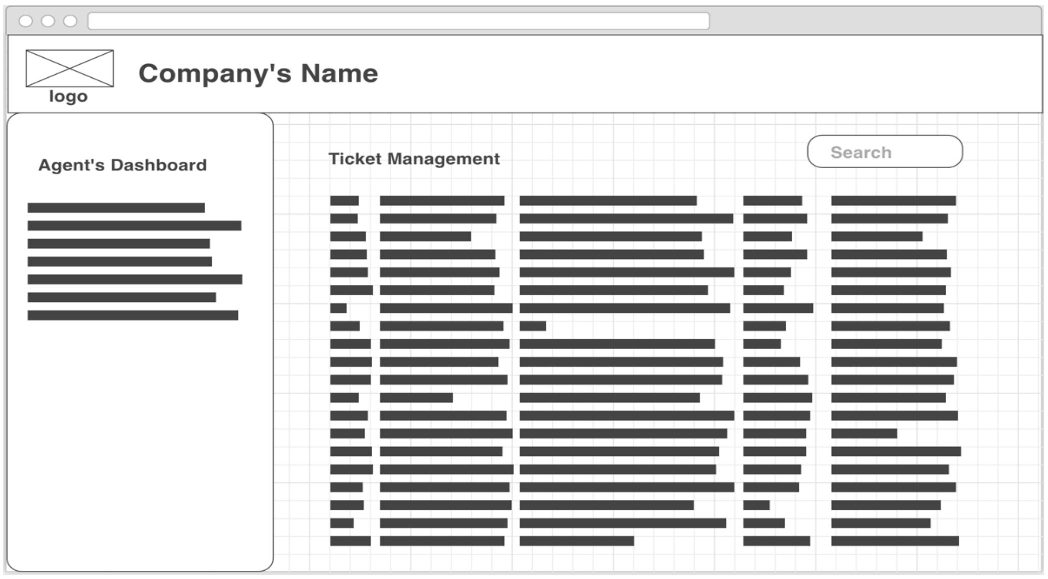


Figure 3.5 User interface representing the agent's dashboard.

The administrator's dashboard (figure 3.6) is the view that is presented to an administrator after setting up or logging into the system. From here, he has access to a menu that allows him to configure a company's ticket, set up company's account, set up the support team for the company, add and edit content of the knowledge base. Here, he is able to configure the system based on the organization's preferences. In addition to configuration, this view gives the administrator the chance to interact with all the reports and analysis that the performs. It provides quality feedback for decision making.

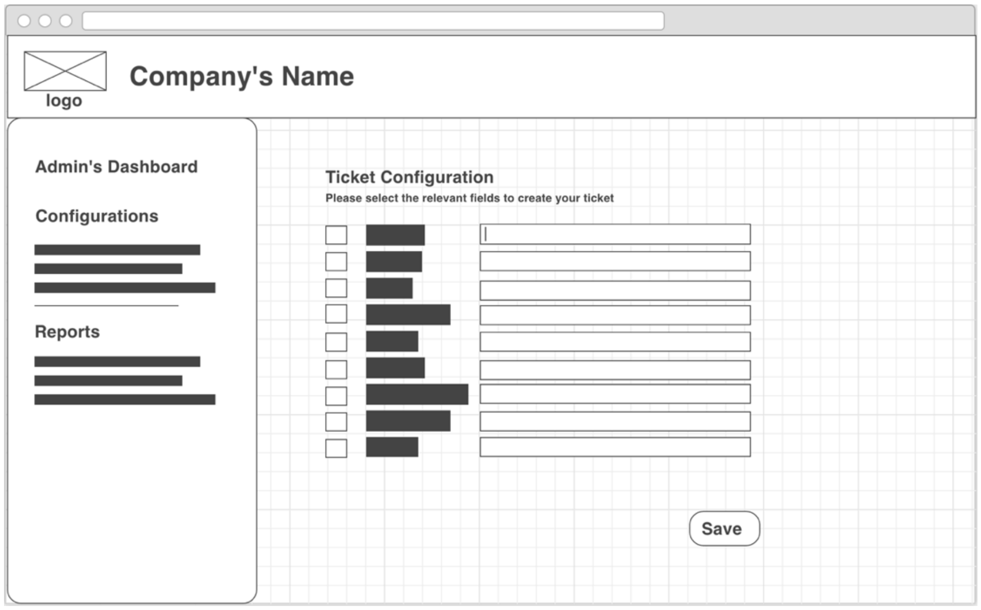


Figure 3.6 User interface representing the administrator's dashboard

Figure 3.7 shows the user interface for the company's client. This view allows a client to submit a ticket, view content in the knowledge base and track the progress of a ticket.

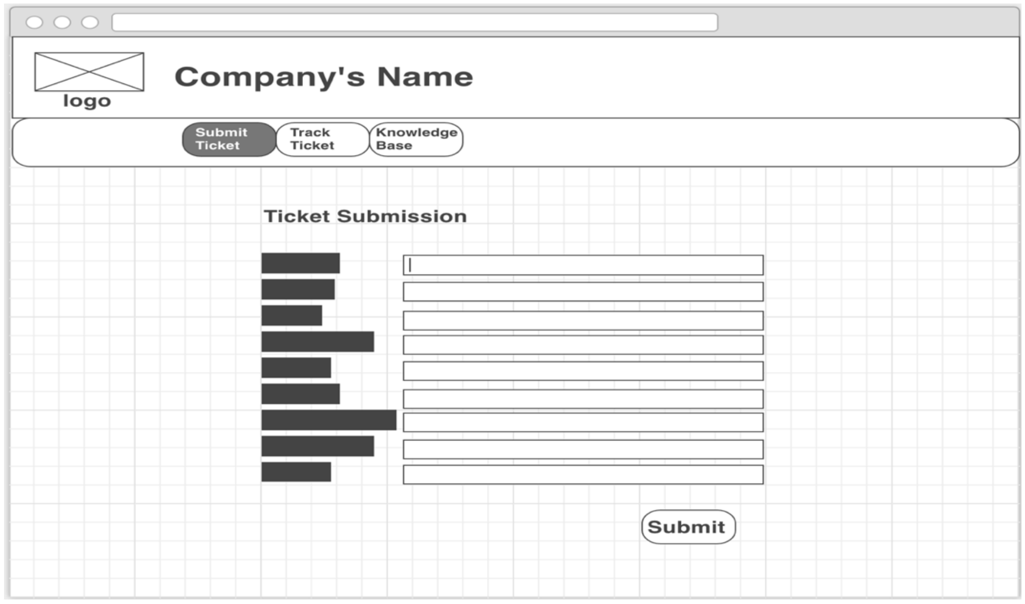


Figure 3.7 User interface representing the client's dashboard.

3.6 Sequence Diagram

This section shows the sequence diagrams that show the interactions of objects in the system. It shows how processes operate with one another and the outcome of various actions performed by the actors of the system.

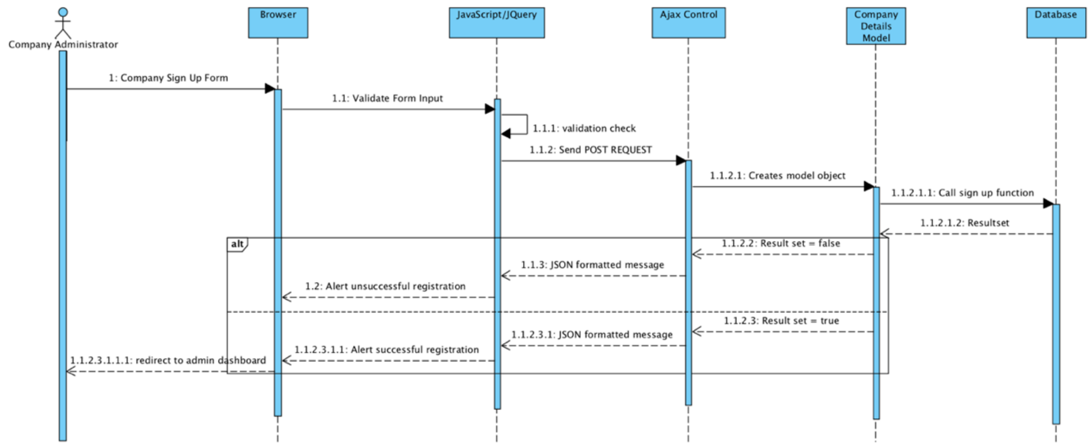

Company Administrator
Browser
JavaScript/JQuery
Ajax Control
Company Details Model
Database
1: Company Sign Up Form
1.1: Validate Form Input
1.1.1: validation check
1.1.2: Send POST REQUEST
1.1.2.1: Creates model object
1.1.2.1.1: Call sign up function
1.1.2.1.2: Resultset
alt
1.1.2.2: Result set = false
1.1.3: JSON formatted message
1.2: Alert unsuccessful registration
1.1.2.3: Result set = true
1.1.2.3.1: JSON formatted message
1.1.2.3.1.1: Alert successful registration
1.1.2.3.1.1.1: redirect to admin dashboard

Figure 3.8 Component a company administrator will step through to sign up a new company.

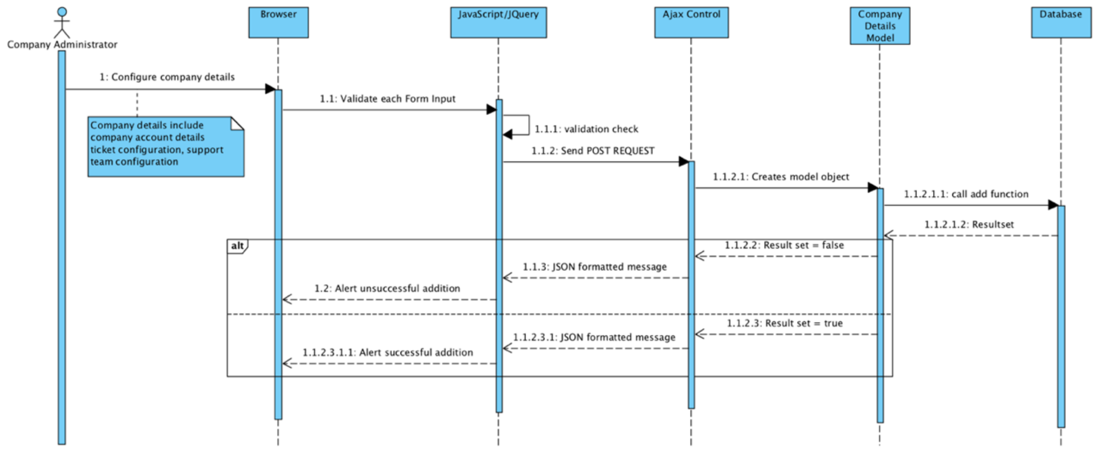

Company Administrator
Browser
JavaScript/JQuery
Ajax Control
Company Details Model
Database
1: Configure company details
Company details include company account details ticket configuration, support team configuration
1.1: Validate each Form Input
1.1.1: validation check
1.1.2: Send POST REQUEST
1.1.2.1: Creates model object
1.1.2.1.1: call add function
1.1.2.1.2: Resultset
alt
1.1.2.2: Result set = false
1.1.3: JSON formatted message
1.2: Alert unsuccessful addition
1.1.2.3: Result set = true
1.1.2.3.1: JSON formatted message
1.1.2.3.1.1: Alert successful addition

Figure 3.9 Steps each configuration goes through in order to exist.

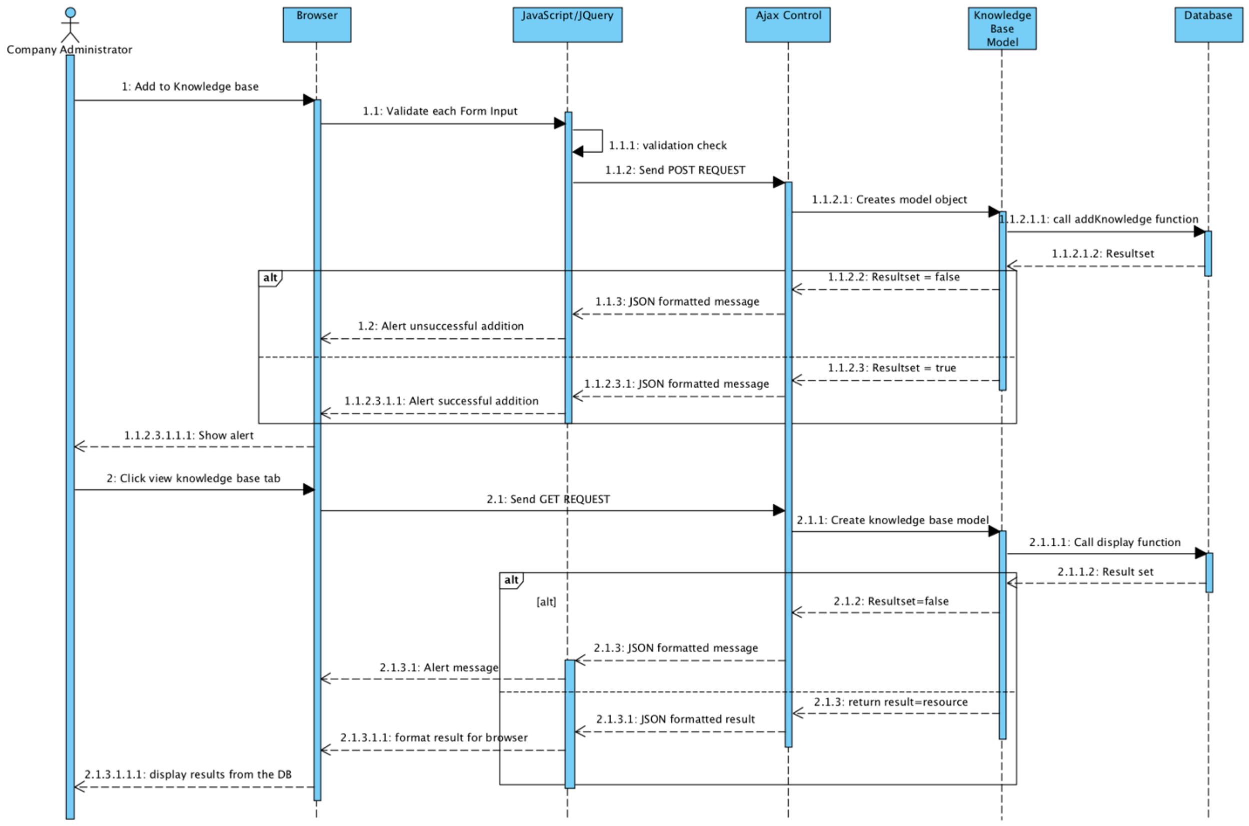

Company Administrator
Browser
JavaScript/JQuery
Ajax Control
Knowledge Base Model
Database
1: Add to Knowledge base
1.1: Validate each Form Input
1.1.1: validation check
1.1.2: Send POST REQUEST
1.1.2.1: Creates model object
1.1.2.1.1: call addKnowledge function
1.1.2.1.2: Resultset
alt
1.1.2.2: Resultset = false
1.1.3: JSON formatted message
1.2: Alert unsuccessful addition
1.1.2.3: Resultset = true
1.1.2.3.1: JSON formatted message
1.1.2.3.1.1: Alert successful addition
1.1.2.3.1.1.1: Show alert
2: Click view knowledge base tab
2.1: Send GET REQUEST
2.1.1: Create knowledge base model
2.1.1.1: Call display function
2.1.1.2: Result set
alt
[alt]
2.1.2: Resultset=false
2.1.3: JSON formatted message
2.1.3.1: Alert message
2.1.3: return result=resource
2.1.3.1: JSON formatted result
2.1.3.1.1: format result for browser
2.1.3.1.1.1: display results from the DB

Figure 3.10 Processes to add and view content of the knowledge base.

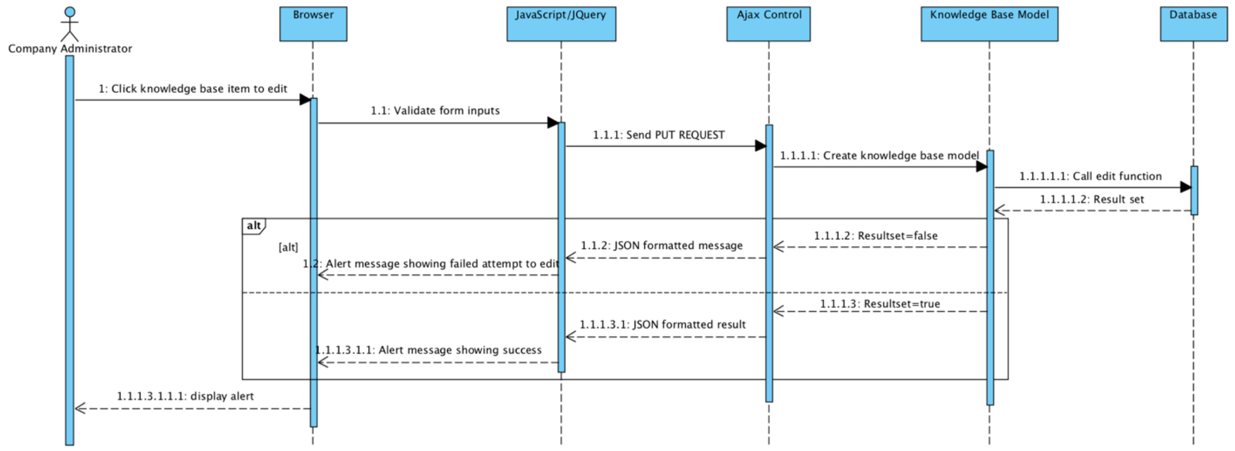

Company Administrator
Browser
JavaScript/JQuery
Ajax Control
Knowledge Base Model
Database
1: Click knowledge base item to edit
1.1: Validate form inputs
1.1.1: Send PUT REQUEST
1.1.1.1: Create knowledge base model
1.1.1.1.1: Call edit function
1.1.1.1.2: Result set
alt
[alt]
1.1.1.2: Resultset=false
1.1.2: JSON formatted message
1.2: Alert message showing failed attempt to edit
1.1.1.3: Resultset=true
1.1.1.3.1: JSON formatted result
1.1.1.3.1.1: Alert message showing success
1.1.1.3.1.1.1: display alert

Figure 3.11 Processes an edit goes through when issued by an administrator

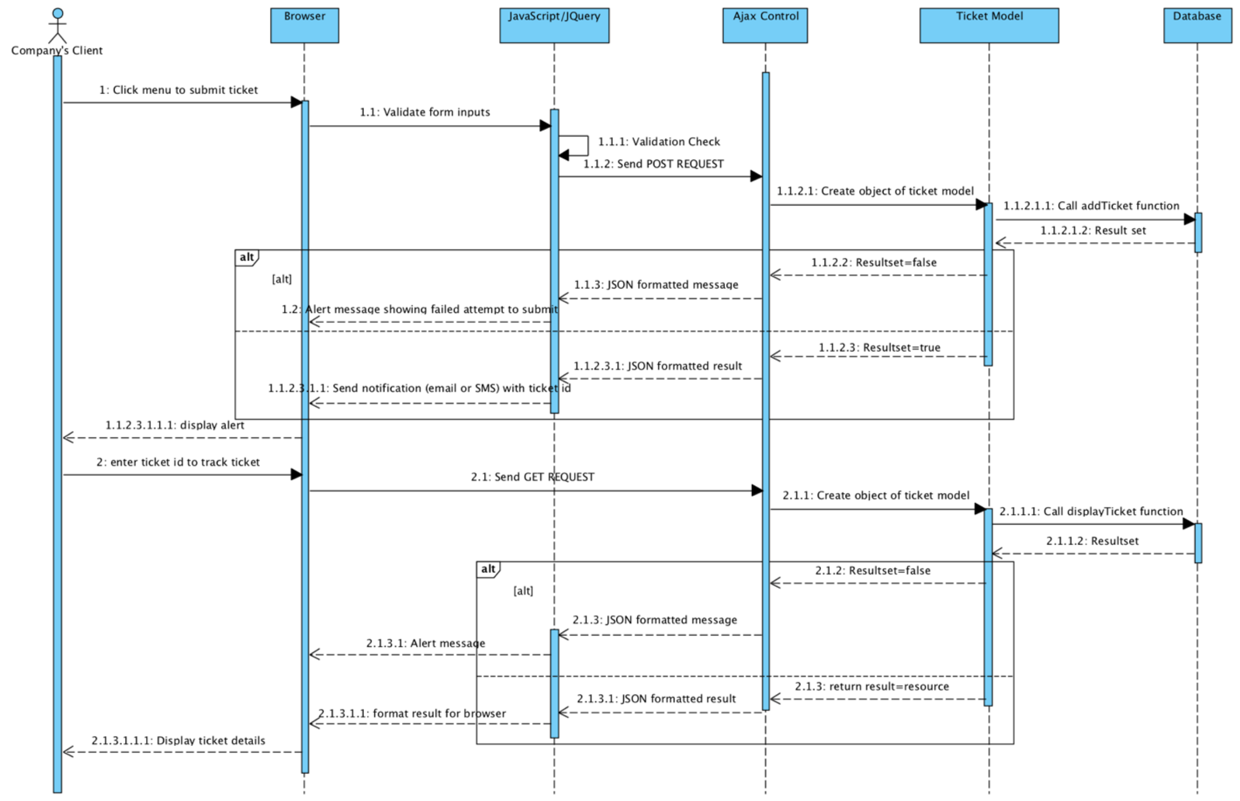

Company's Client
Browser
JavaScript/JQuery
Ajax Control
Ticket Model
Database
1: Click menu to submit ticket
1.1: Validate form inputs
1.1.1: Validation Check
1.1.2: Send POST REQUEST
1.1.2.1: Create object of ticket model
1.1.2.1.1: Call addTicket function
1.1.2.1.2: Result set
alt
[alt]
1.1.2.2: Resultset=false
1.1.3: JSON formatted message
1.2: Alert message showing failed attempt to submit
1.1.2.3: Resultset=true
1.1.2.3.1: JSON formatted result
1.1.2.3.1.1: Send notification (email or SMS) with ticket id
1.1.2.3.1.1.1: display alert
2: enter ticket id to track ticket
2.1: Send GET REQUEST
2.1.1: Create object of ticket model
2.1.1.1: Call displayTicket function
2.1.1.2: Resultset
alt
[alt]
2.1.2: Resultset=false
2.1.3: JSON formatted message
2.1.3.1: Alert message
2.1.3: return result=resource
2.1.3.1: JSON formatted result
2.1.3.1.1: format result for browser
2.1.3.1.1.1: Display ticket details

Figure 3.12 Interactions of components when a client submits a ticket and tracks the ticket with the given id.

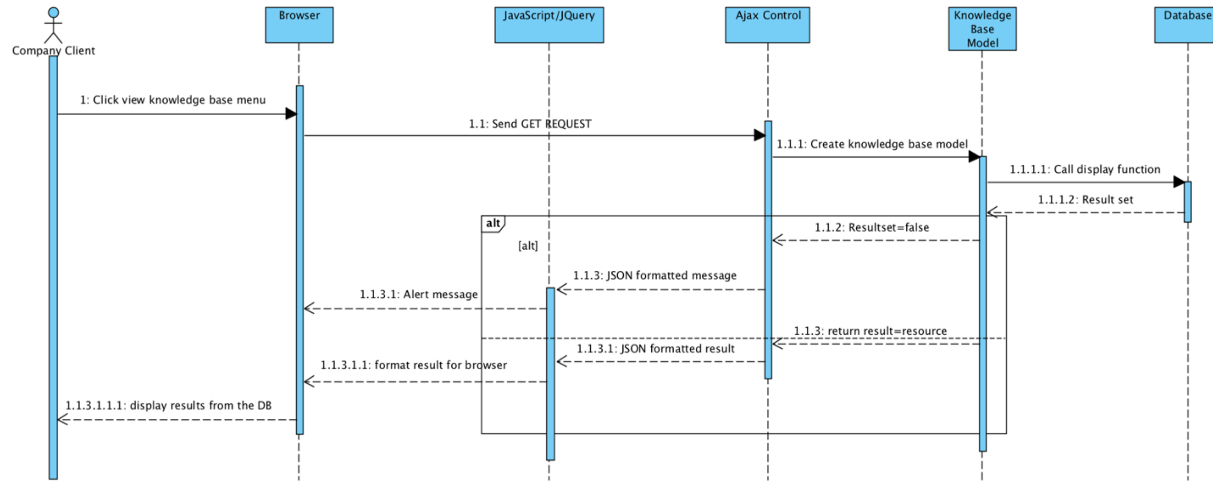
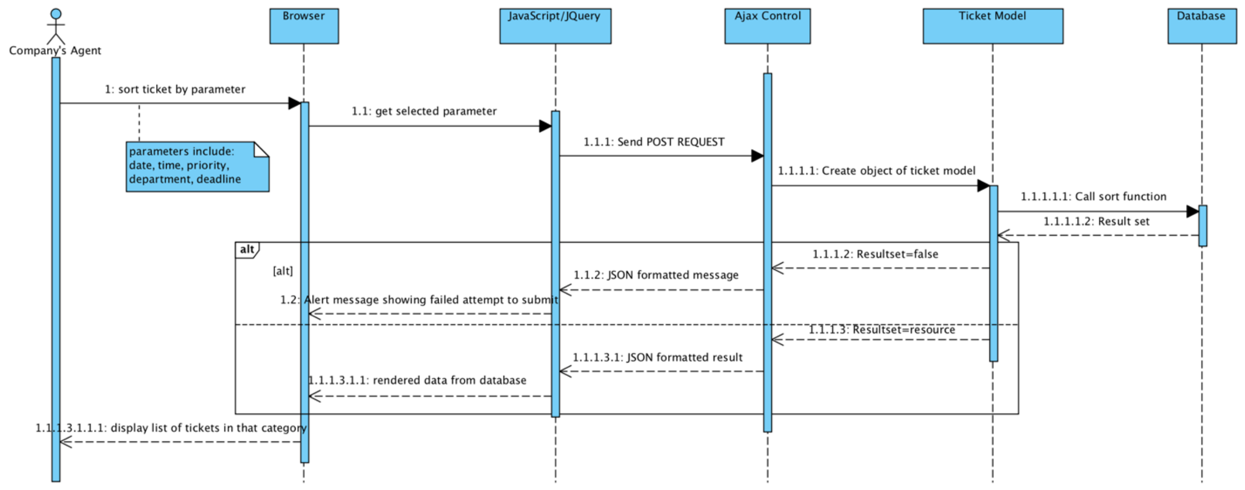
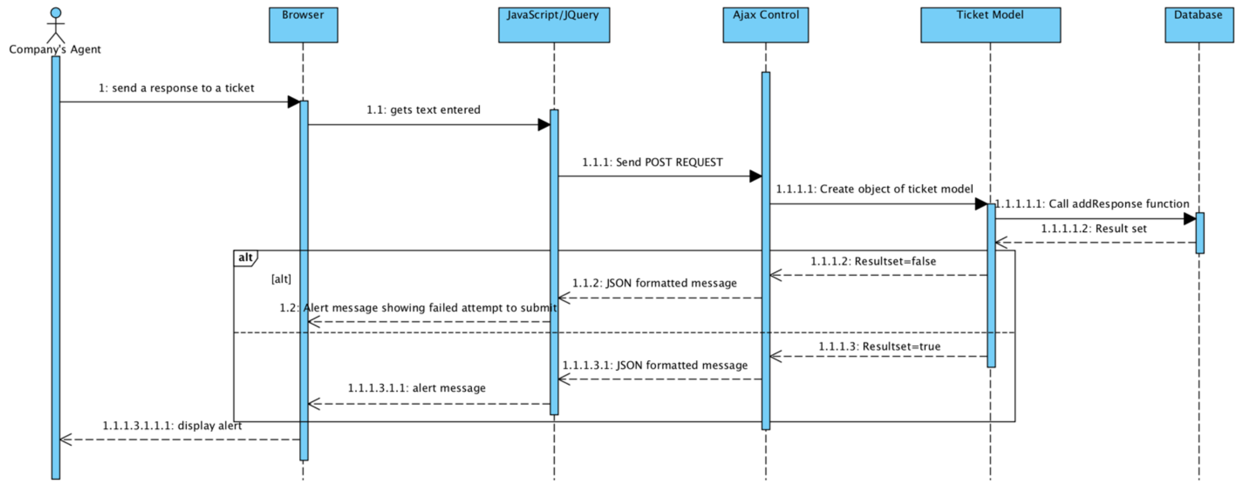

Company Client
Browser
JavaScript/JQuery
Ajax Control
Knowledge Base Model
Database
1: Click view knowledge base menu
1.1: Send GET REQUEST
1.1.1: Create knowledge base model
1.1.1.1: Call display function
1.1.1.2: Result set
alt
[alt]
1.1.2: Resultset=false
1.1.3: JSON formatted message
1.1.3.1: Alert message
1.1.3: return result=resource
1.1.3.1: JSON formatted result
1.1.3.1.1: format result for browser
1.1.3.1.1.1: display results from the DB

Figure 3.13 Interactions of components when a client views the knowledge base



Company's Agent
Browser
JavaScript/JQuery
Ajax Control
Ticket Model
Database
1: sort ticket by parameter
parameters include: date, time, priority, department, deadline
1.1: get selected parameter
1.1.1: Send POST REQUEST
1.1.1.1: Create object of ticket model
1.1.1.1.1: Call sort function
1.1.1.1.2: Result set
alt
[alt]
1.1.1.2: Resultset=false
1.1.2: JSON formatted message
1.2: Alert message showing failed attempt to submit
1.1.1.3: Resultset=resource
1.1.1.3.1: JSON formatted result
1.1.1.3.1.1: rendered data from database
1.1.1.3.1.1.1: display list of tickets in that category

Figure 3.14 Interactions of components when an agent sorts tickets by a given parameter.



Company's Agent
Browser
JavaScript/JQuery
Ajax Control
Ticket Model
Database
1: send a response to a ticket
1.1: gets text entered
1.1.1: Send POST REQUEST
1.1.1.1: Create object of ticket model
1.1.1.1.1: Call addResponse function
1.1.1.1.2: Result set
alt
[alt]
1.1.1.2: Resultset=false
1.1.2: JSON formatted message
1.2: Alert message showing failed attempt to submit
1.1.1.3: Resultset=true
1.1.1.3.1: JSON formatted message
1.1.1.3.1.1: alert message
1.1.1.3.1.1.1: display alert

Figure 3.15 Interactions of components when an agent sends a response to a ticket in the resolution process.

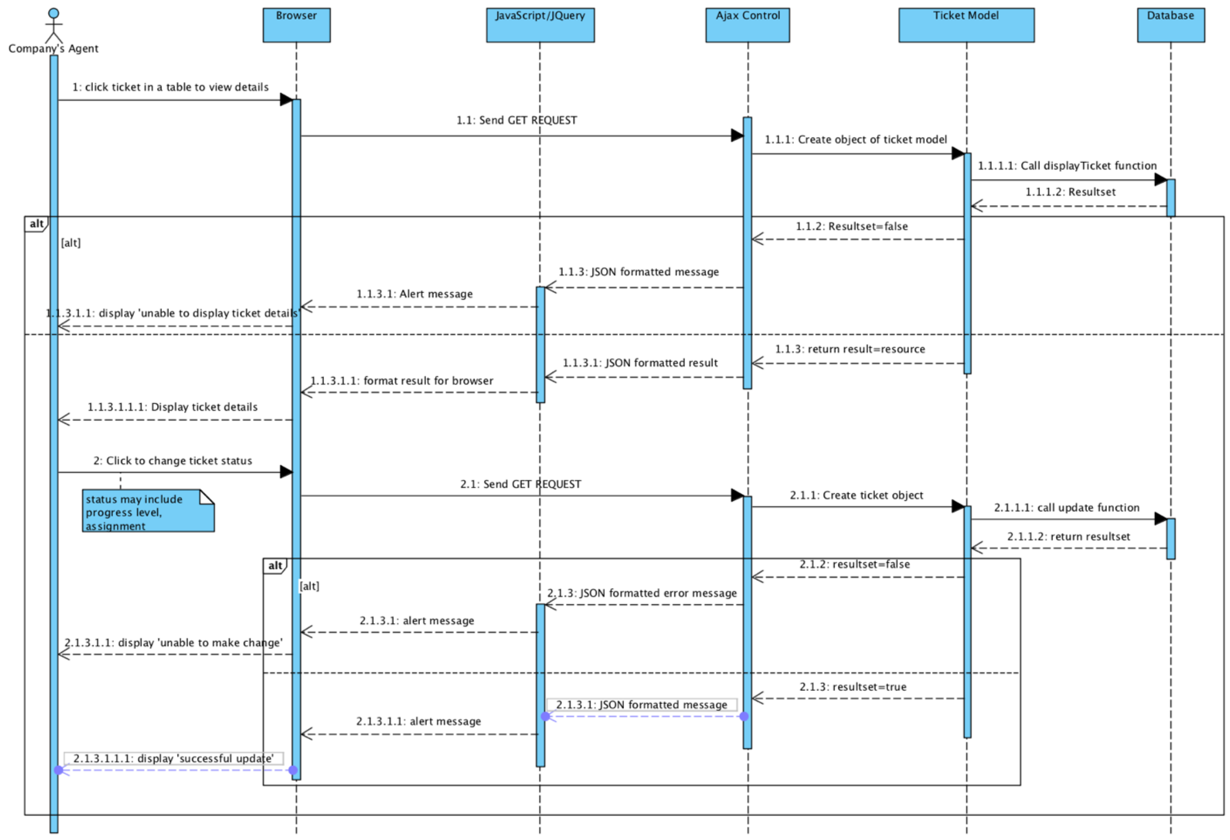

Company's Agent
Browser
JavaScript/JQuery
Ajax Control
Ticket Model
Database
1: click ticket in a table to view details
1.1: Send GET REQUEST
1.1.1: Create object of ticket model
1.1.1.1: Call displayTicket function
1.1.1.2: Resultset
alt
[alt]
1.1.2: Resultset=false
1.1.3: JSON formatted message
1.1.3.1: Alert message
1.1.3.1.1: display 'unable to display ticket details'
1.1.3: return result=resource
1.1.3.1: JSON formatted result
1.1.3.1.1: format result for browser
1.1.3.1.1.1: Display ticket details
2: Click to change ticket status
status may include progress level, assignment
2.1: Send GET REQUEST
2.1.1: Create ticket object
2.1.1.1: call update function
2.1.1.2: return resultset
alt
[alt]
2.1.2: resultset=false
2.1.3: JSON formatted error message
2.1.3.1: alert message
2.1.3.1.1: display 'unable to make change'
2.1.3: resultset=true
2.1.3.1: JSON formatted message
2.1.3.1.1: alert message
2.1.3.1.1.1: display 'successful update'

Figure 3.16 Interactions of components when an agent clicks to view a ticket details out of the list of tickets presented and moves on to resolve/change status of the ticket.

3.7 Traceability Requirement Matrix

This section maps components to requirements. This ensures that all requirements are covered in the design and implementation and validation of the service. Below is a legend of modules used in the traceability requirement matrix. REQ in this case is a reference to requirement identifications from chapter 2. The 'X' in the table shows the mapping of a requirement to a component.

1. System configuration
2. Ticket generation
3. Ticket management

4. Sentiment Analysis Engine
5. Monitoring Engine
6. Machine Learning Engine
7. Report generation

Table 3.1 Traceability requirement matrix

Requirement/Component	1	2	3	4	5	6	7
REQ-Cl-1	X	X					
REQ-Cl-2	X	X					
REQ-Cl-3			X				
REQ-Cl-4			X				
REQ-Cl-5	X	X					
REQ-CA-1			X				
REQ-CA-2			X				
REQ-CA-3			X				
REQ-CA-4			X				
REQ-CA-5			X				
REQ-CA-6	X	X					
REQ-CA-7			X				
REQ-CA-8			X				
REQ-CA-9			X				
REQ-A-1	X						
REQ-A-2	X						
REQ-A-3	X						
REQ-A-4	X						
REQ-S-1		X					
REQ-S-2		X					
REQ-S-3				X			
REQ-S-4					X		
REQ-S-5							X
REQ-S-6						X	

Chapter 4 : Implementation

4.1 Technologies Used

Tools

- PHP: PHP stands for Hypertext Preprocessor. It is a server side scripting language suited for web development. It is compatible with most servers and supports a wide range of databases. In this project, PHP handles all the server side and database interactions because it is fast, flexible and pragmatic
- JavaScript: It is an object-oriented programming language usually used to render and create interactive effects within a browser. Its use was employed in this project because it is lightweight. It also aids in the validation of inputs received from the browser.
- HTML: HyperText Markup Language is a standardized system for tagging test files to achieve font, colour, graphic, and hyperlink effects on web pages. It represents what the user interacts with in the web browser.

Libraries

- jQuery: It is a fast, small and feature-rich JavaScript library that makes it easier to use JavaScript on a website. It is designed to simplify the client side scripting of HTML. It is responsible for event handling and animations on the web pages.
- Chart.js is a highly responsive open source charting library that works very well on all device types. The range of charts use HTML5 canvas element for rendering. This project focused on the use of the Line Chart because it is easy to understand and paints a simple, clean and engaging chart for the company administrator.

- Mailgun-PHP: This RESTful API is built on HTTP provides this software with transactional email services. It uses built-in HTTP capabilities for passing parameters and authentication and returns output in JSON format.

Frameworks

- Bootstrap 3: It is a free and open source collection of tools for creating web applications. This HTML, CSS and JavaScript framework aids in building responsive mobile-first websites suited for all device types.
- Datumbox Machine Learning: It is an open source framework which allows rapid development of Machine Learning and Statistical applications. The web based API service uses REST operations with parameters encoded into the URL over HTTP POST requests into the and a JSON formatted response.

APIs

- SMSGH – MYtxtBox: This provides an online messaging service which engages users via text messages. This covers a range of users who may not have email services.
- Amazon Machine Learning Web Service: This service is a robust cloud based service that makes it easy to obtain predictions for an application without having to manage any infrastructure and using simple APIs.
- Nagios: This is a free and open source software that monitors systems and notifies users when things go wrong in these systems. It gives instant awareness of IT infrastructural problems in order to reduce impact of a downtime on an organization.

4.2 Description of Components

The relevant components for the full functioning of this system are described below. Sub-systems that make up the larger components are also discussed below.

- **Ticket Generation:** This component handles all processes involved with issuing and generating a ticket. In this component, there are sub-systems which deal with the following:
    - Notification Service: This sub-system deals with alerting company clients and personnel when the need arises. These users will be notified via email or SMS based on the configuration of the organization. Clients will receive notifications after the submission of a ticket with the generated ticket's id and ticket summary. They will receive notifications when an agent sends a message for clarity through the portal. Personnel in a particular department will be notified when a new ticket is submitted by a client. When a submitted ticket is unattended for an amount of time, another notification is sent to the personnel in charge until the ticket is resolved.
    - Ticket Submission Service: This sub-system allows a client or agent to submit a ticket. It generates a ticket form based on the organization's configuration. This provides the relevant fields a client has to fill in order to report the issue.
- **Ticket Management:** This component caters for all the support services this software can offer with regards to ticket management. This is where the company agents operate. Sub-systems which make up this component include:
    - Ticket Tracking Service: This sub-system enables a client to follow up on

the progress of an issued ticket with a given id. The only requirement for this service is a valid ticket id which is provided by the system and sent as part of ticket summary to the client from the notification service. This service gives access to the details of the specified ticket.

- Ticket Resolution Service: This sub-system serves the needs of an agent in resolving a ticket. This connects with and receives input from the ticket submission sub-system. An agent gets the details of a ticket from this sub-system in order to resolve it. Here, the agent is permitted to change the status of a ticket from new to in progress or closed. He is allowed give feedback or solicit for clarity from the client who issued the ticket. In the resolution process, this system allows the agent to reassign the ticket to another person who is more fit for the issue at hand.

- **System Configuration:** This component handles the configuration information of an organization. Sub-systems in here allow an administrator to manage his support team members, company account details and the knowledge base content.
- **Machine Learning Engine:** This component makes use of the Amazon Machine Learning Web Service to train a model with a given data source, evaluate model and make predictions. For this software a categorical attribute attribute was selected as target (what should be predicted). The Machine Learning models trained on this target use multinomial logistic regression to train a multiclass classification model.
- **Report Generation:** This component deals with the generation of reports based on the number of tickets available. It represents data using a Line Chart for the purpose of comparison and depicting trends.

- **Sentiment Analysis Engine:** This component employs the sentiment analysis function from Datumbox. This function classifies comments from social media as positive, negative or neutral and feed results into the database.
- **System Monitoring:** This component works together with the Nagios platform to monitor connected systems in order to minimize downtimes of an organization's system.

4.3 Implementation Techniques

During implementation, the whole software system was broken down into smaller sub-units based on characteristics. This was done for easy management of the program. Each component, as listed in section 4.2, was further decomposed to into smaller sub-systems. The composition of these smaller components makes up this trouble ticketing system.

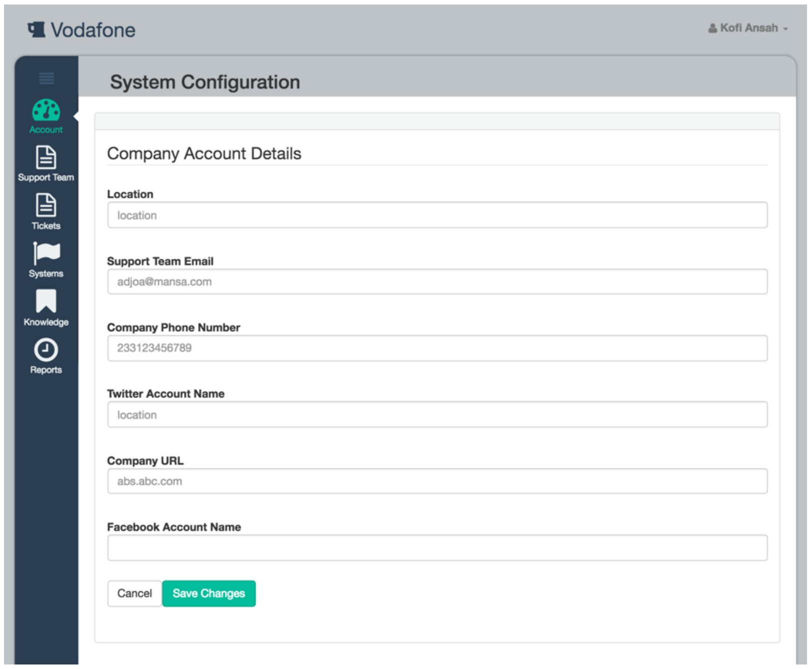
The ticket generation component interacts with the ticket management component. Input from the ticket generation component comes from users of the system – client and or personnel. The output of this component (generated ticket) serves as the input for the ticket management component. After tickets are resolved, output is sent into the database. The machine learning component takes input retrieved from the database in order to make predictions; the sentiment analysis component receives input from social media outlets whereas the system monitoring components take input from the connected company systems. All these components store their outputs into the database which serves as an input for the report generation component. The report generation component takes input from the database, analyzes it and displays it as graphs and charts for the administrator of the system.

4.4 Evidence of Implementation

This section captures screenshots of the implemented prototype.

4.4.1 Administrator's view

Figures 4.1 to 4.3 give an administrator the liberty to configure this system to the preference of the company. The administrator is at liberty to entire valid company details relevant for the uninterrupted performance of the system.



System Configuration

Company Account Details

Location: location

Support Team Email: adjoa@mansa.com

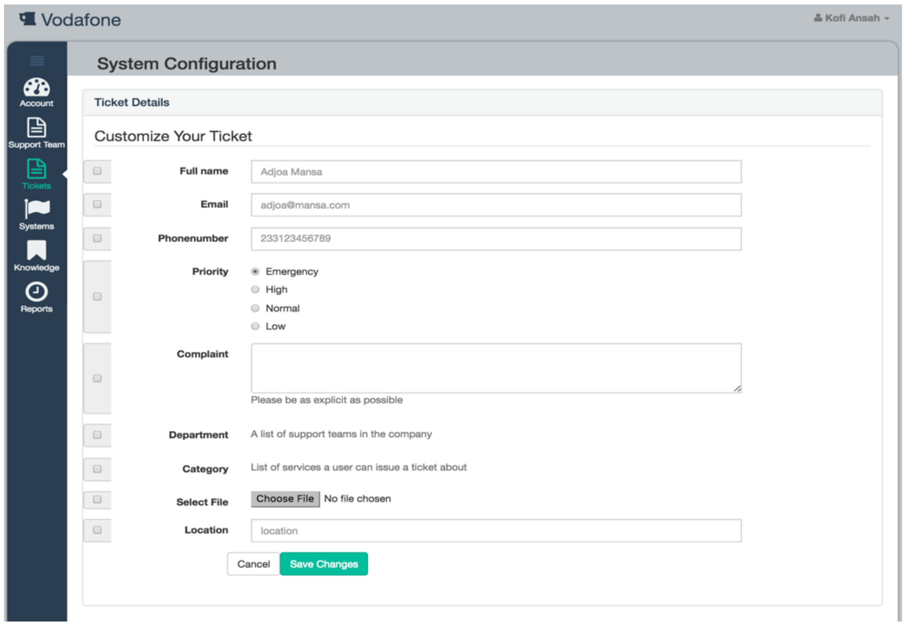
Company Phone Number: 233123456789

Twitter Account Name: location

Company URL: abs.abc.com

Facebook Account Name:

Figure 4.1 Interface to store company details



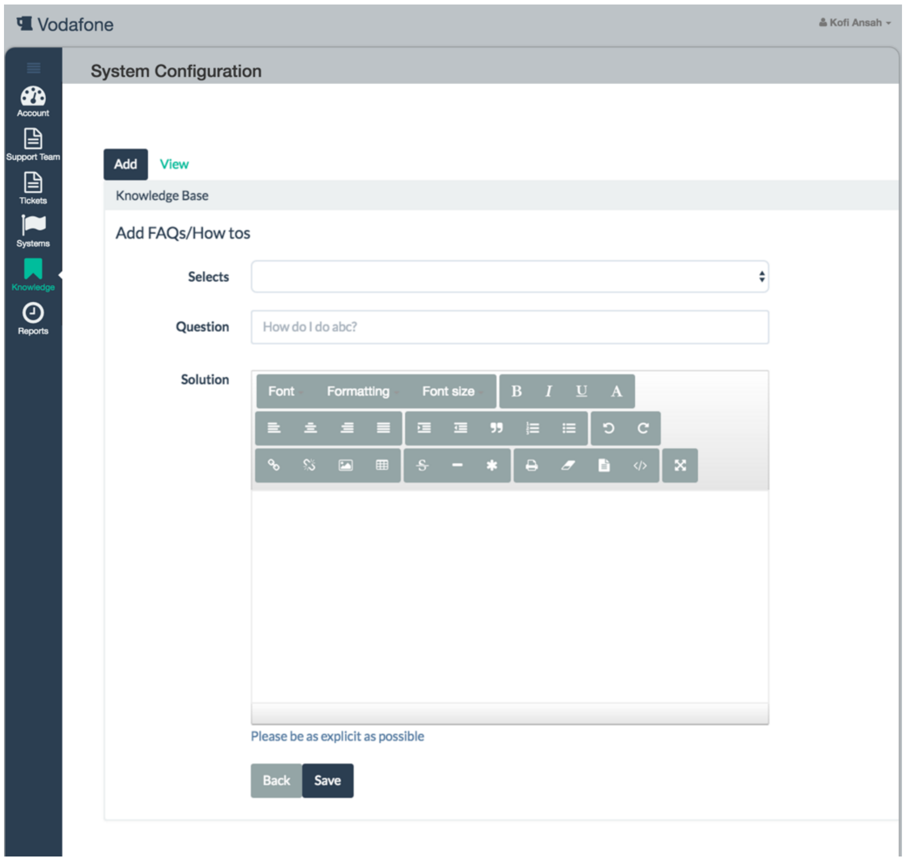
System Configuration

Ticket Details

Customize Your Ticket

Full name	Adjoa Mansa
Email	adjoa@mansa.com
Phonenumber	233123456789
Priority	[✓ Emergency] High Normal Low
Complaint	Please be as explicit as possible
Department	A list of support teams in the company
Category	List of services a user can issue a ticket about
Select File	Choose File No file chosen
Location	location

Figure 4.2 Interface to configure company's ticket



System Configuration

Knowledge Base

Add FAQs/How tos

Selects	
Question	How do I do abc?
Solution	Please be as explicit as possible

Figure 4.3 Interface to add and view content in the knowledge base

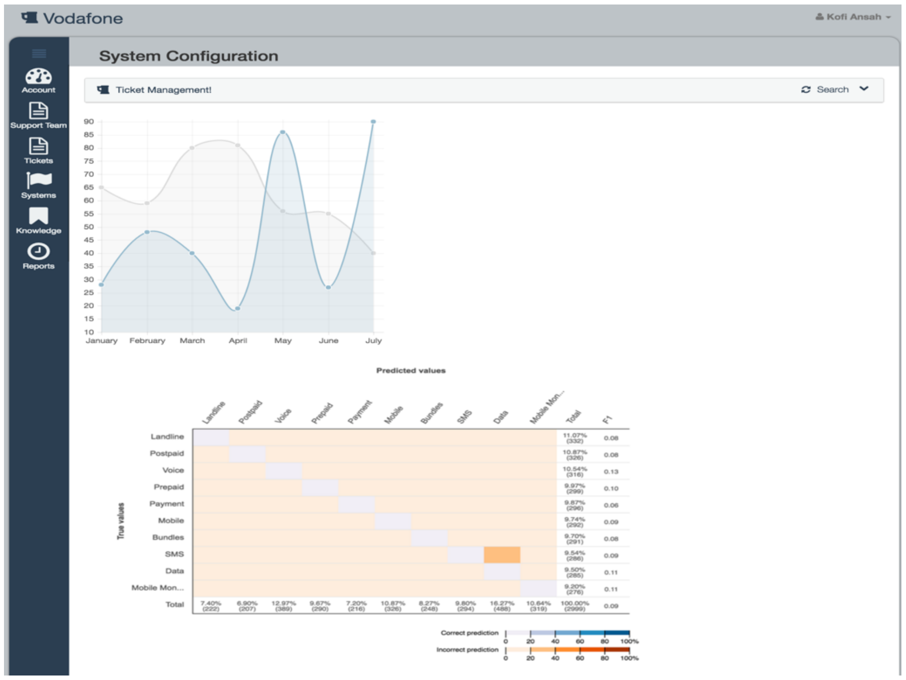
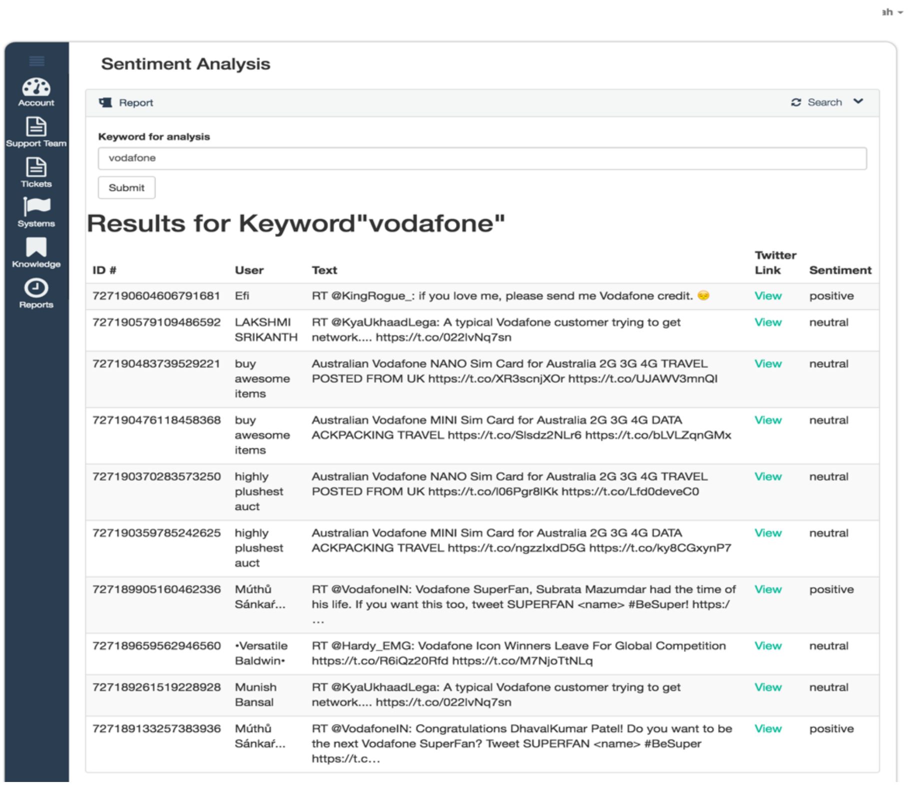


Figure 4.4 Interface to show report generated by the system



Sentiment Analysis

Keyword for analysis: vodafone

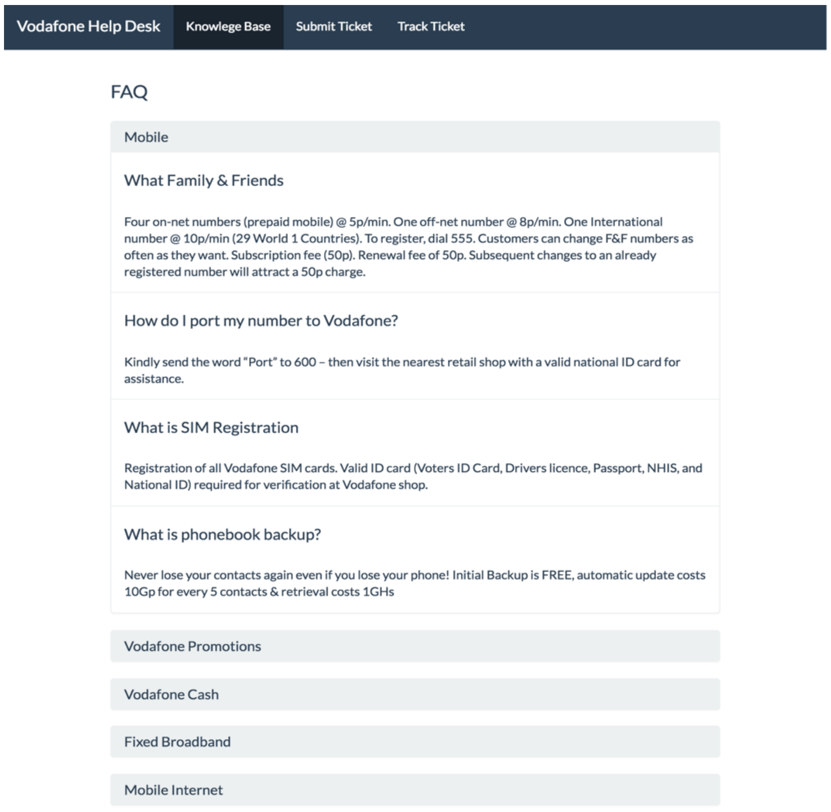
Results for Keyword"vodafone"

ID #	User	Text	Twitter Link	Sentiment
727190604606791681	Efi	RT @KingRogue_: if you love me, please send me Vodafone credit. 😒	View	positive
727190579109486592	LAKSHMI SRIKANTH	RT @KyaUkhaadLega: A typical Vodafone customer trying to get network.... https://t.co/022lvNq7sn	View	neutral
727190483739529221	buy awesome items	Australian Vodafone NANO Sim Card for Australia 2G 3G 4G TRAVEL POSTED FROM UK https://t.co/XR3scnjXOr https://t.co/UJAWV3mnQl	View	neutral
727190476118458368	buy awesome items	Australian Vodafone MINI Sim Card for Australia 2G 3G 4G DATA ACKPACKING TRAVEL https://t.co/Slsdz2NLr6 https://t.co/bLVLZqnGMx	View	neutral
727190370283573250	highly plushest auct	Australian Vodafone NANO Sim Card for Australia 2G 3G 4G TRAVEL POSTED FROM UK https://t.co/l06Pgr8lKk https://t.co/Lfd0deveC0	View	neutral
727190359785242625	highly plushest auct	Australian Vodafone MINI Sim Card for Australia 2G 3G 4G DATA ACKPACKING TRAVEL https://t.co/ngzzlxdD5G https://t.co/ky8CGxynP7	View	neutral
727189905160462336	Múthů Sánkař...	RT @VodafoneIN: Vodafone SuperFan, Subrata Mazumdar had the time of his life. If you want this too, tweet SUPERFAN <name> #BeSuper! https:/ …	View	positive
727189659562946560	•Versatile Baldwin•	RT @Hardy_EMG: Vodafone Icon Winners Leave For Global Competition https://t.co/R6iQz20Rfd https://t.co/M7NjoTtNLq	View	neutral
727189261519228928	Munish Bansal	RT @KyaUkhaadLega: A typical Vodafone customer trying to get network.... https://t.co/022lvNq7sn	View	neutral
727189133257383936	Múthů Sánkař...	RT @VodafoneIN: Congratulations DhavalKumar Patel! Do you want to be the next Vodafone SuperFan? Tweet SUPERFAN <name> #BeSuper https://t.c…	View	positive

Figure 4.5 Interface to show results of sentiment analysis

4.4.2 Client's view

Figures 4.6 to 4.9 enables a client to perform his task, i.e. view knowledge base, submit a ticket and track the progress of his ticket.



FAQ

Mobile

What Family & Friends

Four on-net numbers (prepaid mobile) @ 5p/min. One off-net number @ 8p/min. One International number @ 10p/min (29 World 1 Countries). To register, dial 555. Customers can change F&F numbers as often as they want. Subscription fee (50p). Renewal fee of 50p. Subsequent changes to an already registered number will attract a 50p charge.

How do I port my number to Vodafone?

Kindly send the word "Port" to 600 – then visit the nearest retail shop with a valid national ID card for assistance.

What is SIM Registration

Registration of all Vodafone SIM cards. Valid ID card (Voters ID Card, Drivers licence, Passport, NHIS, and National ID) required for verification at Vodafone shop.

What is phonebook backup?

Never lose your contacts again even if you lose your phone! Initial Backup is FREE, automatic update costs 10Gp for every 5 contacts & retrieval costs 1GHs

Vodafone Promotions

Vodafone Cash

Fixed Broadband

Mobile Internet

Figure 4.6 Client's view for knowledge base grouped by categories

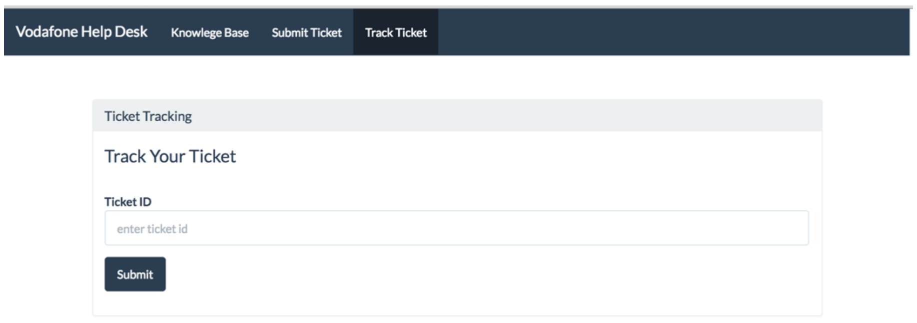
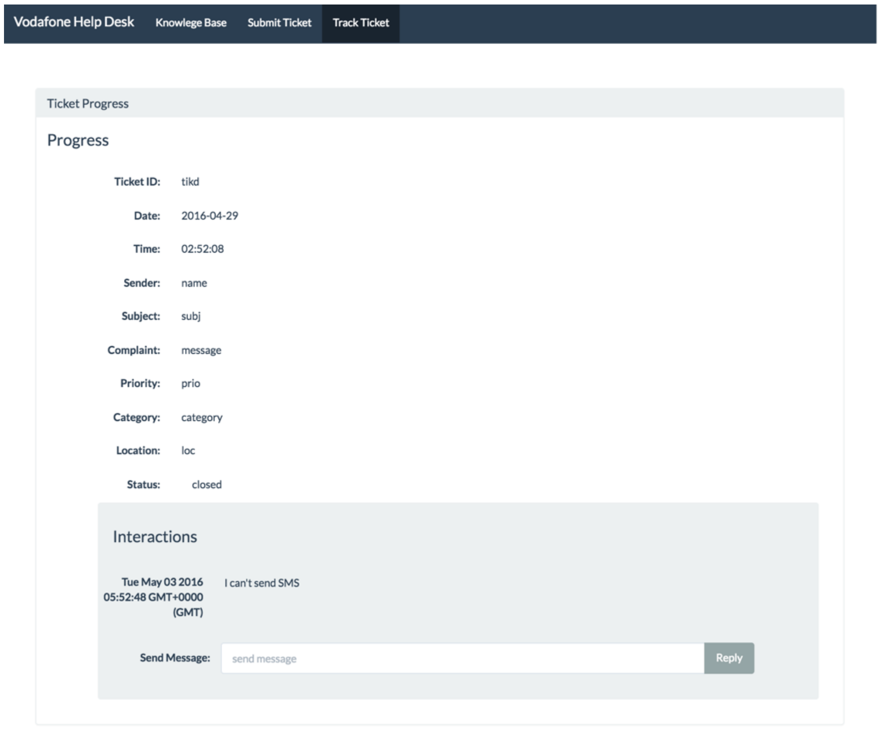


Figure 4.7 Client's view for tracking a submitted ticket



Ticket Progress

Progress

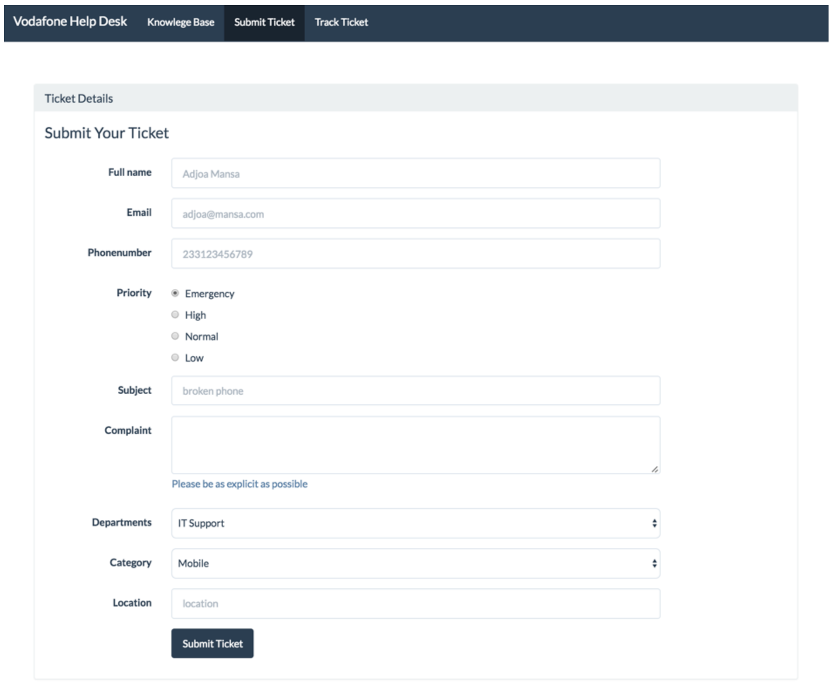
Ticket ID:	tikd
Date:	2016-04-29
Time:	02:52:08
Sender:	name
Subject:	subj
Complaint:	message
Priority:	prio
Category:	category
Location:	loc
Status:	closed

Interactions

Tue May 03 2016 05:52:48 GMT+0000 (GMT) I can't send SMS

Send Message: send message

Figure 4.8 Displays the details of a ticket that is being tracked



Ticket Details

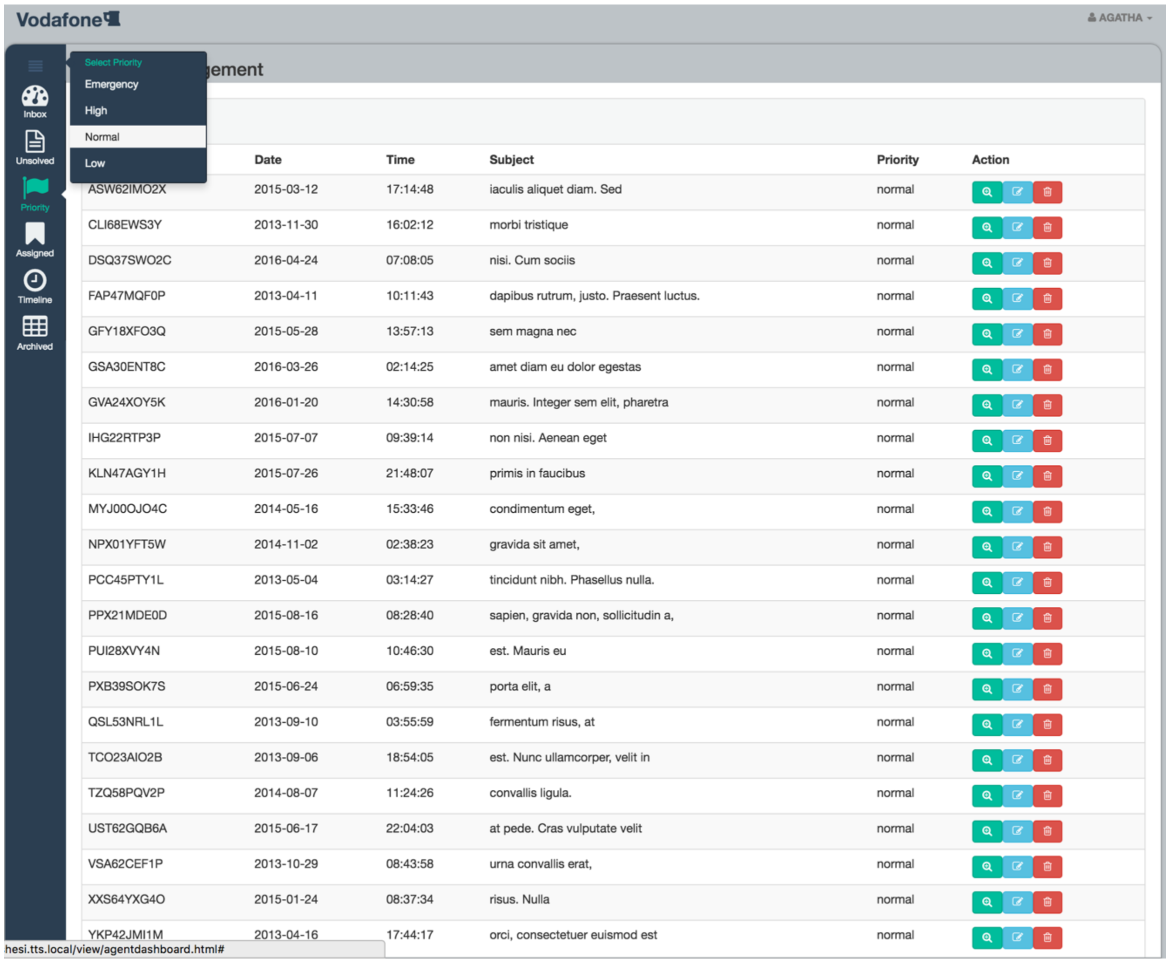
Submit Your Ticket

Full name	Adjoa Mansa
Email	adjoa@mansa.com
Phonenumber	233123456789
Priority	[✓ Emergency] High Normal Low
Subject	broken phone
Complaint	Please be as explicit as possible
Departments	IT Support
Category	Mobile
Location	location

Figure 4.9 Client view to submit a ticket based on the configuration of the company

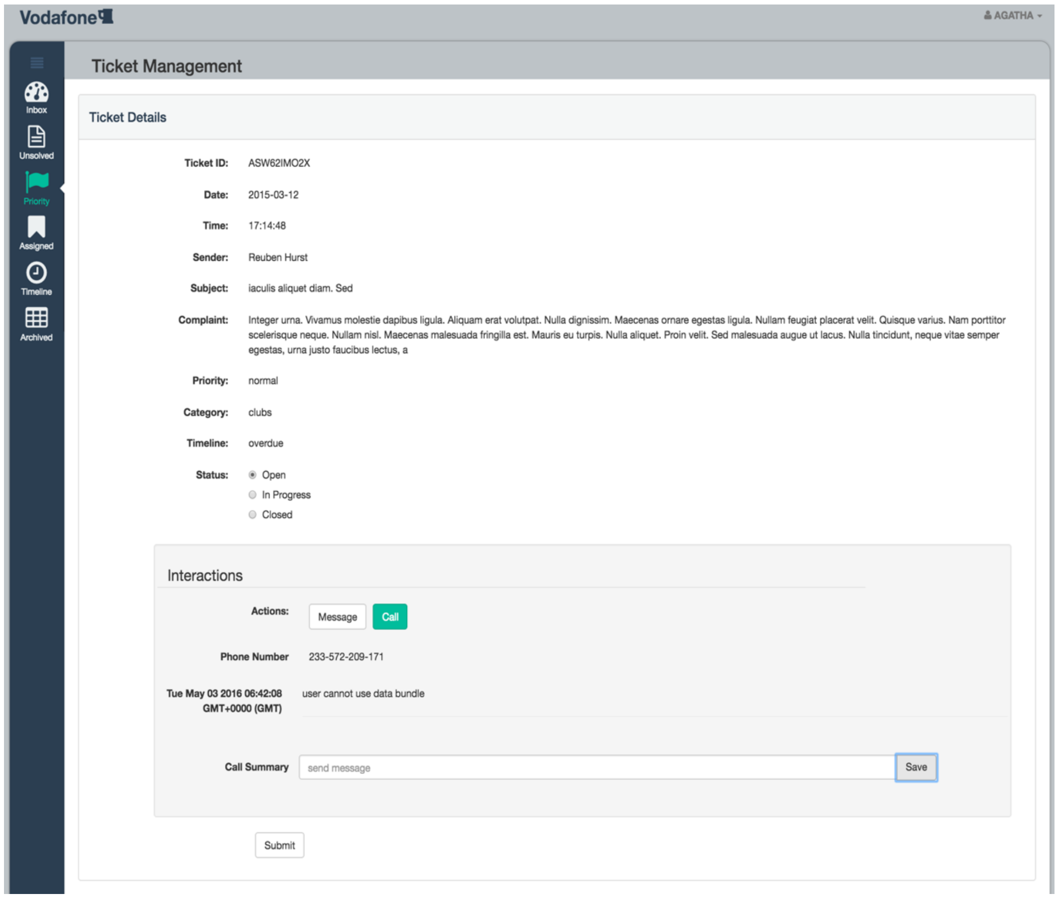
4.4.3 Company personnel's view

Figure 4.10 to 4.11 shows the view of personnel in resolving tickets. An agent is allowed to sort tickets based on the menu provided on the side of his screen. In managing a selected ticket, an agent is allowed to record the interactions in order to complete the resolution process.



	Date	Time	Subject	Priority	Action
ASW62IMO2X	2015-03-12	17:14:48	iaculis aliquet diam. Sed	normal	
CLI68EWS3Y	2013-11-30	16:02:12	morbi tristique	normal	
DSQ37SWO2C	2016-04-24	07:08:05	nisi. Cum sociis	normal	
FAP47MQF0P	2013-04-11	10:11:43	dapibus rutrum, justo. Praesent luctus.	normal	
GFY18XFO3Q	2015-05-28	13:57:13	sem magna nec	normal	
GSA30ENT8C	2016-03-26	02:14:25	amet diam eu dolor egestas	normal	
GVA24XOY5K	2016-01-20	14:30:58	mauris. Integer sem elit, pharetra	normal	
IHG22RTP3P	2015-07-07	09:39:14	non nisi. Aenean eget	normal	
KLN47AGY1H	2015-07-26	21:48:07	primis in faucibus	normal	
MYJ00OJO4C	2014-05-16	15:33:46	condimentum eget,	normal	
NPX01YFT5W	2014-11-02	02:38:23	gravida sit amet,	normal	
PCC45PTY1L	2013-05-04	03:14:27	tincidunt nibh. Phasellus nulla.	normal	
PPX21MDE0D	2015-08-16	08:28:40	sapien, gravida non, sollicitudin a,	normal	
PUI28XVY4N	2015-08-10	10:46:30	est. Mauris eu	normal	
PXB39SOK7S	2015-06-24	06:59:35	porta elit, a	normal	
QSL53NRL1L	2013-09-10	03:55:59	fermentum risus, at	normal	
TCO23AIO2B	2013-09-06	18:54:05	est. Nunc ullamcorper, velit in	normal	
TZQ58PQV2P	2014-08-07	11:24:26	convallis ligula.	normal	
UST62GQB6A	2015-06-17	22:04:03	at pede. Cras vulputate velit	normal	
VSA62CEF1P	2013-10-29	08:43:58	urna convallis erat,	normal	
XXS64YXG4O	2015-01-24	08:37:34	risus. Nulla	normal	
YKP42JMI1M	2013-04-16	17:44:17	orci, consectetuer euismod est	normal	

Figure 4.10 Shows the dashboard of an agent based on selected menu



Ticket Management

Ticket Details

Ticket ID:	ASW62IMO2X
Date:	2015-03-12
Time:	17:14:48
Sender:	Reuben Hurst
Subject:	iaculis aliquet diam. Sed
Complaint:	Integer urna. Vivamus molestie dapibus ligula. Aliquam erat volutpat. Nulla dignissim. Maecenas ornare egestas ligula. Nullam feugiat placerat velit. Quisque varius. Nam porttitor scelerisque neque. Nullam nisl. Maecenas malesuada fringilla est. Mauris eu turpis. Nulla aliquet. Proin velit. Sed malesuada augue ut lacus. Nulla tincidunt, neque vitae semper egestas, urna justo faucibus lectus, a
Priority:	normal
Category:	clubs
Timeline:	overdue
Status:	[✓ Open] In Progress Closed

Interactions

Actions:	Message Call
Phone Number	233-572-209-171
Tue May 03 2016 06:42:08 GMT+0000 (GMT)	user cannot use data bundle
Call Summary	send message Save

Submit

Figure 4.11 Company personnel’s view for managing a selected ticket

Chapter 5 : Test and Results

This chapter discusses the various testing processes the application went through to verify that the software product was built right and satisfied all its functional and non-functional requirements enumerated in chapter 2 of this document. Development testing performed was broken into three aspects: unit testing, component testing and system testing. Each class was tested individually to make sure the functions operate correctly. Test cases and results are shown below.

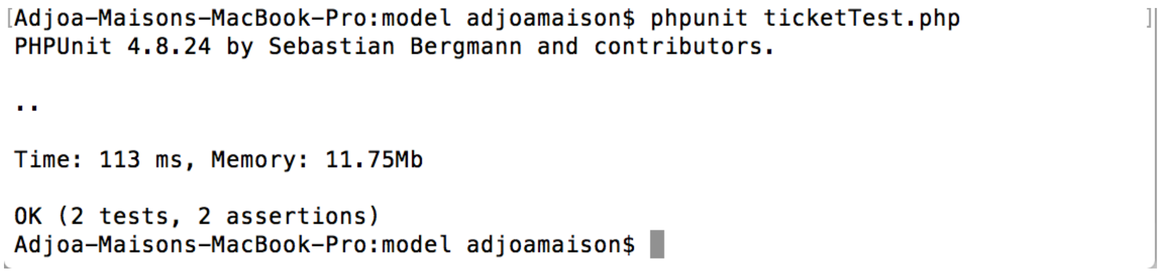
5.1 Unit Testing

Here, object classes were tested in their entirety. Functionality of objects and class methods were verified to ensure proper functioning. This was achieved with the help of PHPUnit testing framework. Input (valid, invalid and null) and output values of methods from the various object classes are are evaluated. Below are code snippets of the PHPUnit test and a screenshot of the test result for two methods in the implementation of this system.

```
<?php
  class TicketTest extends PHPUnit_Framework_TestCase{
    public $tname = "angsTicketTable";
    public $tid = "unittes";
    public $sender = "my name";
    public $status = "new";
    public $msg = "some message to be tested";
    public $number = "2332640452342";
    public $department = "1";
    public $email = "email@email.com";
    public $date = "2016-04-2";
    public $time = "02:23:34";
    public $priority = "high";
    public $category = "1";
    public $file = "NULL";

    public function testAddTicket(){
      include_once ('ticket.php');
      $obj = new ticket();
      $row=$obj->addTicket($tname,   $tid,   $date,   $time,
$status,         $sender, $email, $number, $priority, $msg,
$department,  $category, $file);
      $this->assertEquals(true,$row);
     }

    public function testViewTicket(){
      $ticket = new ticket();
      $row=$ticket->ticketDetails($tname,$tid);
      $this->assertEquals(true,$row);
    }
  }
?>
```
Listing 5.1 A snippet of PHPUnit test code to test the ticket class.


```
Adjoa-Maisons-MacBook-Pro:model adjoamaison$ phpunit ticketTest.php
PHPUnit 4.8.24 by Sebastian Bergmann and contributors.

..

Time: 113 ms, Memory: 11.75Mb

OK (2 tests, 2 assertions)
Adjoa-Maisons-MacBook-Pro:model adjoamaison$
```
Figure 5.1 PHPUnit test result for 2 functions

5.2 Component Testing

In this section, the individual units tested above are integrated to create composite components. This was done to find defects that came about as a result of the unit integration. Below are the test results of a few components in context of the application. A representation of the basic functionalities in all the components are shown with these few tests results below.

5.2.1 Test Case: Create all required tables for an organization in the database

- Precondition: A MySQL query that creates a ticket table given the company name should exist. This should be activated once an administrator completes the sign up process.

- Expected Result: The creation of a new ticket, knowledgebase, interactions, assignment, companyStaff, department, category, companyDetails and report tables with the given company name in the database (this is to initialize an organization).
- Test Results

Table 5.1 Test results for creating all required tables in the database

Valid Inputs	**Result**
Name of company as a string e.g. My Company or MyCompany or My Company’s Name	A valid JSON output returns result to be equal to 1. Ticket table is created with the company’s name as prefix. E.g. “MyCompanyTicketTable” or MyCompanysNameKnowledgeBase.
Invalid Inputs	**Result**
No name	Return a JSON with result being 0 (unsuccessful). Ticket table is not created

5.2.2 Test Case: Add a Ticket to the database

- **Precondition**: A MySQL query that adds a new ticket should exist. A client or agent must have an issue whose solution cannot be found in the knowledge base section.
- **Expected Result**: The addition of a new ticket record in the company’s ticket table with a unique generated id.
- **Test Results**

Table 5.2 Test result for adding a new ticket to the database

Valid Inputs	**Result**
All text inputs are filled with strings, e.g. valid email, phone number, correctly selected priority category or department	A valid JSON output returns result to be equal to 1. New ticket record is successfully stored in the database.
Invalid Inputs	**Result**
No input for any given field	A valid JSON output returns result to be equal to 0.

(text input field, select input field, radio button). Invalid email address Invalid phone number	

5.2.3 Test Case: View Ticket details

- Precondition: A MySQL query that fetches the details of a ticket must exist. The ticket whose details must be viewed must exist given a valid id.
- Expected Result: All details of the selected ticket must be displayed to the user.
- Test Results

Table 5.3 Test result for fetching ticket details for the database

Valid Inputs	**Result**
Valid ticket id	Returns a JSON encoded row that has the details of the given ticket fetched from the database and a result equal to 1.
Invalid Inputs	**Result**
Wrong ticket id	Returns JSON encoded message with result equal 0.

5.2.4 Test Case: Sort Tickets by a given parameter

- **Precondition:** There must be a number of tickets available in the database.
- **Expected Result:** Display a list of sorted tickets based on a given parameter
- **Test Results**

Table 5.4 Test result for sorting tickets by a given parameter

Valid Inputs	**Result**
Click menu item for the desired sorting parameter	Returns a JSON encoded rows that satisfy the given parameter and a result equal to 1.
Invalid Inputs	**Result**
Null input, invalid ticket id	Returns JSON encoded message with result equal 0.

5.2.5 Test Case: Update a given ticket

- **Precondition:** Ticket that needs updating must exist.
- **Expected Result:** Change the status of a given ticket
- **Test Results**

Table 5.5 Test result for updating a ticket

Valid Inputs	**Result**
Select desired ticket from the list of tickets displayed Select value to change ticket status i.e. from 'new' to 'in progress' or 'closed' or archive	A valid JSON output returns result to be equal to 1. Update row in the database with the new values selected.
Invalid Inputs	**Result**
Null input, invalid ticket id	Returns JSON encoded message with result equal 0.

5.3 System Testing

In order to test the integrated system, sample data was generated from generatedata.com. This was stored in the database and various interfacing functions called on it to test component integration. Due to the fact that data generated did not support some level of constraining, there were conflicting fields. For instance, every archived ticket must have its status as closed however, the generated data could not handle this constraint. This therefore did not give a true representation of real life data and therefore generated biased results.

Chapter 6 : Conclusion and Recommendations

6.1 Recommendations

This section discusses some suggested areas to further improve the efficiency of this system.

- Automate ticket assignment. As it stands now, customers have to manually assign tickets issued to a particular department and category or category based on an organizations configuration. Most users may assign these tickets wrongfully since they may not have an idea of the working structure of the organization. In this vein, an automated ticket assignment feature will relief them of this burden as well as see to it that the right group of people handle a new ticket issue. This will increase efficiency and speed up ticket resolutions.
- Natural language processing and classification of complaints. Currently tickets are classified based on their categories which is just one way of describing tickets. For better insights and predictions, the subject and complaints of every ticket could be processed to pick out key areas of interest such as what kind of assistance the ticket is requesting, prior conditions that led to the problem, any similarities with existing tickets or knowledge in order properly classify tickets to make future resolutions easy. This will also help in trend finding and for appropriate measures to be taken.
- Automate ticket creation from phone calls, emails and social media feeds. To fully cover all communication channels that a client is likely to use to voice out their concerns. For future work, these channels could be explored such that the system would automatically generate tickets out of emails, phone calls and social media feeds.

- Deploy this system as a Software as a Service (SaaS). This would be a way to monetize this software since the registered organizations will not have to worry about hosting and infrastructural management. This service will be made available to organizations via the internet.

6.2 Conclusion

To sum up, this project sought to develop a centralized database repository for all customer and company systems complaints or faults respectively. To this end, a trouble ticketing system was developed to serve as a portal to bring together these complaints or faults. The centralized repository in turn serves as input for the Amazon Machine Learning web service to to automatically find patterns in these reported errors in order to make predictions for new data points as and when they become available. Also, with the Datumbox Machine Learning framework, sentiments of customers were retrieved and analyzed from social media (Twitter) in order to give the organization a general sense of how they are perceived by the public. Altogether, the administration of the organization is presented with enough insights into what areas of support they should invest in to sell themselves off as proactive.

The relevance of this project can be measured by how much organizations will save in their bid to retain customers, provide better customer care by meeting them at the point of need in ample time and proactively taking care of faults even before their customers realize them.

REFERENCES

Abiw-Abaidoo, K. (2011). *Predicting customer churn in the mobile telecommunication industry, a case study of MTN Ghana, Kumasi* (Unpublished thesis Kwame Nkrumah University of Science and Technology, Kumasi). Retrieved from http://ir.knust.edu.gh/bitstream/123456789/4431/1/Kojo%20Abiw-Abaidoo.pdf

American Express. (2011). *Good service is good business: American consumers willing to spend more with companies that get service right, according to American Express survey* [Press Release]. Retrieved from http://about.americanexpress.com/news/pr/2011/csbar.aspx

Ban, F., Gbahoué, B., & Schneider, J. (2013, July). *Higher satisfaction at lower costs: Digitizing customer care*. Retrieved March 30, 2016, from McKinsey&Company: http://www.mckinsey.com/~/media/mckinsey/industries/telecommunications/our%20insights/lessons%20from%20digigtal%20telecos%20initiatives%20to%20improve%20business%20performance/higher_satisfation_at_lower_costs_digitizing_customer_care.ashx

Barwise, P., & Meehan, S. (2004). *Simply better: Winning and keeping customers by delivering what matters most.* Boston: Harvard Business School Press.

Hull, P. (2013, December 6). *Don't get lazy about your client relationships.* Retrieved from Forbes:http://www.forbes.com/sites/patrickhull/2013/12/06/tools-for-entrepreneurs-to-retain-clients/#7c6f6efd4b7c

Lumor, C. (2014). *press*. Retrieved from About MTN: www.mtn.com.gh/about-mtn/press/mtn-responds-to-nca-consumer-survey-and-assures-customers-of-continous-improvement

News Ghana. (2015). *MTN wins four Ghana Telecom awards. Retrieved from http://www.newsghana.com.gh/mtn-wins-four-ghana-telecom-awards/*

Sommerville, I. (2011). *Software engineering.* Boston: Pearson.

Tutorialspoints. (n.d.). *Software Design Strategies*. Retrieved April 20, 2016, from Tutorialspoint: http://www.tutorialspoint.com/software_engineering/software_design_strategies.htm