**Knowledge Base for Incident Management System**

Table of Contents

[Project Definition 2](#_Toc489308629)

[**Objective:** 2](#_Toc489308630)

[**Current Business Problem:** 2](#_Toc489308631)

[**Proposed Solution:** 2](#_Toc489308632)

[**In Scope and Out of Scope** 2](#_Toc489308633)

[Technology Stack: 2](#_Toc489308634)

[Business Context: 3](#_Toc489308635)

[Wireframes/Navigation: 3](#_Toc489308636)

[Data Model/Expected Data Feed and types: 3](#_Toc489308637)

[Test strategy: 3](#_Toc489308638)

[Timelines: 3](#_Toc489308639)

# **Project Definition**

## **Objective:**

Create a standalone interface that can plug into the Incident and request management system, built using latest Dot Net MVC framework version. The input for this system can be either relational data from the existing system, different type of flat files or images with Metadata describing the type of issue and resolution comments.

## **Current Business Problem:**

The current system does not support:

* Centralized tracking and storing of information from current and past incidents
* Repeated incidents need quicker resolution and preferably an automated process

## **Proposed Solution:**

The proposal is to develop a system that can

1. Give a searchable interface to search for similar incidents from the past for quick resolution
2. The system should auto update itself with machine learning from new requests and incidents input from day-to-day for future reference.
3. The solution should be plugged in to existing system with little or no integration changes

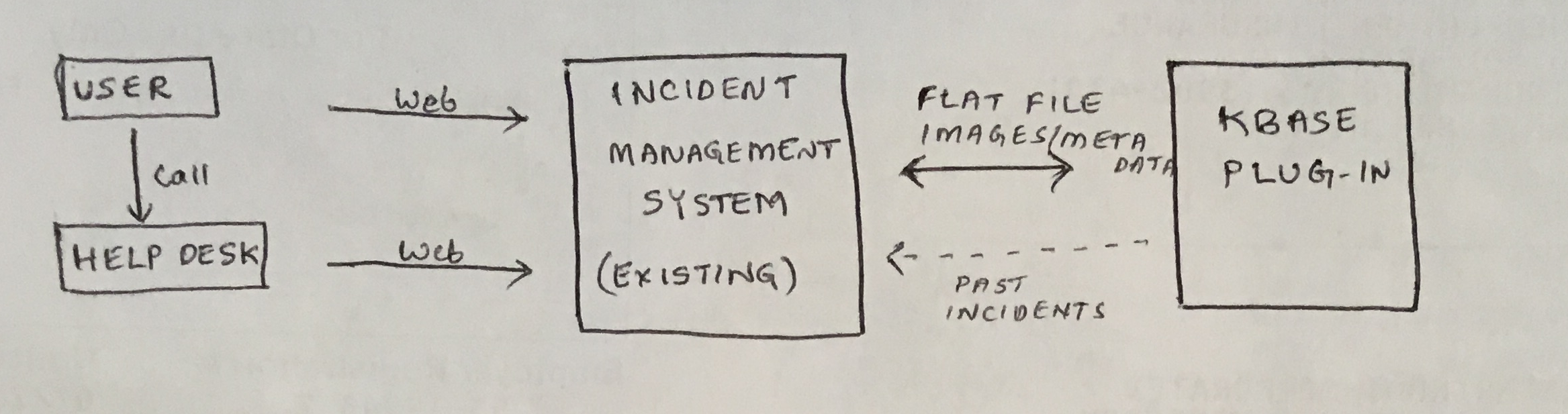
## **In Scope and Out of Scope**

Any components related to incident management system are out of scope. The

# **Technology Stack:**

|  |  |
| --- | --- |
| Front End | Asp.net MVC 5 c#, AngularJS and Bootstrap |
| Database | SQL Server 2016 |
| ETL | SSIS |
| Architecture Type | SOA |

# **Business Context:**

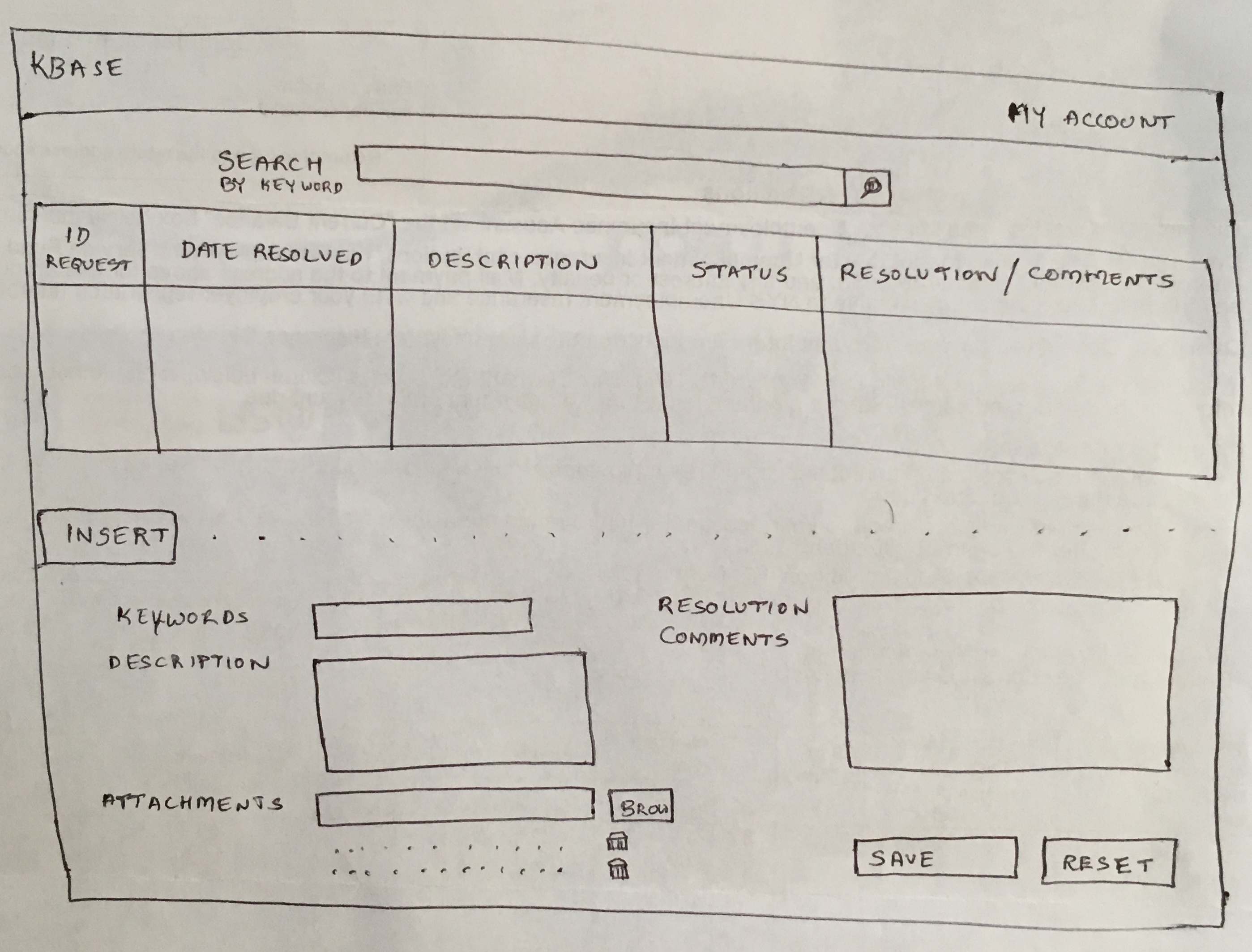


# **Wireframes/Navigation:**

A screen for navigating thru the knowledge base and add more references to the existing requests. User will be able to click on the request Id and go into the details for more specifics of the resolution or knowledge base item. This will be plugged into the existing client’s incident management system.

The data gets loaded into the database on the backend and gets auto updated.

**Future feature**: We will write a module to run a crawl job to identify and update the keywords every night and hence, make the system more efficient by having an auto update and easy/quick future searches. Please note this is not in the scope of initial prototype we are building.



# **Data Model and Data Sources:**

**Input feed:** The incident and request management system in use by the client is source of data for this system. Data is anticipated to come in either flat file or image formats – images should have meta data. If enterprise data transfer tools like SSIS are used to load into the source tables for the system, it is recommended and efficient method.

Core KB table#1:

|  |  |  |
| --- | --- | --- |
| **Data Field** | **Data type** | **Description** |
| KBid | Int | Unique id for each knowledge base item |
| Title | Nvarchar |  |
| Description |  |  |
| Logged Date |  |  |
| CreatedFor |  |  |
| Createdby |  |  |
| Comments/Resolution |  |  |
| Status |  |  |
| Requestid |  |  |
| AttachmentId | Int | Foreign key (table #3: attachment id) |

Table#2: Keyword

|  |  |  |
| --- | --- | --- |
| **Data Field** | **Data Type** | **Description** |
| Keyword | Varchar |  |
| Request Id | Varcahr | Foreign key (table #1: request id) |

Table#3: Attachments

|  |  |  |
| --- | --- | --- |
| **Data Field** | **Data Type** | **Description** |
| Attachment | Varchar | File Path to attachment |
| Attachment Id | Varcahr |  |

# **Test strategy:**

# Unit testing would be performed using the data created by our team on the stand-alone system for the time being. Assuming the client provides the real world samples; Hexaring would use the data and provide the testing results. If the end product is agreed to be installed on the client environment, Hexaring will help in setting up the system for Integration testing and performance testing, if needed. In this case, Hexaring will perform the smoke test for the scope of system developed by Hexaring.

# **Timelines:**

2-3 weeks