# **Rule Based Alerting System**

# Introduction

Rule Based Alerting System is a Web application that provides services for the following purposes:

* Admitting a patient to the ICU
* Generating alerts for the patients in the ICU
* Discharging the patient from the ICU

# Rationale

This project models the entire workflow of a patient as he/she goes through the process of critical care.

# Guide to understand

The following steps are involved in the entire workflow:

The first step is the admission of the patient to the ICU ward. The configuration of the ICU, ie., the number of beds, occupancy of each bed, and details of the admitted patients are maintained in a SQL database called ICUDB. The number of beds has been determined beforehand by the ICUDB.

ICUDB also maintains the vitals of the patients admitted as well as the number of free and occupied beds in the ICU in a table called “ICUState” and the occupancy of each bed is maintained in a table called “Beds”.

To setup this database, one must download the SQL Server Management Studio application and build a database called “ICUDB” with 2 tables called “ICUState” and “Beds” with the following schemas:

ICUState :

PatientId varchar(50) Primary key

Spo2 int

Temp float

PulseRate int

bedsOccupied int

bedsFree int

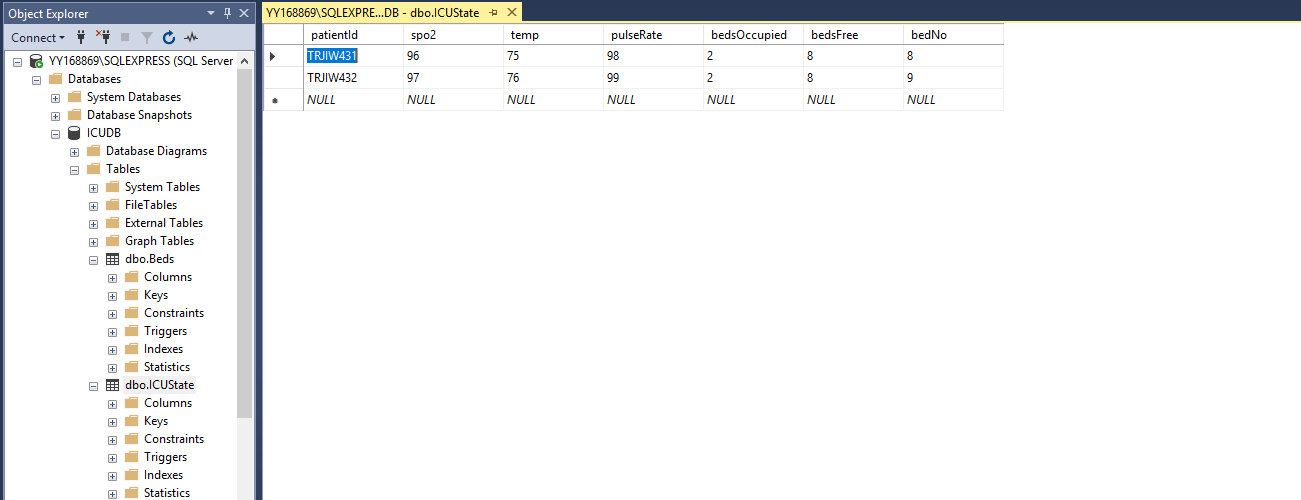
bedNo int

Foreign key bedNo references Beds.bedNo

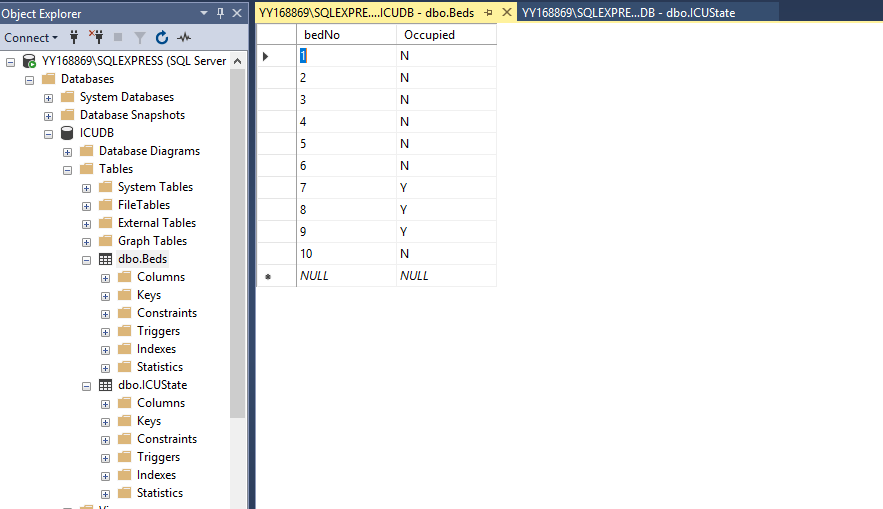
Beds:

bedNo int Primary Key

Occupied char(10)

The URL for the PatientAdmitter can be found on Swagger and be used to admit a patient upon which the ICUDB shall be updated to look like this:

Also, corresponding to the bed number provided in the URL, the beds table will also be updated. It will look somewhat like this :

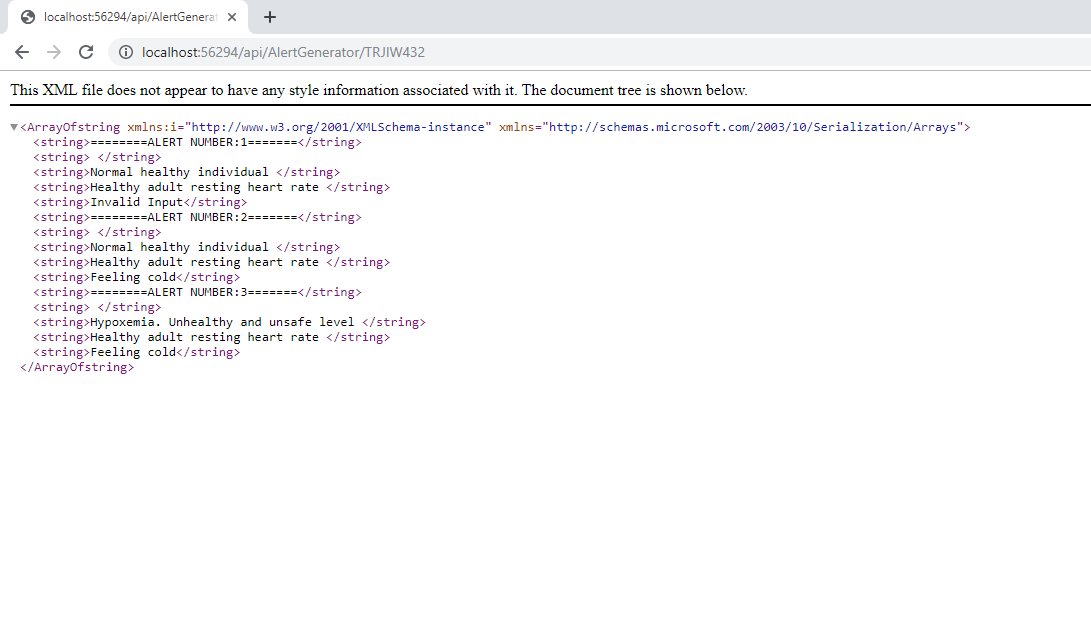


This completes the patient admission process.

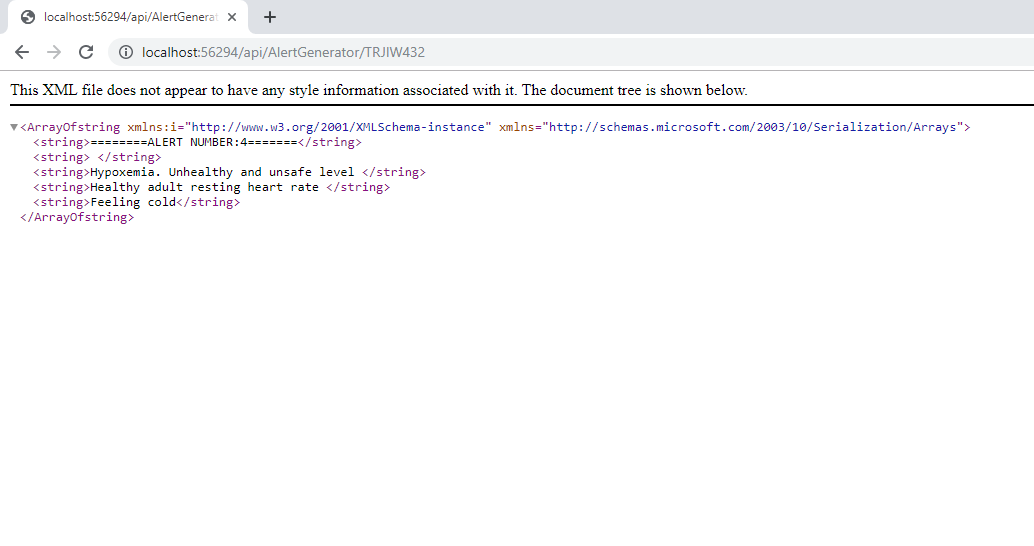
Once a patient is admitted in the ICU, we have an option to view alerts for that patient based on the patient’s vitals. A buffer of alerts for the patient is maintained until we use the AlarmShutter to clear all the alerts for that patient. We also have options for discharging the patient, which consequently frees up the corresponding bed in the ICU ward.

The AlertGenerator and AlarmShutter are provided under AlertController and PatientAdmitter and PatientDischarger are provided under NurseStationController.

Each time the AlertGenerator URL is hit, it provides the updated list of alerts for that patient which would look somewhat like this:



Once the alarm shutter URL has been hit, the previous list of alerts for the patient shall disappear and only the current alert will be shown:



Finally, we also have the option to discharge a particular patient given their patientId and their bedNo.

This would delete their entry from the “ICUState” table and update the “Beds” table to make their bed free.

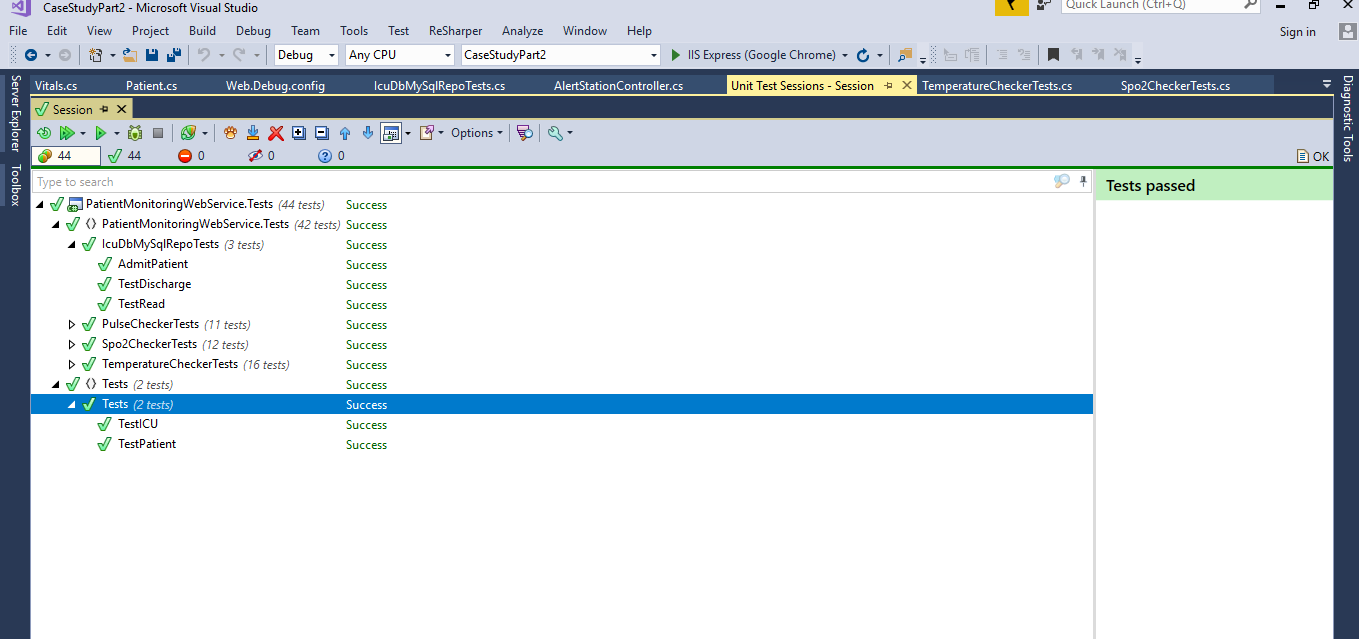
# Code Quality Evidence

## Duplication

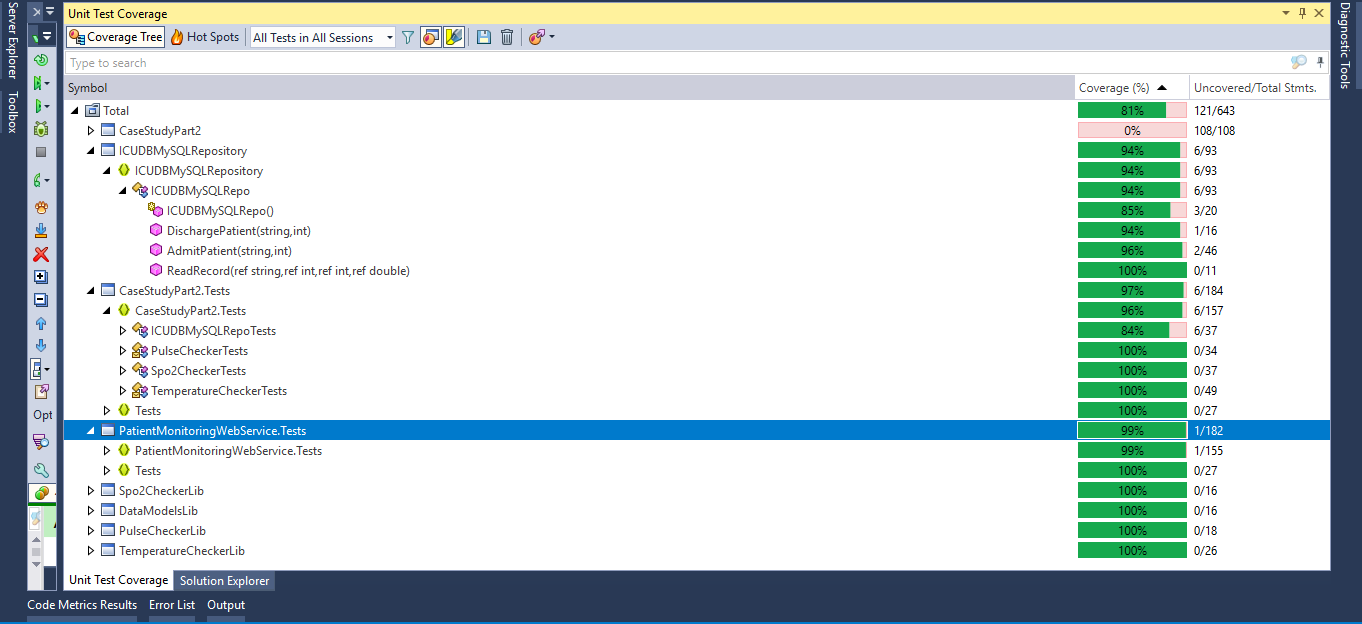
We have used the simian tool to analyze the duplication. We have a 0% duplication for a threshold value of 6.

## Cyclomatic Complexity

## Unit Test Report



## Unit Test Coverage Report



## Rationale for not Covered Code

Coverage for Global.asax.cs and SwaggerConfig.cs files has not been provided as they are configuration files which can’t be tested.

Also the API Controllers aren’t covered because the APIs (services) inside the controllers are being tested using Postman not by NUnit.

The uncovered files are:

##### Global

##### SwaggerConfig

##### WebConfiguration

##### AlertStationController

##### NurseStationController

API Testing:

We have tested the API using Postman. The following is a sample screenshot of one of the tests for alert generation:

