Project Report

On

# TICKET MANAGEMENT SYSTEM



|  |  |  |
| --- | --- | --- |
| **Professor:** | | **Prepared By:** |
| Bill Gauvin | | Mayank Yadav (U94319493) |
|  |

###### April 2018

Master of Science

In

Information Technology

University of South Florida

# Tampa, FL 33620

##### **INTRODUCTION**

##### A Ticketing System streamlines, centralizes and manages all your tickets. Take email requests and transform them into service tickets automatically. From the initial ticket request and creation to resolution and closure ticketing system software helps you through every stage.

##### A Ticketing System allows you to automate ticket assignment, routing, and escalation to the right agent at the right time. Save time and manual effort for the help desk team, and improve help desk agent productivity. With the help of Web-based portal to log tickets, and an intuitive console for technicians to manage tickets. A ticketing system software enables you to:

##### Automate ticket assignment, routing and escalation to the right agent, at the right time

##### Centralize Ticket Management – from request creation to resolution

##### Save time and resources on manual and repetitive tasks

##### Track and monitor help desk and technician performance in real time

##### Improve operational efficiency of customer service

##### Realize higher levels of customer satisfaction

A Ticketing System centralizes the channel for receiving service requests via an interactive Web portal. Teams can benefit from avoiding the use of disparate sources such as chat, phone, email and in-person discussions for customer communication. Once the request is logged in the Ticketing System, pros communicate back to the end-user with acknowledgement and update ticket fulfillment progress including technician assignment and expected request fulfillment date/time. Ticketing software also converts inbound service requests via email into tickets in the Ticketing System.

System DEsign

### **Use Case Diagram**

##### E:\Education\Sem 8\8th Sem.CE-IT -- Project Report Format (w.e.f May 2015)\8th Sem.CE-IT -- Project Report Format (w.e.f May 2015)\UseCaseDiagram2.jpg

Figure : Usecase Diagram

**Entity Relationship Diagram**

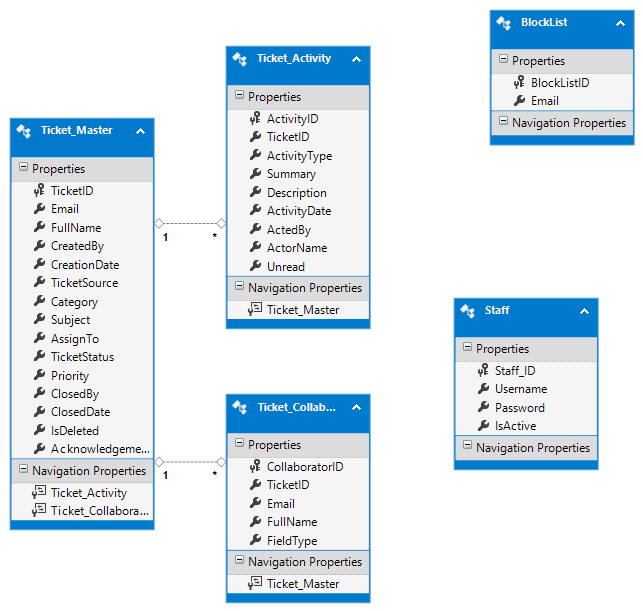


Figure : ER Diagram

### **DATA DICTIONARY**

###### **TicketMaster**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| Ticketid | int | Primary Key |
| Emailaddress | nvarchar(500) | Not Null |
| Fullname | nvarchar(500) | Null |
| Createdby | int | Null |
| Creationdate | datetime | Not Null |
| Ticketsource | nvarchar(50) | Not Null |
| Category | nvarchar(50) | Null |
| Subject | nvarchar(max) | Not Null |
| Assignto | int | Null |
| Ticketstatus | nvarchar(50) | Not Null |
| Priority | tinyint | Null |
| Closedby | int | Null |
| ClosedDate | datetime | Null |
| Isdeleted | bit | Not Null |
| Acknowledgementsent | bit | Not Null |

Table 1 - Ticket MAster

##### “TicketID” is the primary key of “TicketMaster” table and is auto incremental. “CreatedBy” stores the “UserID” of the user who created the ticket. If the ticket is created by emailing them value is Client’s ID. “TicketSource” has following values: Web, Phone and Email. Ticket created via staff is in the category of Web and Ticket created via email is in the category of Email. “AssignTo” stores the “UserID” of the user who has been assigned the particular ticket to solve. If the ticket is created by emailing them value is null and later can be updated. “TicketStatus” has three values: New, Open, Awaiting Customer Response, Resolved, Closed. The default is New. Priority has five values: 0 - Low, 1- Normal, 2- High, 3 – Emergency. The ticket on delete from user interface does not delete, only the “IsDeleted” bit value is set to true, this is done so that if required the tickets can be inspected in the future.

###### **TicketCollaborator**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| CollaboratorID | int | Primary Key |
| TicketID | int | Foreign Key |
| EmailAddress | nvarchar(500) | Not Null |
| FullName | nvarchar(500) | Null |
| FieldType | nvarchar(500) | Null |

Table 2 - Ticket Collaborator

##### This table will store any Collaborator Details that Client request or Client send with e-mail. “CollaboratorID” is the primary key of table “TicketCollaborator” and auto incremental. “TicketID” is Foreign Key from “TicketMaster”. “FieldType” has following values: CC, From, Manual.

###### **TicketActivity**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| ActivityID | int | Primary Key |
| TicketID | int | Foreign Key |
| ActivityType | nvarchar(100) | Not Null |
| Summary | nvarchar(500) | Null |
| Description | nvarchar(500) | Null |
| ActivityDate | datetime | Not Null |
| ActedBy | int | Null |
| ActorName | nvarchar(500) | Null |
| Unread | bit | Null |

Table 3 - Ticket Activity

##### This table will store any activity done on Ticket. “ActivityID” is the primary key of table “TicketActivity” and is auto incremental. “TicketID” is Foreign Key from “TicketMaster”. “ActvityType” can have following value: Created, Updated, Deleted. “ActedBy” will have the “UserID” but if Ticket is created via email then it will be null and “ActorName” will be End User. “Unread” is a Boolean value set to keep track of unread activities.

###### **Staff**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| userid | int | Primary Key |
| username | nvarchar(50) | Not Null |
| password | nvarchar(50) | Not Null |
| isactive | bit | Not Null |

Table 5 - User

##### Staff table store all the staff member who can access the backend facility. “UserID” is the Primary key for User table and is foreign key for “TableMaster” table.

###### **Blocklist**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| Blocklistid | int | Primary Key |
| Email | nvarchar(500) | Not Null |

Table 6 - Ticket Blacklist

##### “BlocklistID” is the Primary key for this table and is auto incremental. This table will keep track of email who are not allowed to create ticket.

**Client**

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Data Type** | **Constraints** |
| Client\_ID | int | Primary Key |
| FullName | nvarchar(500) | Null |
| Gender | nvarchar(50) | Null |
| Email | nvarchar(200) | Unique Key Not Null |
| Password | nvarchar(500) | Not Null |
| StreetAddress1 | nvarchar(500) | Null |
| StreetAddress2 | nvarchar(500) | Null |
| City | nvarchar(500) | Null |
| State | nvarchar(500) | Null |
| Country | nvarchar(500) | Null |
| Zip | nvarchar(50) | Null |

Client\_ID is Primary Key and auto increment. Email and Password is Not Null. A client can create ticket by email or by creating an account. This table will store all details who create an account and access system via Web Portal.

**ABOUT THE PROJECT**

The zip will have four folders:

* ClientTicket
* StaffTicket
* MailService
* Screenshot

**NOTE** – Sometimes you might have to click a Button or Link Button two times (not double click) to launch its “on\_click” method. Also I had to replace some Link Button with Button on some places because they were unresponsive , don’t know why, but replacing them with simple Button worked.

**Screenshot**

This folder contains screenshots of all above working three project. This folder also contains SQL Scripts, to run in Microsoft SQL Server. I had Microsoft SQL Server 2008 R2 on my system.

**MailService**

It is a Windows Service. This Module is responsible for creating ticket via mail. The service will check every five minutes that it has received an email or not. Based on the subject it will decide to create a new one (insert in ticket master and create an activity of it) or append to an existing one (insert in ticket activity). If the Subject is in the form “Ticket:6 ; This is subject”, then it will extract the Ticket Id from subject using Regular Expression. If it does not find any Ticket ID, then it will create new ticket instead. It will also check whether the email has been blocked or not. It uses IMAPX for reading emails. It also checks for any collaborator in the email and add them ticket collaborator table. The company email id is “helpdeskcts6716@gmail.com” and password is “AP\_5@ukH”. For this functionality to work, we have to turn on the “Less Secure App Access” Permission on the email, its already been done.

**ClientTicket**

It is an ASP.NET Website. This Module is for Client Side. The default page is ClientLogin.aspx, the web config was updated for this. Used ADO.NET and Entity Framework to communicate with Database. The client can do following:

* Login (Email and Password) – Create a Session
* Register (Full Name, Email, Password, Confirm Password, Gender, StreetAddress1, StreetAdddress2, City, State, Country, Zip) – Insert record in Client Table
* Client Dashboard – View all tickets created on this account (Ticket ID, Email, Subject, Assigned To, Priority, Creation Date) – Populate Grid View with all Tickets from Ticket Master
* Create Ticket (Category, Subject, Description) – Insert record in ticket master and creating activity of the same.

##### Delete Ticket – Set the IsDeleted to true (or 1). The ticket on delete from user interface does not delete, only the “IsDeleted” bit value is set to true, this is done so that if required the tickets can be inspected in the future. Also create activity on the same.

* View all activities on a particular ticket – Populate List Views and Labels from ticket master and ticket activity
* Post Reply – Create an ticket activity of the reply posted
* Logout – Assign the Session Variable to NULL

**StaffTicket**

It is an ASP.NET Website. This Module is for Server side or Staff side. The default page is StaffLogin.aspx, the web config was updated for this. Used ADO.NET and Entity Framework to communicate with Database. The Staff can do following:

* Login (Email and Password) – Create a Session
* Register (Email, Password and Confirm Password) - Insert record in Staff Table
* Staff Dashboard – View All Created Tickets where IsDeleted is false (or 0). Populate Grid View with all Tickets from ticket master.
* Staff Dashboard – View All Created Tickets Assigned to them. Populate Grid View with all Tickets where “assign to” is equal to logged in staff
* Create Ticket (Full Name, Email, Ticket Source, Category, Subject, Description, and Priority) – Insert record in Ticket Master and also create activity for the same.
* Block list – View, Add and Delete emails to block list. Insert or Delete from Block list. Populate the grid view with block list table.
* View all activities on a particular ticket - Populate List Views and Labels from ticket master and ticket activity.
* Post Reply and send an email about the posted reply to the client - Create a ticket activity of the reply posted and also send email using SMTP, also get collaborator from ticket collaborator table (if any).
* Add Client to Block list – Insert client’s email to block list table.
* Delete Ticket - Set the IsDeleted to true (or 1). The ticket on delete from user interface does not delete, only the “IsDeleted” bit value is set to true, this is done so that if required the tickets can be inspected in the future. Also create activity on the same.
* Assign Ticket to them self – Create an activity for the same and update the ticket master table, “assign to” their staff id.
* Change the Ticket Status of a Ticket – Update in Ticket Master and create an activity of the same
* Change Priority of a Ticket – Update in Ticket Master and create an activity of the same
* Logout - Assign the Session Variable to NULL

**REFERENCES**

* <http://www.os-templates.com/> - Website Design
* <https://stackoverflow.com/questions/11369889/linq-how-to-get-values-from-var?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> - LINQ
* <https://stackoverflow.com/questions/6799120/script-all-data-from-sql-server-database?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> – SQL Script
* <https://msdn.microsoft.com/en-us/library/system.threading.threadabortexception(v=vs.110).aspx> – Thread Error
* <https://www.daniweb.com/programming/software-development/threads/500431/best-way-to-remove-entity-framework-from-a-project> - Entity Framework
* <http://net-informations.com/q/faq/stringdate.html>
* <https://forums.asp.net/t/1452154.aspx?Date+format+in+LINQ> – Date Format
* <https://www.lifewire.com/unique-constraints-in-microsoft-sql-server-1019841> - Unique Constraints
* <https://www.codeproject.com/Questions/847705/Sending-email-using-Gmail> - Send Mail
* <https://stackoverflow.com/questions/23484503/how-to-sending-email-to-multiple-recipients-with-mailmessage?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> – Send mail to multiple recipient for collaborator
* <https://msdn.microsoft.com/en-us/library/system.net.mail.smtpclient(v=vs.110).aspx> – SMTP Mail
* <https://msdn.microsoft.com/en-us/library/f02979c7(v=vs.110).aspx>
* <https://stackoverflow.com/questions/22672368/how-to-make-a-transparent-html-button?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> – CSS
* <https://stackoverflow.com/questions/16658803/insufficient-stack-to-continue-executing-the-program-safely-asp-net-mvc-4?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> – Insufficient Stack Error
* <https://stackoverflow.com/questions/18002744/how-to-set-default-page-asp-net?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa> - Set a web form as default page