Microservice Architecture: Emergency Room

By Shalin Singh

May 7 2022

1 Abstract

In this paper, we examine the semantics that go into building medical software using a modern microservice architecture approach. This paper includes the invdividual contribution done by Shalin Singh, Computer Science student at the University of Hartford. We'll look at the components done for the server side and client user interface. The application developed was a Hospital Emergency room system that manages patient, employee and billing information. The objective of the system is to intake and manage patient information in a medical emergency. The user interface is supposed to be easily understood from a client and adminstrator aspect. This document will also include the requirements and deliverables requested by the client, Dr. Ingrid Russel.

2 Mapping out the first set of requirements

For the initial set of requirements, It was hard to understand what use cases were needed. It was easier figure out the requirements by modeling the system. As a result, I went straight to diagramming and modeling how the data would be stored and processed for the application. I started out by understanding the entitles that would be needed in the database. The entities I started off with were Patient, Intake Patient, Medical Condition, Employee (Doctor/Nurse), Results, Permissions, Medication, and a hospital room. These entities were constructed based on the requirements of the client. I spent most of the time understanding which entitles would require a 1 to Many, Many to Many or 1 to 1 relationship. After these data relationships were established, then it was easier to understand what basic requirements are needed for the application. Some of the product functions we produced were employee management service, patient intake service and a procedure/room assignment service. All these "services" were constructed because of the DIA diagram. We formulated the user characteristics based on the employee permissions service. Some of the user characteristics were based on whether the user is a patient, nurse, administrator, Doctor, or Janitor. These permissions are necessary since a doctor and nurse have different permissions to access a patient. After the product functions and user characteristics were created, we produced specific product functions. As for my individual contribution in the first set of requirements, I focused on what use cases would be generated off (One to Many) and (Many to Many) relationships. Some fo these use cases included, prescribe patient medication and assigning a patient a medical procedure. I was also responsible for structuring the inital code base and how our database, web and backend services would interact. I structured the code based on a modern day microservice architecture pattern. The pattern involved seperating code into sections for the database, HTTP controllers and models for a custom ORM (Object Relational Mapper). The client requested that object oriented concepts should be implemented in the final product. Overall, these were my initial individual contributions for the first client deliverable.

3 Functions Generated From Modeling the System

I formulated the specific requirements by looking at functions needed for each individual entity. The first universal requirement that's needed in all software is CRUD (Create, Read, Update, Delete). For every entity I added a create function, which inserts a instance of the entity in the database. I added a "read" function which views all instances of the entity in the database. We also added "update and delete/delete all" for all entitles. All these functions are Soley responsible for populating and manipulating data in the database. Our first set of functions was for our first entity (Patient). The functions included Verifying the date of birth and calculating the BMI. The next set of functions was for the nurse (Employee). The client specifically mentioned having a function that allows the nurse to update patient notes on intake, as a result we added a function for that. I created a function for the nurse that admits/intakes the patient. The nurse also has a function to add symptoms to the patient on intake. The next set of functions were related to the Doctor (Employee). As a group, we decided that the doctor is a "parent" to the nurse and inherits all the functionality of the nurse. Therefore, the doctor can add symptoms, intake patients and update notes as well. Some additional functions that the doctor has are being able to prescribe medication, diagnose a patient with a medical condition and discharge a patient from the emergency room. Overall, these functions are what created the foundations for our use cases in Deliverable 3. Next, I will go into other requirements such as Performance, Maintainability and Architecture requirements. I will also touch on some security concerns for the application such as memory leaks and networking vulnerabilities.

4 Implementation And Source Code

Now I will go into the code base and it how it was structured. In short, we used a modern-day micro service architecture pattern. This means that all the components of our system can be deployed on different machines etc. We separated the application into 4 modules. One module was dedicated to all database functions for sqlite3. Our second module was the "Controllers "or Json rest Api web endpoints. Our third module was a custom ORM with all our database entities implemented as classes. This module was made so that we could implement the required object-oriented concepts the client wanted. If the requirements were to expand for "business logic," we would have a module dedicated for the services. This application does not have a large repository for business logic, so we have combined it into the controller's module.

5 Data Analysis

The settings/environment of this research can be replicated on most computers with atleast 4GB of ram. The specific specs was Ubuntu 20.04 8GB ram on Jupyter Notebooks running multi notebook kernels. The dataset takes 10.65 seconds to load on the initial dataframe. We'll go over the data processing for each algorithm implemented. First was K means clustering for severity clustering. We wanted to create severity clusters based off the attributes ”Harmed/Killed” and ”Property Value Damage”. On the inital data processing, we gathered the Harmed/Killed and Property damage on a yearly basis. After plotting the initial data points, I collected the Harmed/Killed and property value on a monthly basis to add more dense clusters on the graph for K Means Clustering, lastly we label encoded the data in case text values occured in the ”Property value” column. Overall, this was the data processing involved for K means clustering. The next implemented aglorithm was ”K Nearest Neighbors”. The first thing that was done was renamed the columns to camel case to make it easier to identify specific columns. Then we filtered for our specific columns which were ”Harmed”, ”Killed” and ”Terrorist Group” to build the model. Then I used ”StandardScaler” (Another form of label encoding) from sci-kit learn. I used the Scalar to encode all the columns into each column. Then we created a loop of 31 iterations and fed in K values on an incremental basis. Then I graphed the max key value each time the classifer was called. The data was prepared in this format to create a elblow like curve when viewing KNN. The next algorithm was a Decision Tree to view which attacks lead to suicide. The end of the tree is supposed to lead to 2 options. ”1” Is the class that represents Suicide and ”0” is the class that represents ”Not suicide”. The first thing I did was grab the columns that I wanted such as ”Harmed/Killed”, ”Weapon Type” and ”Attack Type”. Just like the other algorithms, we label encoded the columns, then used the DecisionTree classifier from sci-kit learn to plot the results. Lastly, for the Natural Language processing algorithms, we created functions to aggregate all the summaries into large paragraphs. The libraries used were Spacy and NLTK for the natural language algorithms. The algorithms I used were named entity recognition and a knowledge graph. Both these algorithms used the sentence aggregation function to prepare the language data I needed. In the next section we’ll examine the results from the data prepared for the algorithms.