**SE 4485 Software Engineering Project Final Report**

Design Build and Enrich Knowledge Base Based on Machine Learning

Sponsored by DXC Technology

Chandra Kamalakantha

Jonathan Lawrence

Jeremiah Ramilo

Jacob Wilson

**Executive Summary**

The document is meant to cover the purpose and scope, organization, requirements, architecture, design, and testing plan of our project *Design Build and Enrich Knowledge base Based on Machine Learning*. By reading this document, the reader gains an effectual understanding of all aspects of our project.

**Table of Contents**

List of Figures

List of Tables

1. Introducduction

1.1 Purpose and Scope

1.2 Product Overview

1.3 Structure of the Document

1.4 Terms, Acronyms, and Abbreviations

2. Project Management Plan

2.1 Project Organization

2.2 Lifecycle Model

2.3 Risk Analysis

2.4 Hardware and Software Resource Requirements

2.5 Deliverables and Schedule

2.6 Monitoring, Reporting and Controlling Mechanisms

2.7 Professional Standards

2.8 Evidence all Artifacts Have Been Placed Under Configuration Management

2.9 Impact of the project on individuals and Organizations

3. Requirements Specification

3.1 Stakeholders

3.2 Use Case Model

3.2.1 Graphic Use Case Model

3.2.2 Textual Description for Each Use Case

3.3 Rationale for Use Case Model

3.4 Non Functional Requirements

4. Architecture

4.1 Architectural Styles Used

4.2 Architectural Model

4.3 Technology, Software, and Hardware Used

4.4 Rationale for Architectural Style and Model

5. Design

5.1 GUI

5.2 Static Model Class Diagrams

5.3 Dynamic Model Sequence Diagrams

5.4 Rationale for Your Detailed Design Model

5.5 Traceability from Requirements to Detailed Design Model

6. Test Plan

6.1 Requirements/ Specifications Based System Level Test Cases

6.2 Traceability of Test Cases to Use Cases

6.3 Techniques Used for Test Generation

6.4 Assessment of Goodness of Your Test Suite

Acknowledgment

References

**List of Figures**

Graphic use case model

Logical View

Process View

Development view

Physical view

Scenarios

Static model – class diagrams

Dynamic model – sequence diagrams

**List of Tables**

Deliverables and Schedule

Traceability from requirements to detailed design model

Traceability of test cases to use cases

**1.Introduction**

Reading this document will grant the user an understanding of all aspects of project. Including the decisions we made and why we made them. All aspects of our project are covered including organization, requirements, architecture, design, and testing. There are also several diagrams and charts in this document to visualize certain elements of our project. Overall, this document will cover anything the reader would like to know about our project.

1.1.Purpose and Scope

The purpose of our project was to provide a product that could return articles from a database based on keywords entered by the user. The product also uses a weighted search algorithm to provide the best and most relevant results to the user. Overall, this project was designed to give the user easy access to relevant and accurate information based on what they are looking for.

1.2.Product Overview (including capabilities, scenarios for using the product, etc.)

This project uses a weighted search algorithm to return relevant articles based on a keyword entered by the user. The users will access the project through a webpage hosted on Amazon’s S3. The backend is written in python and is a lambda function. All communication between the frontend and backend is handled by Amazon’s Api gateway. Finally, The articles are stored in a DynamoDB database. The main scenario for the use of this project is for a support employee to enter a keyword based on a client submitted ticket and receive all relevant knowledge base articles in return. Overall, this project is a very useful tool for giving relevant information to clients.

1.3.Structure of the Document

This document is broken up into several sections that cover various aspects of the project. The project management plan is the first topic, which is followed by requirements analysis. After requirements analysis is discussed the project’s architecture is detailed for the reader. The final two sections of this report are the design of our project, and our testing plan. It is the hope of the authors that by reading this report you gain an accurate and detailed understanding of our project.

1.4.Terms, Acronyms, and Abbreviations

AWS - Amazon Web Services

API - Application Program Interface

**2.Project Management Plan**

2.1.Project Organization

Team Members

* Jeremiah Ramilo - Team Leader
* Jonathan Lawrence
* Jacob Wilson

2.2.Lifecycle Model Used

We will be using the waterfall model for this project. The waterfall model has a very sequential structure which will be of great advantage to our timeline. This is because we can work backwards from our deadline and divide certain periods of time for each stage of the model. This will ensure a well designed and tested system. Furthermore, this project will be a learning experience for all team members and will require collaborative work to accomplish the goal. The waterfall model will allow for the team to be in sync as we collaboratively work on requirements, design, implementation, etc.

2.3.Risk Analysis

As with any project, there can be a number of associated risks in its execution that may affect the effort required for its delivery. Some of the possible risks include that:

* The team has limited practical experience on the subject of machine learning and artificial intelligence. Although the team have all been exposed to these concepts through coursework and lectures, we lack the hands-on knowledge of developing and training learning machines. The team can mitigate this risk by prioritizing tasks correctly, discussing with the team for guidance, and starting tasks early to allow for ample time to produce the necessary deliverables.
* The team is working remotely from the project sponsor. As both parties have unpredictable schedules and limited windows of time for meeting, it may be difficult to correspond with one another. To mitigate the risk, the team will plan to meet virtually on most Fridays at 1:00 PM. If another meeting time is necessary, the team should discuss and set aside time for a makeup meeting. If a makeup meeting is not possible, team members are responsible to report updates on the status of their work on the project by email to all members. To report updates on progress, direct the email to the team leader and CC all other team members.
* The team has a high workload for the current semester. As college seniors, we are all finishing up our final courses in hopes to graduate soon. The stress and difficulty with managing our various responsibilities may potentially act as a hindrance. To help manage the workload, any team member can request help on any portion of the project they do not understand. Moreover, we will use GitHub’s built-in Kanban board to manage tracking the progression and assignment of tasks.

2.4.Hardware and Software Resource Requirements (Do not forget to describe what new software or hardware each team member learned during the project)

* Postman for RESTful web service development
* Visual Code or Atom for Python development
* ServiceNow for knowledge base test data
* RabbitMQ for messaging framework
* GitHub for version control
* GitHub Kanban board for task management
* Cloudcraft for architectural diagrams
* AWS
  + Lambda
  + DynamoDB
  + S3
  + API Gateway

During the course of this project each team member learned new software that will benefit them over the course of their career. For example, Jacob learned DynamoDB, Jeremy learned a lot about Python and how to write use Python to write powerful programs, and Jonathan learned how to use AWS S3 to build the front end of our project. We also all had to learn and understand AWS API Gateway and Lambda functions.

2.5.Deliverables and Schedule

|  |  |
| --- | --- |
| **Project Management Plan** | **Friday, September 6th, 2019** |
| **Requirements Documentation** | **Friday, September 20th, 2019** |
| **Architecture Documentation** | **Friday, October 4th, 2019** |
| **Detailed Design Documentation** | **Friday, October 25th, 2019** |
| **Testing Plan** | **Friday, November 11th, 2019** |
| **Final Project Registration** | **Thursday, December 5th, 2019** |
| **Final Project Report** | **Friday, December 6th, 2019** |
| **Weekly Meetings with Sponsor** | **Fridays at 1:00 PM** |
| **Makeup Meetings with Sponsor** | **Upon discussion, if weekly meeting not possible** |

Any portion of the project that a group member is responsible for should be completed **by 10:00 AM on the due date**. If any group member is not able to meet that deadline, alert other group members as soon as possible. The sooner everyone is alerted, the more efficiently we can address the issue.

2.6.Monitoring, Reporting, and Controlling Mechanisms

Monitoring, reporting, and control mechanisms are very important to any project, and this is even more true for a project that includes several members and will span several months. Management reports should be produced weekly and include progress towards next deliverable, events of the weekly group meeting, and any problems that need to be addressed by the group. Project monitoring and version control will be done through Github. The rationale for using detailed weekly reports and Github is to provide the group with an easy way to monitor project progress, and to allow for the easy management of different branches of the project while it is in development.

2.7.Professional Standards

Since our group is working with a professional organization we must maintain a high standard of conduct throughout the entire project. Scholastic dishonesty will not be tolerated in our group, and if it occurs will be immediately reported to the professor. Meetings will occur once weekly on Fridays at one in the afternoon with attendance being mandatory for all members. If someone cannot attend they should alert the group as soon as possible so arrangements can be made for them or the meeting can be rescheduled. All deliverables need to be of a high quality that would be acceptable in a professional setting. It should be clear that deliverables had effort put into them and were not rushed through at the last minute. The rationale for these standards is to allow for the successful completion of the project to our sponsor's satisfaction.

2.8.Evidence all the artifacts have been placed under configuration management

We are using Github for configuration management. The link to our Github is <https://github.com/orgs/Senior-Design-F19/teams/dev>.

2.9.Impact of the project on individuals and organizations (Include a description of what impact your project will have on individuals and society)

This main impact of this project will be on DXC Technology’s customers and its employees. This project will allow DXC Technology to better serve its customers by granting its employees easy access to relevant information on a wide range of topics that their customers might inquire about. The employees lives will be improved by this project making their jobs easier, and the lives of DXC Technology’s customers will be improved by the project increasing the quality of the information that they receive.

**3.Requirement Specifications**

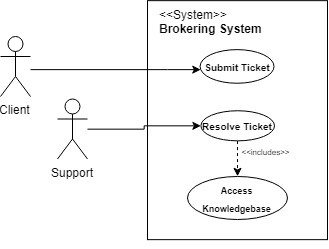
3.1.Stakeholders for the system

The stakeholders for this system are DXC Technology, its support employees, and the clients who will use the system.

3.2.Use case model

Are use case model has the system interact with two actors: the client, and the DXC support employee. The client will submit a ticket, and the support employee will take the ticket and query the system based on the search term in the ticket. After querying the system the support employee will return the results to the client.

3.2.1.Graphic use case model



3.2.2.Textual Description for each use case

**UC-0: Access Knowledgebase**

**Participating Actors**: Client, Support

**Entry Condition(s)**: Support employee receives technical issues service ticket from Client

**Normal Flow of Events**:

1. Support Employee will search for a knowledgebase article based off of Client’s ticket
2. Relevant knowledge base articles will appear as results

**Exit Condition(s)**: Support employee will choose article that is used for solving Client’s request

**Special Requirements**: Help Desk Employee has written relevant knowledgebase article

**UC-1: Resolve Ticket**

**Participating Actors:** Client, Support

**Entry Condition(s):** Support employee successfully retrieves relevant knowledgebase article to

resolve a ticket for Client’s technical issue.

**Normal Flow of Events:**

1. Support employee sends knowledgebase article to Client
2. Client confirms that knowledgebase article satisfies their request

**Exit Condition(s) -** Support employee marks ticket as resolved

**Special Requirements -** Help Desk employee has written relevant knowledgebase article

**UC-2: Submit Ticket**

**Participating Actors:** Client, Support Employee

**Entry Condition(s):** Client submits a ticket detailing their technical issue

**Normal Flow of Events:**

1. The client submits a ticket to DXC Technology.
2. The support employee receives the ticket.

**Exit Conditions(s):** The support employee beings the process of resolving the ticket.

**Special Requirements:** The client must have a reason to submit a ticket.

3.3.Rationale for your use case model

The system interacts with two actors: the Client, and the Support employee. There are two main functions of the NLP machine learning system. Firstly, the system will handle the submission of technology-related tickets from the client and assign them to support employees. This function is defined as *Submit Ticket*. Using the aid of the model, our knowledgebase queried for related articles. These articles are returned to the assigned support employee and used to resolve the technical issue indicated by the client. This function is defined as *Resolve Ticket*.

3.4.Non-functional requirements

* **NF-0:** The system must be secure
* **NF-1:** Search response time needs to be under 2 seconds
* **NF-2:** System should have a certain uptime
* **NF-3:** The system must be scalable
* **NF-4:** The system should return accurate results

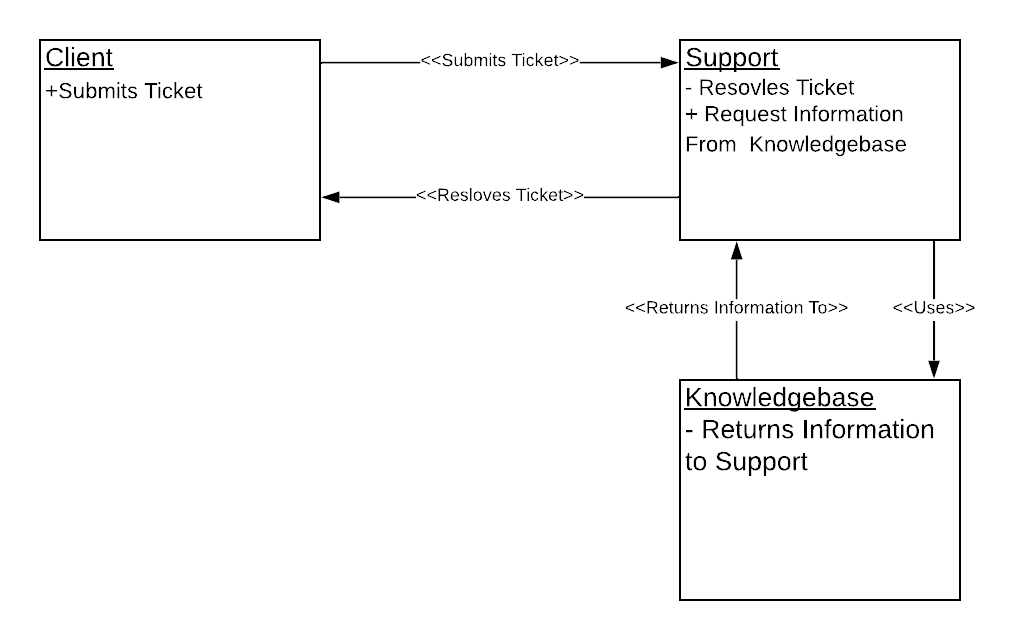
**4.Architecture**

4.1.Architectural style(s) used

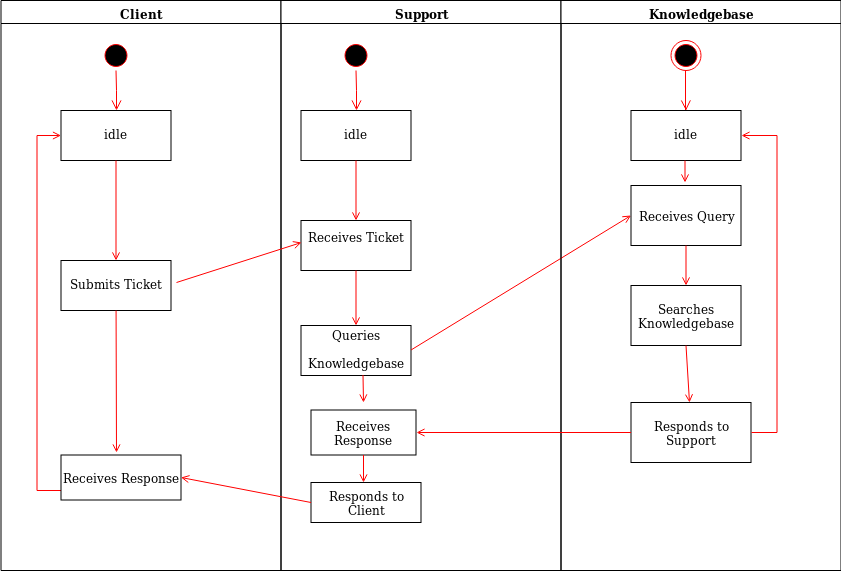
The architectural style we decided would best fit our project was client server because it supports various features of our application. For instance, a DXC employee will request information from our knowledge base, and the knowledge base will return the requested information to the DXC employee. This is classic client server architecture with the DXC employee acting as the client and the knowledgebase acting as the server. Overall, this architectural style is the best fit for our project, and will allow us to complete this product to our clients satisfaction.

4.2.Architectural model

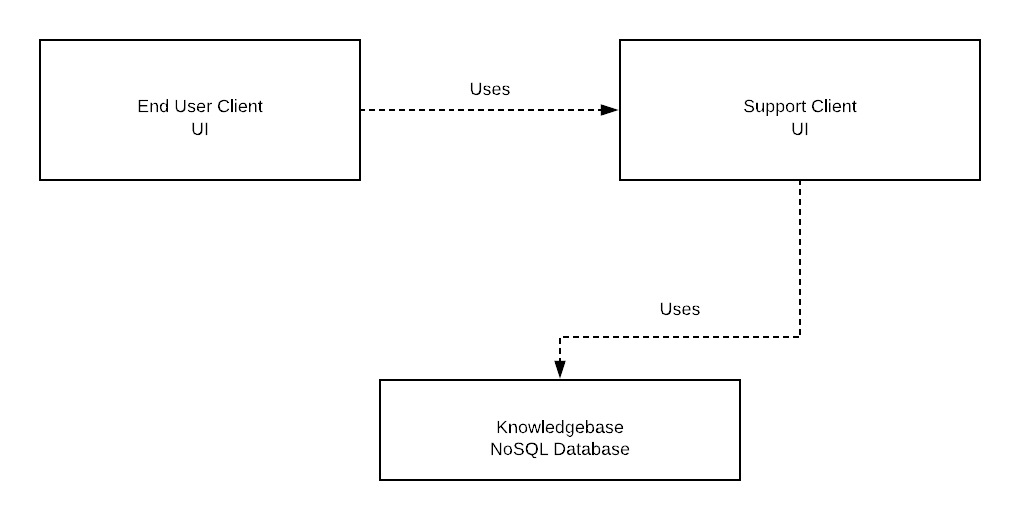
Logical View



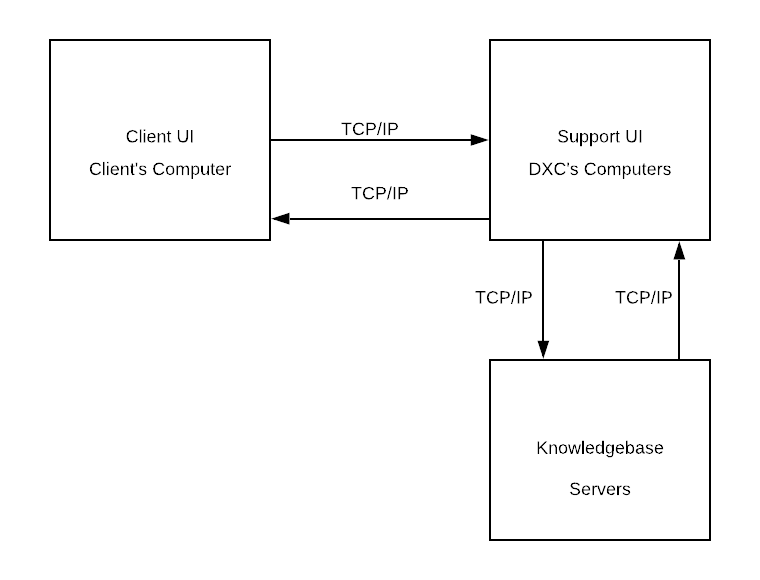
Process View



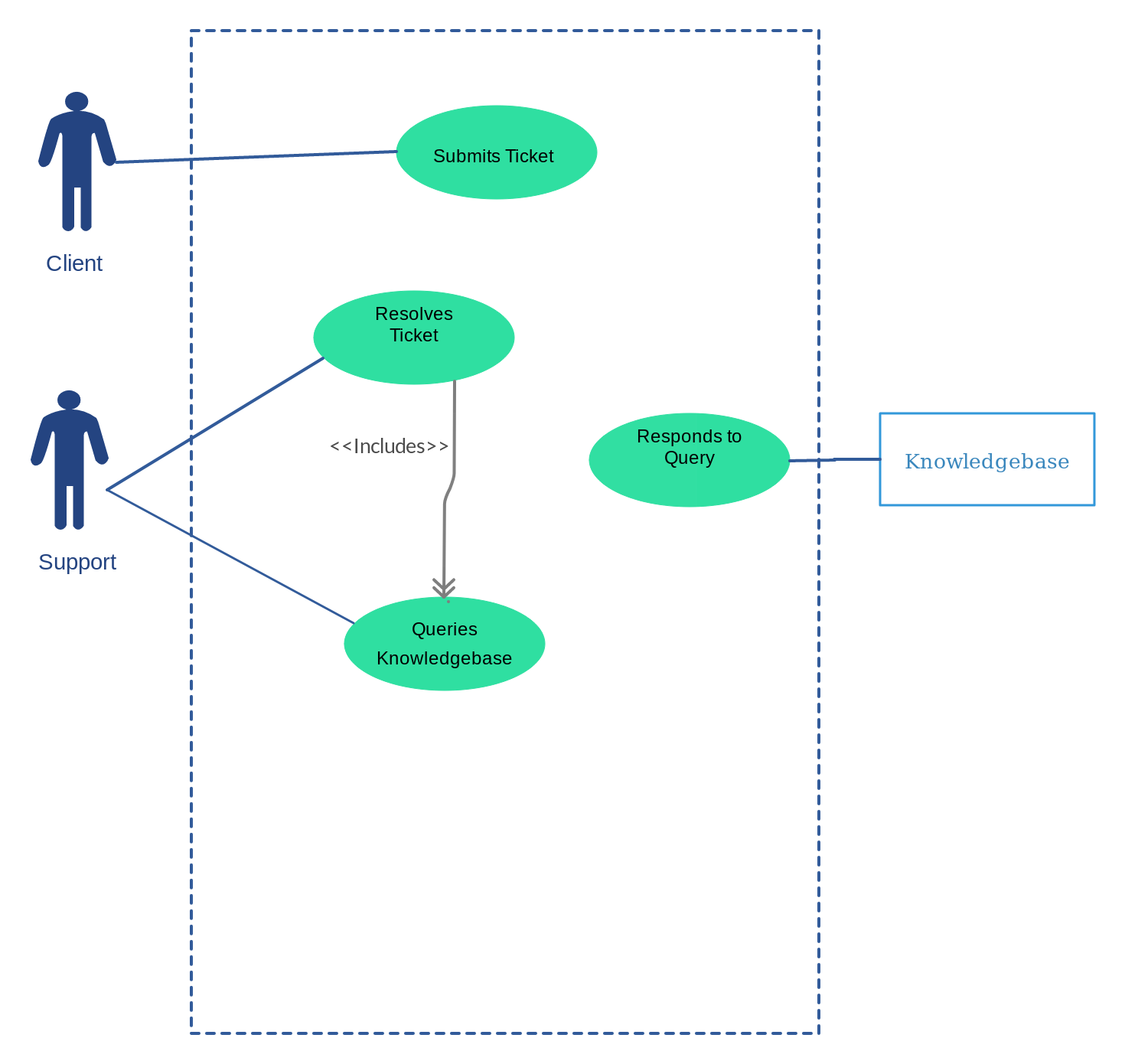
Development view



Physical view



Scenarios



4.3.Technology, software, and hardware used

For this project we will use several technologies including but not limited to Python MongoDB, JSON, and Service Now. Python will be used as the main programming language for this project. We decide MongoDB would be the best way to implement it our database because of its versatility and ease of use. Json will be used to format data, and finally Service Now will be used as well. Overall these technologies will make implementation of this project simple and efficient.

4.4.Rationale for your architectural style and model

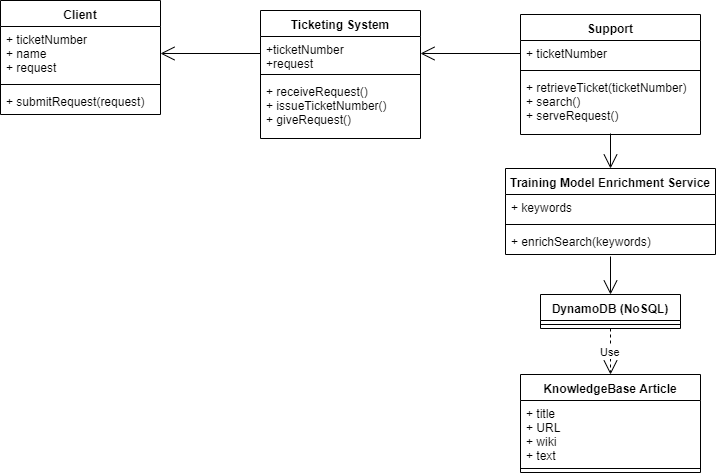
Our rationale for choosing Client Server as our architectural style was because it fits our project very well. The DXC employee will act as the client and the knowledgebase will act as the server. Our decision to use the four plus one model was based off our desire to provide a very clear model of our architecture, and we believe that the four plus one model is the best way to provide that clarity.

**5.Design**

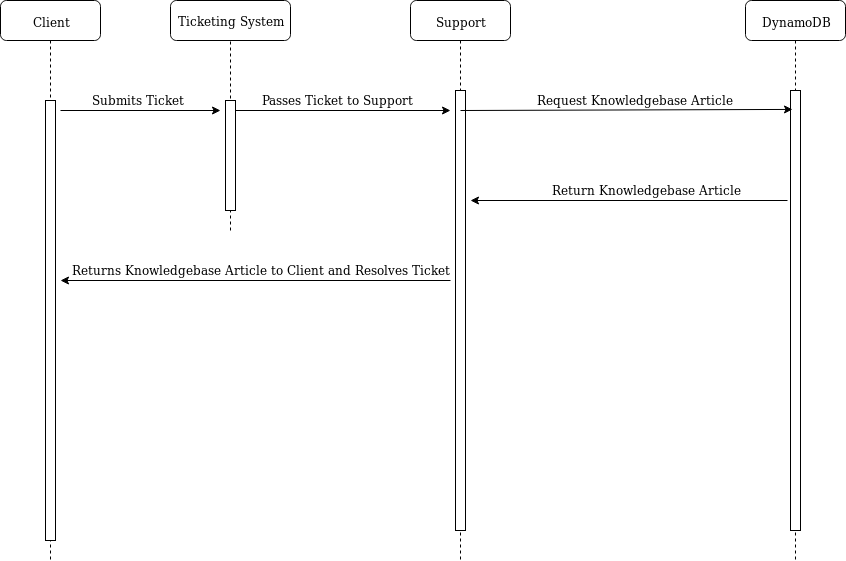
5.1.GUI (Graphical User Interface) design

N/A

5.2.Static model – class diagrams



5.3.Dynamic model – sequence diagrams



5.4.Rationale for your detailed design model

Our rationale for our detailed design model was to provide a usable product that fulfills the project requirements (including searching from a database, using a search enrichment model to populate results, and retrieving results). Our design shows the relationships between each entity involved in the process and what each entity can do. Using this design gives a clear representation of of our project and will also make implementation relatively simple so that completing the project will not be a problem.

5.5.Traceability from requirements to detailed design model

**F-0:** The system shall return a knowledge base article results should be relevant to the keywords used

**F-1:** Knowledgebase articles should be found based off of client’s ticket

**F-2:** Model should be able to process natural language

**F-3:** Model should use machine learning to become more accurate over time

**NF-0:** The system must be secure

**NF-1:** Search response time needs to be under 2 seconds

**NF-2:** System should have a certain uptime

**NF-3:** The system must be scalable

**NF-4:** The system should return accurate results

**TC-0:** To test the functional requirements we will submit several client tickets written in plain english and test what is returned. We will judge the knowledgebase articles returned based on relevance to keywords and relevance to the test client ticket. This test case will also judge how the system handles natural language and if the system improves over time.

**TC- 1:** The database will be tested for its ability to handle NoSQL injections

**TC- 2:** Support personnel will search for a knowledge base article and await the response time of the database.

**TC- 3:** The database and search enrichment model will be tested for documentation and ability to increase or decrease functionality

**TC- 4:** Support personnel will search for a knowledge base article and view for the relevancy of the returned articles

**TC- 5:** The system will be monitored regularly and have system data to measure uptime or the presence of any failures or crashes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **TC-0** | **TC-1** | **TC-2** | **TC-3** | **TC-4** | **TC-5** |
| **F-0** | X |  |  |  |  |  |
| **F-1** | X |  |  |  |  |  |
| **F-2** | X |  |  |  |  |  |
| **F-3** | X |  |  |  |  |  |
| **NF-0** |  | X |  |  |  |  |
| **NF-1** |  |  | X |  |  |  |
| **NF-2** |  |  |  |  |  | X |
| **NF-3** |  |  |  | X |  |  |
| **NF-4** |  |  |  |  | X |  |

**6.Test Plan**

6.1.Requirements/specifications-based system level test cases

**TC-0:** To test the functional requirements we will submit several client tickets written in plain english and test what is returned. We will judge the knowledgebase articles returned based on relevance to keywords and relevance to the test client ticket. This test case will also judge how the system handles natural language and if the system improves over time.

**TC- 1:** The database will be tested for its ability to handle NoSQL injections

**TC- 2:** Support personnel will search for a knowledge base article and await the response time of the database.

**TC- 3:** The database and search enrichment model will be tested for documentation and ability to increase or decrease functionality

**TC- 4:** Support personnel will search for a knowledge base article and view for the relevancy of the returned articles

**TC- 5:** The system will be monitored regularly and have system data to measure uptime or the presence of any failures or crashes

6.2.Traceability of test cases to use cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **TC-0** | **TC-1** | **TC-2** | **TC-3** | **TC-4** | **TC-5** |
| **UC-0** |  |  |  |  | X |  |
| **UC-1** |  |  | X |  |  |  |
| **UC-2** | X |  |  |  |  |  |

6.3.Techniques used for test generation

To generate test cases we looked at how the system would be used on a day to day basis and tried to make test cases that would reflect that usage by basing them of use cases. We also made a few test cases to test technical requirements, such as uptime. Most of our test cases are black box test cases because the most important outcome for this project is how it functions for the end user. We did however generate a few white box test cases to test non-functional requirements like security and up time. Our criteria for judging test cases was whether or not they covered a functional, non-functional requirement, or a use case. If the test case did cover one of those it was considered a useful test case.

6.4.Assessment of the goodness of your test suite (Which metrics were used for such assessment?)

The metrics we used to asses our testing suite was whether or not each of our use cases were covered. We wanted to make a test suite that covered all ways in which our project was likely to be used. Overall, our test suite was of a high quality because it covered each of our use cases as well as our non functional requirements.

Acknowledgment

We would like to acknowledge our sponsor Chandra Kamalakantha for guiding us through this project. Without him, this project would not be possible. We would also like to Acknowledge Professor Eric Wong, and the teaching assistant LinghuanHu.

References (Must be complete, correctly formatted using the standard for IEEE Conference Proceedings)

[1] S. W. Ambler, “UML Use-Case Diagrams,” *The Elements of UML™ 2.0 Style*, pp. 33–46.

[2] “AWS General Reference: Reference Guide,” *AWS General Reference* , no. 1.0, 2019.