**Project :- Customer Query Track Domain :- CRM**

**Technologies:- .Net(ASP.Net Core,Linq,MVC,C#,WCF, WebAPI(Rest Services) )**

<https://www.youtube.com/watch?v=7rcCfCdiB9A>

**Description**: - This project is mainly used to track the queries of the customers in the organization andresolve the queries. This Project also sends the services to the customer to find out the customer satisfaction and employee performance.

The system has a major problem in finding the product support and service activities that are undertaken by the employees. Once the products have installed at the client's site, the cycle of warranty maintenance starts. To keep the customers in a proper chain of satisfaction the system needs the data to be organized in a proper re lational way.

The project is to maintain the relational communication among the project is to maintain this relational communication among the data, such that consistency, reliability and integrity can be achieved. The major technicality within the system lies in managing the coordination among the sales and service departments, which is the major hold upon the system to gain or achieve profits.

**NUMBER OF MODULES**

The system after careful analysis has been identified to be presented with the following modules:

**The Modules Involved in the project are:**

1. Customer’s Module

2. Employee’s Module

3. Product Admin’s Module

**1. Customer’s Module**:

The Customer’s Module is designed to provide the following services.

i)Posting a Query:

* Customers should be free to post their queries to the concern department.
* The system must identify each query with an automatically generated Query Id, for future references.

ii) Survey Details:

* Customers should be able to post the survey details on a department by simply selecting the department and post the survey details to the organization.
* In survey details service the system provides the rate of satisfaction of the customer on a particular employee.
* The rate of satisfaction can be known by giving ratings to the employee.

iii) Status

* Customers should be allowed to check the status of their query, by using the unique Query Id provided by the system.
* The system must display the answers along with the Query to the customer, after posting the query.

**2. Employee’s Module:**

In this Employee’s Module we have two services, which are

i). Posted Queries:

* The employees should be able to see the posted queries, which are posted by the customers.
* An employee can visit the queries posted to that department only.
* The employee should post back the solution to the queries he has received.
* An employee cannot only see his rating but also rating of other employees who are working in other departments.

ii). Survey:

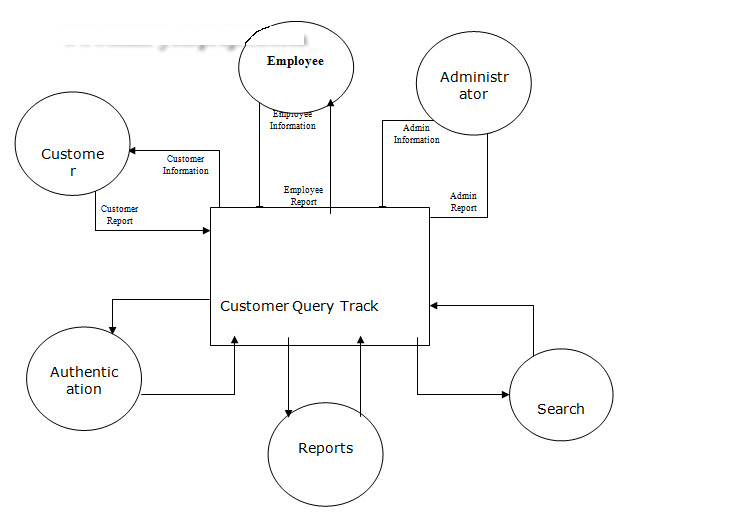
* The employees should be able to see the survey details posted by the customers.
* He can able to see the survey details of all the departments.
* The customers should be able to see the frequently asked question asked by the customers.

A password protection with secure mode option is provided to guard from un-authorized access to database.

**PROJECT INSTRUCTIONS:**

Based on the given requirements, conceptualize the Solution Architecture. Develop the application for ultimatedotnet.com. Depict the various architectural components, show interactions and connectedness and show internal and external elements. Design the web services, web methods and database infrastructure needed both and client and server.

Provide an environment for upgradation of application for newer versions that are available in the same domain as web service target.



|  |  |
| --- | --- |
| Customer | Employee |
| Login – Authorized | Login – Authorized |
| * Customer Details | * Employee Details |
| Customer purchases a product & product installed – Warranty date begins |  |
| **Post a Query** | **Queries** |
| * Auto generated Query ID should be provided for future reference. | * Employee can see the list of queries posted by customers to their specific dept only. |
|  | * Emp sh post back to queries he receives. |
|  | * Emp can see ratings of him & emps in other dept’s too. |
| **Survey** | **Survey** |
| * Selecting Dept, cust sh be able to submit survey/feedback to org. | * Can see survey details of all depts’. |
| * Cust can provide feedback for specific employee. |  |
| * Ratings will be given to Employee |  |
| **Status of Query** | **Status of Query** |
| * Using query ID can know the status of query | * Status sh be changed by Emp On responding to a query. |
| * If answered, ans should be displayed along the query. |  |
| **FAQ** |  |
| * Cust can see the FAQ asked by customers |  |

**Front-End – Angular 2/6**

Customer – Login

Employee – Regular Emp & Admin Logins,

**Back-End – ASP.NET CORE MVC**

Models –

Controllers –

Tables:

1. CustomerMaster
   * CustomerId
   * FirstName
   * LastName
   * Email
   * UserName
   * Password
2. EmployeeMaster
   * EmpId
   * FirstName
   * LastName
   * Email
   * UserName
   * Password
   * EmpRoleId
   * EmpAvgRatings
3. Survey
   * SurveyId
   * CustomerId
   * DeptId
   * EmployeeId
   * RateLevel
4. ProductMaster
   * ProductId
   * Description
   * Company
5. WarrantyMaster
   * CustomerId
   * ProductId
   * WarrantyStartDate
   * WarrantyDueDate
6. QueryMaster
   * QueryId
   * CustomerId
   * ProductId
   * Message
   * QueryDate
   * Status
7. QueryAssigns
   * QueryId
   * CustomerId
   * EmployeeId
   * ResponseDate
   * Message
8. EmpRoles
   * RoleId
   * Role
9. Departments
   * DeptId
   * DeptName
   * DeptAvgRatings

**Completed:**

1. Add/Edit/Delete Employee Front & Back End
2. Add/Edit Customer Front & Back End with Web API

**Define Predicate & Pass**

**To-Do**

1. Survey
   1. Customer -> Add Survey to Emp of Dept
   2. Employee-> See his Survey
   3. Employee -> See Survey of all Dept
2. Login
   1. Customer Login
   2. Employee Login
      1. As Admin or regular
3. Query
   1. Customer -> Get/View all UnResolved queries Front & Back End with Web API
   2. Customer -> Get/View Recent Resolved Queries Front & Back End with Web API
   3. Customer -> Post Query (Master) Front & Back End with Web API
   4. Customer -> See all responses on posted Query Front & Back End with Web API
   5. Employee -> Receive Query of his Dept Front & Back End with Web API
   6. Employee -> Response Query to Customer Front & Back End with Web API
4. Product
   1. Get All Employees Front & Back End with Web API
5. ss

**To-Do Improvements:**

1. Notify user (alert, timed dlg) on every action – post/update/etc.
2. QueryService – Get Recent Resolved – Date criteria

Compile time Dependency

ASP.NET Core Application Run time Dependency

Controllers

Model Validation Filters

Response Cache Filters

EF Core Context

Domain Entities

Data Migrations

Interfaces

ASP.NET Core Identity

Views

Infrastructure Project

In Memory Data Cache

Web API (With Controller)

Email Service

View Models

Domain Class Library’s

Application Exceptions

{ }

SQL

Domain Events

Service Implementation

Links

1. Add/Scaffold Login UI - <https://www.c-sharpcorner.com/article/how-to-implement-authentication-using-identity-model-in-asp-net-core/>
2. Add Customize Identity - <https://www.youtube.com/watch?v=tW5QwWPKXaY>
3. Create Custom Roles - <https://gooroo.io/GoorooTHINK/Article/17333/Custom-user-roles-and-rolebased-authorization-in-ASPNET-core/32835#.XNhGSY5KjIU>
4. Use Identity on existing DB - <https://www.youtube.com/watch?v=elfqejow5hM>
5. Add Identity to existing MVC - <https://timmydinheing.com/2018/04/11/adding-asp-net-identity-2-0-into-existing-mvc-project/>
6. **Modern Web Apps Architecture** - <https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/architectural-principles>
7. **Advanced Architecture for ASP.NET Core Web Apps** - <https://www.infoq.com/articles/advanced-architecture-aspnet-core>
8. Access Web API thru jQuery & Javascript - <https://www.c-sharpcorner.com/UploadFile/4d9083/how-to-create-web-api-in-Asp-Net-mvc/>
9. How to use Web API in Web App ASP.NET Coe – **Good one** <https://www.youtube.com/watch?v=wYMdqhid4Fs>
   1. Implementing now – created WebAPI. Need to implement in Web App ( 8:35 time)
10. Tutorial for Web API with MVC **(Detailed)** <https://www.youtube.com/watch?v=e2qZvabmSvo>
    1. JWT Auth from 35 min
11. Working with Web APIs
    1. Attribute Routing - <https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/attribute-routing-in-web-api-2>
    2. REST API - <https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/create-a-rest-api-with-attribute-routing>
    3. Create Web aPI - <https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-2.1>
    4. Routing - <https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-2.2>
12. JWT Auth in Core - <https://www.youtube.com/watch?v=7tgLuJ__ZKU> **NEED TO WATCH**
13. <https://www.youtube.com/watch?v=RQyGFjrjy6Y>
14. Ajax POSt - <https://www.youtube.com/watch?v=HuniMNaWP6U>
15. Bootstrap Modal - <https://getbootstrap.com/docs/4.0/components/modal/>
16. Call partial Views - <https://www.google.com/search?rlz=1C1GCEA_enUS771US771&q=how+to+call+partial+view+in+mvc&sa=X&ved=2ahUKEwi5goz1zabiAhVIHjQIHZQyB1IQ1QIoA3oECAoQBA&biw=1029&bih=554> “how to call partial view in mvc”
17. ASP.net Core ajax modal - <https://softdevpractice.com/blog/asp-net-core-mvc-ajax-modals/> NEED TO IMPLEMENT