

National College of Ireland
BSc in Computing Evening
2016/2017

Stephen Tyrrell
X134516925
Stephen.Tyrrell@student.ncirl.ie

SalesBoost

Microsoft Dynamics CRM and Salesperson Portal

Technical Report

Supervisor: Dr. Catherine Mulwa



Contents

Project Declaration

Executive Summary	5
1. Introduction	6
1.1 Background	6
1.2 Aims.....	7
1.3 Technologies	7
1.4 Strategies	8
2. SalesBoost Implementation.....	9
2.1 Requirements Specifications	9
2.2.1 Functional requirements for Microsoft Dynamics CRM	9
2.2.2 Functional requirements for the portal.....	12
2.2.2 Data Requirements	14
2.2.3 User Requirements	14
2.2.4 User Stories	14
2.2.5 Use Case Diagram.....	16
2.2.5 Environmental Requirements	16
2.2.6 Usability Requirements	16
2.2 Architecture Design	17
2.3 Implementation	20
2.3.1 Technical Approach	20
2.3.2 Graphical User Interface (GUI).....	28
2.4 Testing and Evaluation	30
2.4.1 System Function Testing.....	30
2.4.2 User Interface Evaluation.....	42
2.4.3 Customer testing.....	45

2.4 Conclusion	45
3 Conclusion and Future Work.....	47
Further development or research	48
References.....	50
Appendix 1 - Project Plan	53
Microsoft Dynamics CRM Salesperson Portal.....	53
1. Objectives	54
2. Background.....	54
7. Technical Approach	55
Technologies	55
8. Project Plan.....	56
9. Technical Details.....	56
10. Evaluation.....	57
Project Plan	57
Appendix 2 - Monthly Journals	58
Appendix 3 – Other Materials Used	62

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name:
Student ID:
Supervisor:

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: _____

Date: _____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Executive Summary

This project is based around the installation, configuration and deployment of a Microsoft Dynamics CRM On Premise Environment, Microsoft's Customer Relationship Management software that is widely used by business for Sales, Marketing and Service purposes. This project also addresses the issue of portability and usability of Microsoft Dynamics CRM as a sales/opportunity tracking system for Relationship Managers and other Sales based employees, by implementing a 'Portal Application'. The Microsoft Dynamics CRM web client is resource-heavy and quite cluttered, meaning low-powered devices can lead to slow loading times and crashing due to the JavaScript loads and heavy HTML forms. Microsoft's options for portability (Dynamics CRM for Mobile/Tablet) are also considerably poor ports of the system to a mobile platform.

I plan to address this by creating an external portal, using a hosted ASP.Net application using web forms, hosted in Azure, that will also be ported for mobile devices using Xamarin.

For the CRM environment, I will be deploying a Microsoft CRM 2015 On Premise environment. For the purposes of the project, the deployment's authentication will be done through the user's Active Directory credentials. In this instance, the user's credentials/profiles are stored and managed on an external AD server, using CodeTwo as the Active Directory management software.

The basis of the portal will be created around the ASP.Net web application, connected to a Microsoft Dynamics CRM 2015 organisation, which will be out of the box configuration. Authentication will be done through Form based authentication.

Integration between the two systems will be done through a combination of the Microsoft CRM Developer Extensions, and the Microsoft Dynamics CRM API endpoint.

1. Introduction

1.1 Background

The background to this project comes from my personal experience in working with and implementing Microsoft Dynamics CRM solutions for customers through my job as a CRM Developer.

Microsoft Dynamics CRM is one of the market's leading Customer Relationship Management systems, and is used by companies of all sizes, across the globe. It is renowned for its sales processing capabilities, for tracking leads and opportunities as well as a marketing tool which can be integrated with several 3rd party systems such as HubSpot and EventBrite.

However, typically the web interface for Dynamics CRM is bulky and slow, due to the amount of functionality and resources used for the different parts of the system (Marketing, Project, Service, Sales etc.), and frequently we see concerns raised by users in regards to slow loading times, browsers crashing on low powered machines due to the heavy client side scripting on the pages due to JavaScript functionality. To combat this, my project also looks to deploy an ASP.Net application that serves as a 'portal' to allow Sales reps and other users to access their CRM records and create new records for just the Sales area of the system, particularly on lower powered systems.

My experience with implementing solutions using a combination of ASP.Net applications using Web Forms and deploying within IIS from personal projects and at a business level will benefit me greatly I feel, as I'm aware of the challenges and strategies that have to be undertaken in order to complete this project correctly and on time with my project plan. This will also be the first time I've implemented a solution such as this, which I feel will be a great learning curve to push my development skills as well as my personal experience with the system, as implementing the full application lifestyle and then extending beyond the system's capability/function as standard.

1.2 Aims

The aim of this project is to implement a Microsoft Dynamics CRM on premise deployment from start to finish, including the configuration and deployment of the application. This includes configuring the servers and databases, configuring the Active Directory service that will allow users to login using their Active Directory credentials that they use to login to their machines and other services such as Office 365 for Skype and Outlook, configuring the IIS website that the CRM environment will connect to and sit on, deploying the CRM software on the main server and also configuring the environment for the users to be able to use the system to its full capability and it's relating services such as asynchronous workflows and business process flows to implement the way how the users will deal with working in the system.

On top of this CRM environment, I will be developing a custom, light-weight interface for sales people to be able to access, create and update records for CRM. The idea behind it being light-weight means that it can be run across many devices, including tablets and mobile devices without being strenuous on resources. This mobility factor is a key aspect to how sales people would endear to use CRM, when they are on a customer site or on the road.

1.3 Technologies

The Microsoft Dynamics CRM infrastructure consists of the Application itself which will be deployed on a standalone Application Server, and will be available through an IIS website. It will communicate with a SQL server which will store the database and allow for read and write functions. There will also be an AD server to store user profiles and credentials.

The project also consists of a 'portal' which is an ASP.NET web application using Web Forms, that will be hosted within azure, that connects to a Microsoft Dynamics CRM organisation. As mentioned before, all the front-end components will be Web Forms, and the overall application will be hosted in Azure. The project will be implemented using C#, and the Dynamics CRM organisation that will be used will be standard out of the box functionality, bar some minor adjustments to the contact form to allow for user authentication.

For the mobile port, Xamarin will be used to transform the completed Web Application into a mobile friendly version that will be able to run locally. This will primarily be focused on Android devices due to Android being a common choice amongst businesses, however Windows and iOS ports will also be considering depending on user feedback and time constraints.

1.4 Strategies

The project will be developed using an Agile development methodology, implementing the project incrementally.

The first stage of the project will involve the installation and configuration of the servers. They will be hosted VMs through Microsoft Hyper V. Once the servers, databases and services are fully configured and running, the CRM instance will then be installed on the application server, and deployed to the IIS website. Once up and running, the CRM environment will then be configured to the specifications/requirements need.

The first stage of development for the portal will be configuring the connection to CRM, then generating the base classes to pull in the CRM entity fields and then moving on to the authentication factor. From there the next step will be moving on to the entity definitions and relationships. Once those are defined, Web Forms will then be created to allow for the user to input/edit/view records. This will be the final point of the initial development.

The testing process will be rather rigorous and long, with stress testing, load testing, performance testing and functional testing for both applications, as they will both be test as separate entities. Testing of the infrastructure in regards to not only penetration testing to ensure the security of the system will take place, as well as testing of the failover procedures will also be done. Once the all taken place first, and after any revisions or bug fixes, will move on to User Acceptance Testing, where end users will get access to the system to test and run through the system. Any bug fixes or changes will be made, and then put through another round of User Acceptance Testing until the project is completely signed off on by the end users in regards to usability and design.

2. SalesBoost Implementation

2.1 Requirements Specifications

The requirements of the system are defined in the below section of the document. This will cover the functional requirements gathered at a high level to verify the functionality of the application, user stories which will walk through the processes and functions from a user perspective that they will need to complete as day to day tasks, as well as non-functional requirements.

For the Dynamics CRM application, the requirements we will concentrate on will revolve around the Sales element, in particular the Opportunity and Lead entities as this is the intended use of the system for the purpose of the project. The requirements will consist of the functionality that the end users.

In regards to the portal, due to the simplistic GUI, and the fact customers/users will have experience using Dynamics CRM and logging opportunities, the system will be user friendly and a minimal learning curve. Little training will be needed to upskill users, as it's merely a simplified extension of the existing Dynamics CRM client.

2.2.1 Functional requirements for Microsoft Dynamics CRM

REQUIREMENT 1 <LOGIN>

Description

User can log in to their account

Flow Description

Precondition

User has an existing Active Directory account

Main flow

1. User enters the site address in their browser address bar
2. User is prompted with a login page
3. User fills in login criteria
4. User clicks submit

Post condition

User is logged into their account and taken to homepage

REQUIREMENT 2 <Create Accounts>

Description

User can create Account records to store the relevant account information for customers

Flow Description

Precondition

User has an existing profile with the relevant permissions to create accounts

Main flow

1. User logs in to site and is authenticated
2. User navigates to the account entity
3. User selects 'Create new Account'
4. User inputs relevant account information including mandatory fields
5. Users saves account
6. Information is verified and account is created.

Post condition

Account has been created and records can now be associated to this account.

REQUIREMENT 3 <Create Leads>

Description

User can create Leads records to track potential new sales and customers

Flow Description

Precondition

User has an existing profile with the relevant permissions to create leads

Main flow

1. User logs in to site and is authenticated
2. User navigates to the lead entity
3. User selects 'Create new Lead'
4. User inputs relevant lead information including mandatory fields
5. Users saves lead
6. Information is verified and lead is created.

Post condition

Lead has been created and the user can now track the progress of this lead.

REQUIREMENT 4 <Create Opportunity>

Description

User can create Opportunity records to track potential new sales to existing customers.

Flow Description

Precondition

User has an existing profile with the relevant permissions to create opportunity

Main flow

1. User logs in to site and is authenticated
2. User navigates to the opportunity entity
3. User selects 'Create new Opportunity'
4. User inputs relevant opportunity information including mandatory fields
5. Users saves opportunity
6. Information is verified and opportunity is created.

Post condition

Opportunity has been created and the user can now track the progress of this opportunity and the income that it will generate can be used for reporting.

2.2.2 Functional requirements for the portal

REQUIREMENT 1 <LOGIN>

Description

User can log in to their account

Flow Description

Precondition

User has an existing account

Main flow

1. User navigates to site
2. User clicks login button
3. User fills in login criteria
4. User clicks submit

Post condition

User is logged into their account and taken to homepage

REQUIREMENT 2 <CREATE RECORDS>

Description

Users will be able to create opportunities for their customers through the portal

Precondition

User has an existing account with permissions to create records

Main flow

1. User logs in to site and is authenticated
2. User clicks create new record button
3. User fills in fields with valid information
4. User clicks submit

Post condition

User's record is created and a prompt is shown to confirm.

REQUIREMENT 3 < Browse records>

Description

Users will be able to view their open opportunities and closed opportunities that they have created.

Flow Description

Precondition

User has an existing account with permissions to create records

Main flow

1. User logs in to site and is authenticated
2. User is redirected to the homepage which is set to their Open Opportunities
3. User can then navigate to their Won Opportunities

Post condition

User has access to view their records.

REQUIREMENT 4 <UPDATE RECORDS>

Description

Users will be able to update existing opportunities for their customers through the portal Flow Description

Precondition

User has an existing account with permissions to create records

Main flow

1. User logs in to site and is authenticated
2. User clicks update record button
3. User fills in fields with valid information
4. User clicks submit

Post condition

User's record is updated and a prompt is shown to confirm.

2.2.2 Data Requirements

A SQL database connected to the Dynamics CRM environment is where all of the data/records will be stored and used for read/write operations. Although the application will not interact directly with the database, the Dynamics CRM environment is used as the slave to perform these operations.

2.2.3 User Requirements

The user has to have a local Active Directory account, and be a valid user within Dynamics CRM, with the relevant permissions for their role i.e. permissions to view and create Opportunities.

2.2.4 User Stories

2.2.4.1 Sales Rep

John is a sales representative for Test Corporation. John's primary job role is to capture new contacts or leads which he can then turn into sales opportunities.

John gets a tip from a colleague of a new business in town looking to implement a new software system to replace the legacy one they are using. John enters the basic account information that he finds online about the business into CRM such as the account name and address, and the main phone number, into a new account form and creates the account.

John then gets the contact details for a point of contact within the company from his friend, and creates a lead within CRM to be able to track the progress. As John receives more information about the potential lead, he updates CRM with details such as their existing systems and what they require.

John contacts his lead, and they agree to engage in a formal manner of a proposal for a new solution to be implemented to be delivered by Test Corporation. John then transitions the Lead to an Opportunity, and inputs further information such as the proposed solution for the customer, their proposed budget that they have given as an indicator of what they want to spend on the new system, and when they expect to purchase the system. As the process progresses through time, John updates the opportunity to reflect the status of the customer, such as if they are getting closer to make the purchase, or if there are any changes to the solution being provided or

they're budget i.e. if they decide to invest more money into the system or financial needs mean they may need to decrease the budget allocated to it.

Towards the end of the process, if the company decides to purchase the system, John can then mark the opportunity as won, and his GP is calculated from the selling price and notional cost of the product that was provided. If the company decides not to invest in the solution that John has proposed, he can then close the opportunity as lost and it will be noted as lost potential revenue.

2.2.4.2 Sales Manager

Julie is a Sales Manager at Test Corporation, where she manages a team of sales people. For Julie to perform her job, she needs to be able to see the figures that her team are currently hitting from a team and also from an individual perspective. This way Julie is able to track their progress and see if they are hitting their projected figures for the period, as well as other goals that may have been set for them.

In order to do this, Julie has been given a custom built dashboard, where she can track the profit made by each sales person and by the team for the month, the number of open opportunities for each person and by team, and also the number of won opportunities for the financial period.

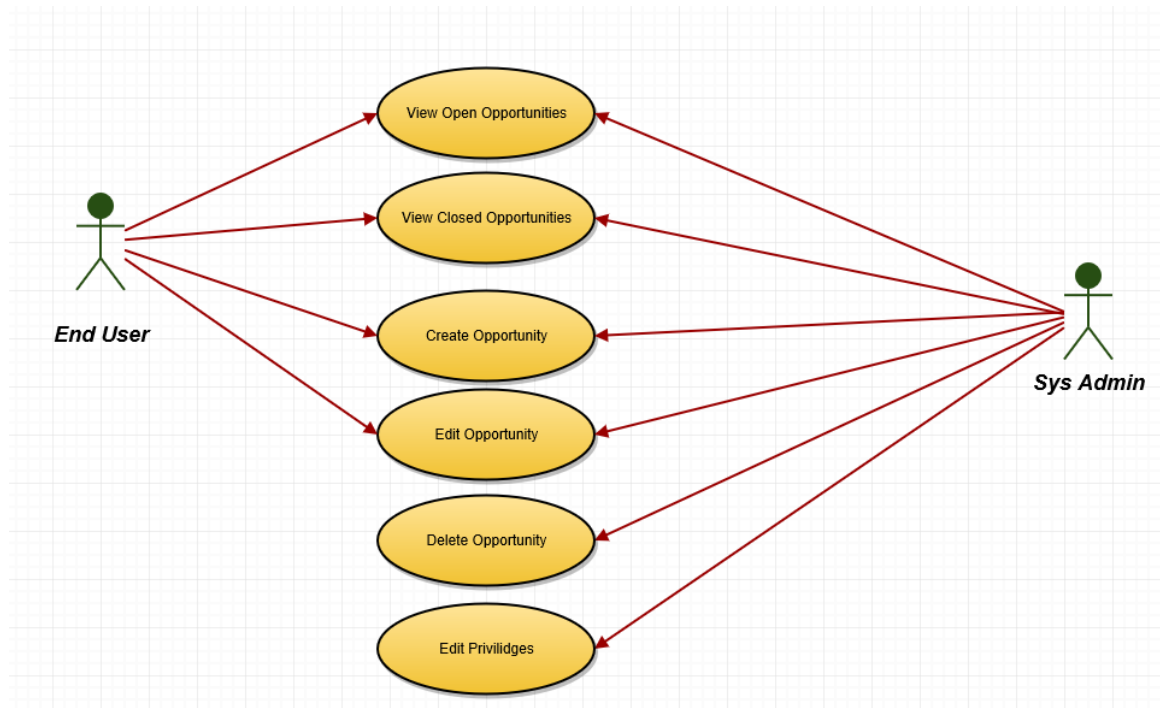
As well as this, Julie can also see the revenue by customer, allowing her to indicate the key accounts within the business.

Julie is also sent an email notification any time an opportunity is closed as won, containing the sales person, the amount the opportunity was worth and the customer.

Similarly, when an opportunity is marked as lost, Julie is sent an email notification containing the sales person, the customer the opportunity was for, and the reason why the opportunity was closed as lost.

These email alerts can assist Julie in staying on top of her staff's work, allowing her to see when they are successful at their jobs and also when they are wasteful and are not capitalising on opportunities, and she can then assist them if it is a regular occasion, or possibly even step in to try and save the opportunity if it is an important account.

2.2.5 Use Case Diagram



2.2.5 Environmental Requirements

The portal will be hosted in the cloud using Microsoft Azure, connected to an on premise deployment of Microsoft Dynamics CRM 2015, which will be out of the box to show the flexibility and versatility of the portal for reuse by other businesses/organisations.

In regards to user environment, the requirement is that users have a device, such as a phone/tablet/computer with a stable internet connection that allows them to open the application within a web browser.

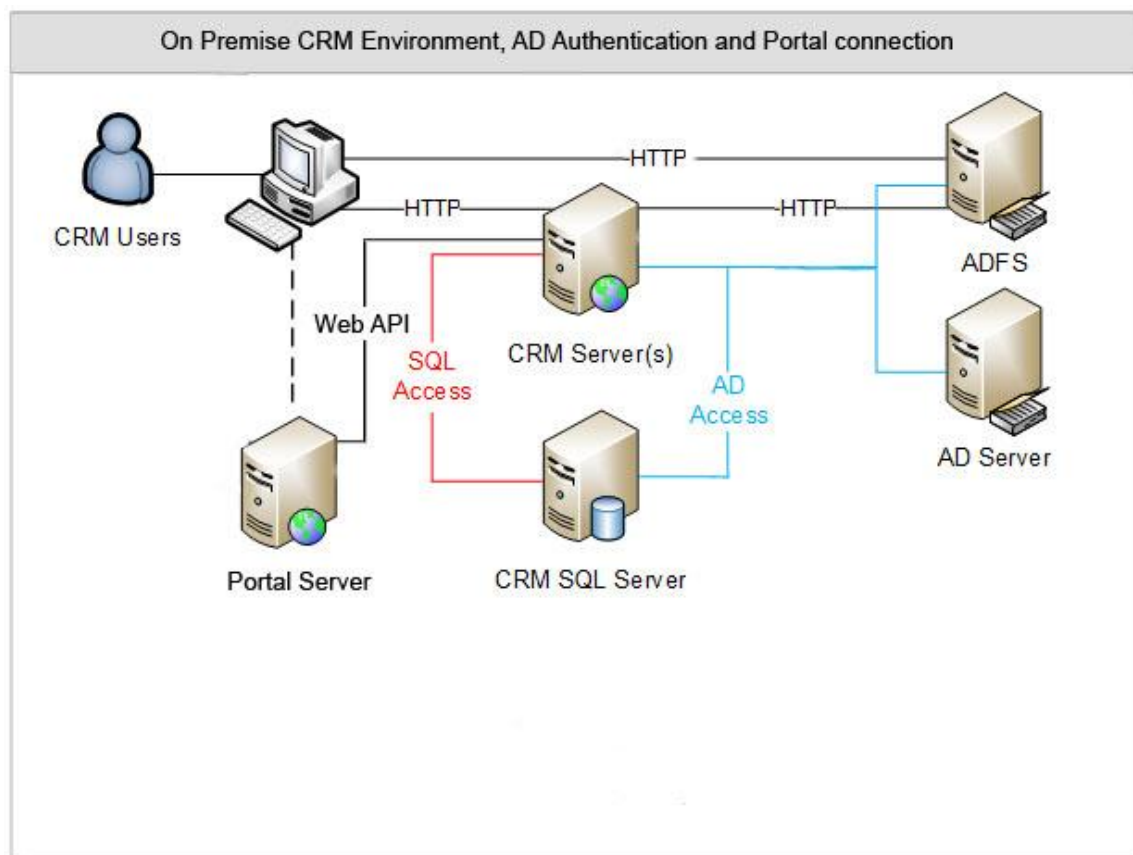
2.2.6 Usability Requirements

As previously discussed, the portal will be light weight client, which will be ported for mobile/tablet also, allowing users to utilise it across all devices. Usability is one of the

main factors of the project, as the aim of the project is to provide an easier/less strenuous way for users to access/create their records without having to access Dynamics CRM through a web browser on a computer, and offers the option to be able to do this on a mobile device or a computer, and provide a simplistic and much cleaner option to allow for reduced time spent entering/accessing records.

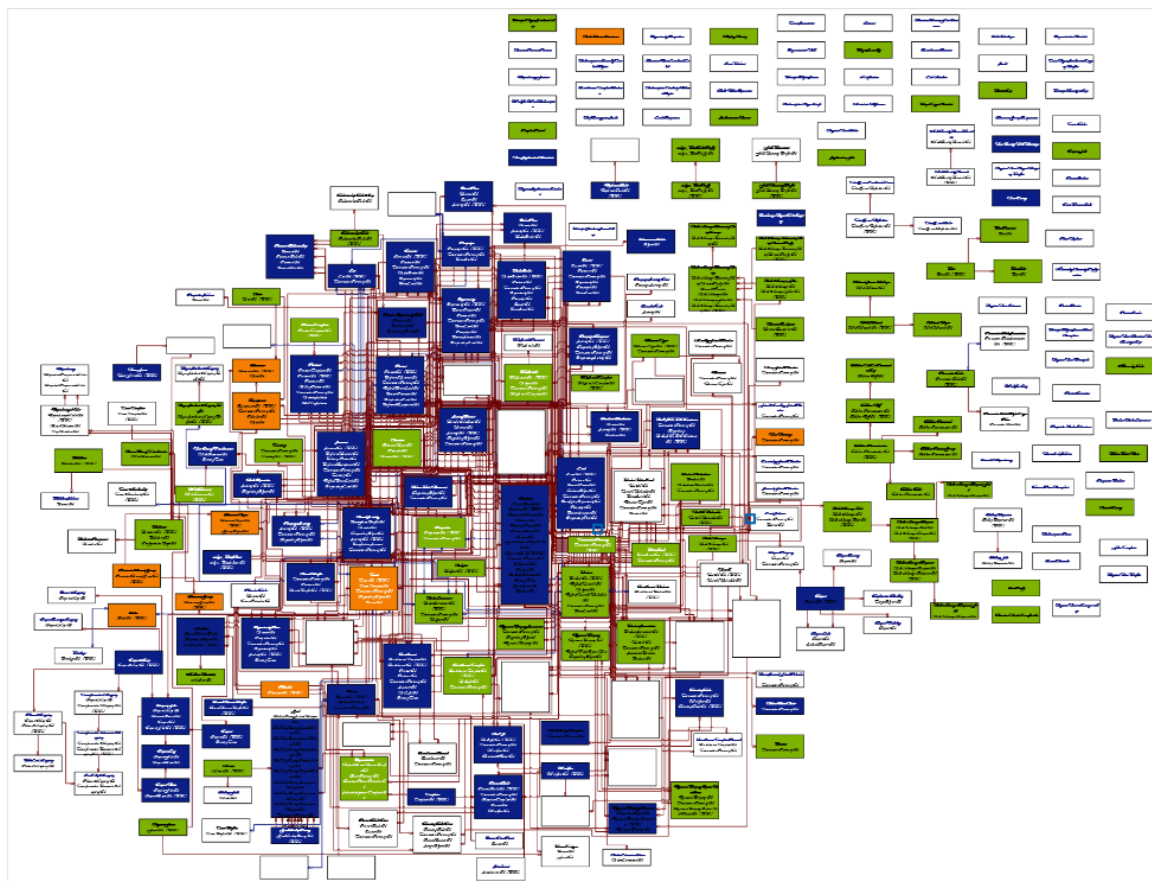
2.2 Architecture Design

The system architecture for the Microsoft Dynamics CRM on premise environment consists of 5 separate Hyper V VMs running Windows Server 2012r, that all sit within the same cluster and on the same network. Communication between the servers is done using internal network calls, using HTTP protocols. The decision to use HTTP and not make any of the servers accessible from external sources revolved around security concerns. Making the network closed off from the outside world outside of the internal network means that unwanted connections are not feasible unless they gain access to the internal network, which means that the data is secure and meets necessary security requirements for large corporations and other industries where data compliance is an important measure, such as childcare or government bodies.



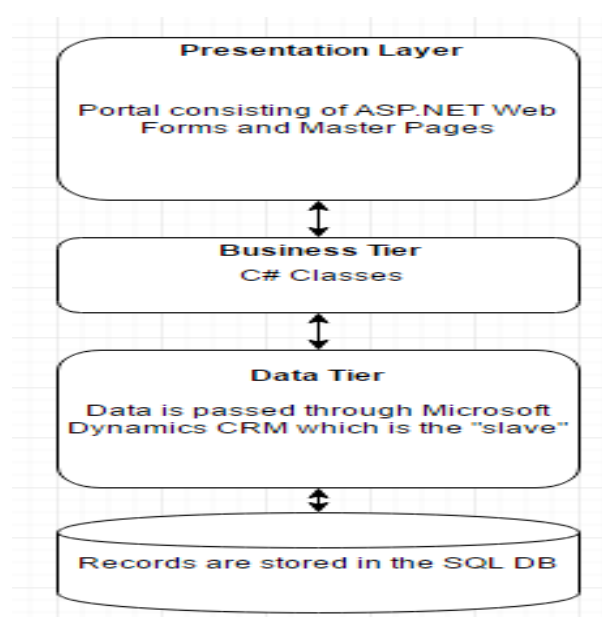
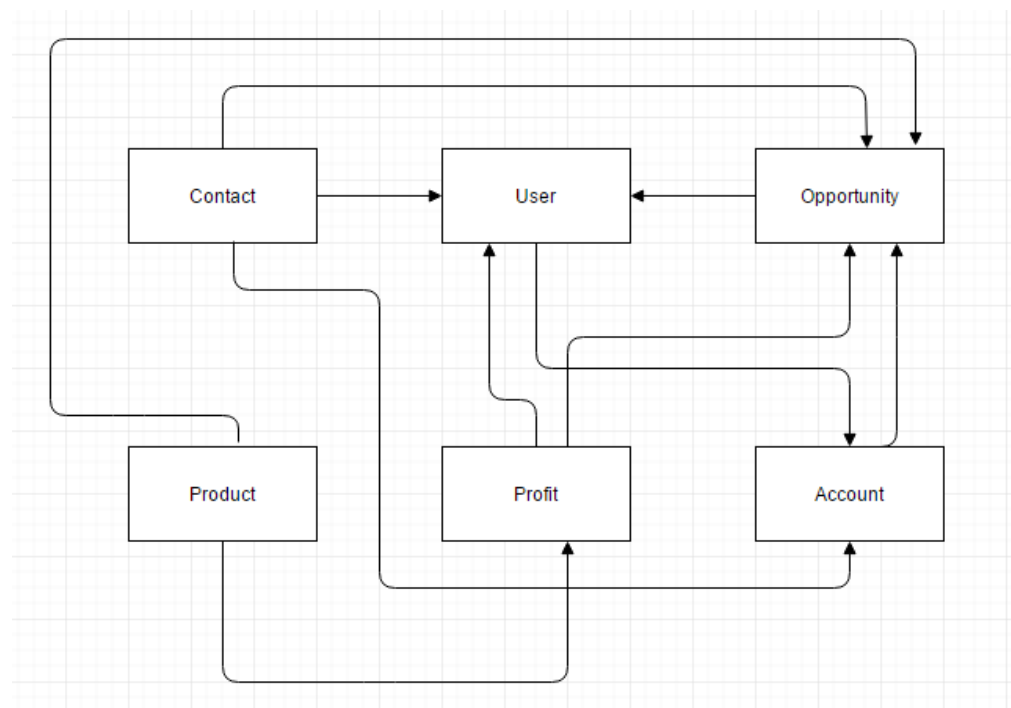
As seen in the above diagram, the end users client (be it a phone/tablet/pc) connects to the ADFS server to authenticate the user, which in turn communicates with the Active Directory server to verify the user account, and then the calls or actions are made to the CRM server. The CRM server communicates with the SQL server for read/write access for data, which is also authenticated by Active Directory.

Moving on to the design of the CRM environment, as you can see in the below Entity Model Diagram, the system is quite complex in regards to the amount of entities within the system, and particularly the communication between each entity. The system as a whole promotes communication and collaboration between entities, which allows for improved workflow and process.



For the portal, the system design consists of a four tier architecture. The Portal itself is the front end, the presentation layer consisting of ASP.Net Web Forms. Using C# classes which contain connection strings and parameters to pass data, the portal passes the data through these class to the Data Tier, consisting of the Microsoft Dynamics CRM environment, which acts like a “slave” to store the records in the SQL database.

Using the Dynamics CRM environment as a slave means all transactions from the portal to the database will pass through CRM first.



2.3 Implementation

2.3.1 Technical Approach

For the first step of implementation, I took about creating each of the individual Virtual Machines in Hyper V that would consist of the projects infrastructure. All servers were configured to the specifications listed on the Microsoft Website, which the services listed below:

Microsoft Dynamics CRM (on-premises) versions require the following software:

- Microsoft Windows Server
- A Microsoft Windows Server Active Directory infrastructure
- An Internet Information Services (IIS) website
- Microsoft SQL Server 2012
- Microsoft SQL Server 2012 Reporting Services

The following table lists the minimum and recommended hardware requirements for Microsoft Dynamics CRM Server 2015 running in a Full Server configuration. These requirements assume that additional components such as Microsoft SQL Server, Microsoft SQL Server Reporting Services, Microsoft SharePoint, or Microsoft Exchange Server aren't installed or running on the system.

Component	*Minimum	*Recommended
Processor	x64 architecture or compatible dual-core 1.5 GHz processor	Quad-core x64 architecture 2 GHz CPU or higher such as AMD Opteron or Intel Xeon systems
Memory	2-GB RAM	8-GB RAM or more
Hard disk	10 GB of available hard disk space	40 GB or more of available hard disk space

The first step of configuring a CRM deployment is installing Active Directory onto the CRM Server that I planned. After installing Active Directory Domain Services through the server roles and features, and configuring the domain to the domain the existing

AD server that the organisation uses, the next step was then to configure the SQL server. For the purpose of this project I chose to use SQL Server 2014, as I was more familiar with it than the latest MS release, SQL Server 2016. After installing SQL Server 2014, we set up our base Database, that will hold all of our CRM records. For the purpose of this project we will use SQL Server's out of the box backup service to run a daily backup of the database. However there is no failover in place, as both databases will co-exist on the same SQL server, due to the lack of resources available to myself at this time. In an ideal situation, using a KEMP load balancer across two independent servers on independent networks would be preferred.

Once the SQL server is configured and the databases are configured and in place, we then move onto configuring the CRM Server itself.

There are a number of different features required on the CRM server. Microsoft's CRM Deployment Administration Tool, Internet Information Services and Microsoft's Asynchronous Processing Service are the main tools/services needed to configure the server and deploy the environment.

First I configure the IIS website, so that the system has a front end and is accessible over the internet, by dedicating a domain and port. In this instance the deployment will not be external facing as I do not have access to the necessary external HTTPS certificates and also due to the lack of security on the infrastructure that is in place. Without a secure infrastructure, making the environment externally facing and accessible to the wider internet, which could lead to security breaches or leaks unless the environment is secure.

Once the full environment was deployed and up and running, I could then move onto the configuration element. As this project was mainly going to concentrate on the Sales part of the system (as this ties in with the portal), I chose to spend the majority of the effort configuring this element as opposed to concentrating on the marketing and service aspects.

First step was to configure the Accounts. Accounts are customers, where all of the customer's information such as contact details, addresses and preferences are stored. The reason why we use accounts is that all of the leads/opportunities can be tied back to the account, which is important for tracking and reporting purposes, as it gives the option to do breakdown by account and also prevents any overlap or duplicates.

Another element for the Account that was configured, was Bing Maps. Using the CRM End Point and the Bing Maps API, we are able to display the Account's listed address on a Bing Maps iframe on the account form.

The screenshot shows the Microsoft Dynamics CRM interface for an Account entity. The breadcrumb trail is 'SALES > Accounts > Test Account'. The left sidebar shows the 'ACCOUNT' entity selected. The main content area is divided into several sections:

- Summary:** Displays 'Test Account' with a profile picture icon.
- ACCOUNT INFORMATION:** Fields include Account Name (Test Account), Phone (01 02202222), Fax, Website (http://test.ie), Parent Account, and Ticker Symbol.
- ADDRESS:** Field 1 shows 'Test Street', 'Test, Test', and 'Test'. Below is a Bing Maps iframe showing a location in Ireland.
- POSTS, ACTIVITIES, NOTES:** A section for creating and viewing posts, activities, and notes. A post is visible: 'Test Account Account Created By Stephen Tyrrell. On Test Account's wall Today'.
- Primary Contact:** 'Test Contact' with email 'test@test.ie' and business phone '01123456'.
- CONTACTS:** A table with columns 'Full Name' and 'Email'. It shows 'No Contact records found.'
- RECENT OPPORTUNITIES:** A table with columns 'Topic', 'Status', 'Actual Close Date', and 'Actual Revenue'. It shows 'No Opportunity records found.'
- RECENT CASES:** A table with columns 'Status' and 'Case Title'. It shows 'No Case records found.'

The bottom status bar indicates the account is 'Active'.

Contact was then configured as a sub-entity of Account, as the contacts all belong under an account. At the contact level, we capture position within the company they work for, their individual contact details, and any correspondence they have sent in.

The screenshot shows the Microsoft Dynamics CRM interface for a Contact entity. The breadcrumb trail is 'SALES > Contacts > Test Contact'. The left sidebar shows the 'CONTACT' entity selected. The main content area is divided into several sections:

- Summary:** Displays 'Test Contact' with a profile picture icon.
- CONTACT INFORMATION:** Fields include Full Name (Test Contact), Job Title (Manager), Company Name (Test Account), Email (test@test.ie), Business Phone (01123456), Mobile Phone (0855555555), Fax, Preferred Method of Address (Any), and Address (2 Test Street, Test T5L23). Below is a world map from Bing.
- POSTS, ACTIVITIES, NOTES:** A section for creating and viewing posts, activities, and notes. A post is visible: 'Test Contact Contact Created By Stephen Tyrrell. On Test Contact's wall Today'.
- Company:** 'Test Account'.
- RECENT CASES:** A table with columns 'Case Title', 'Case Number', 'Priority', and 'Origin'. It shows 'No Case records found.'
- RECENT OPPORTUNITIES:** A table with columns 'Topic', 'Status', 'Actual Close Date', and 'Actual Revenue'. It shows 'No Opportunity records found.'
- ENTITLEMENTS:** A table with columns 'Entitlement Name', 'Remaining Terms', and 'Status'. It shows 'No Entitlement records found.'

The bottom status bar indicates the contact is 'Active'.

Once the Account entity was in place and configured, I then moved onto Lead, which would form the base of the Opportunity. Leads consist of Accounts or Contacts that sales representatives feel could be potential opportunities, and this is generally where

the base information of the opportunity is captured. For the Lead forms, I configured them with just the basic information such as the contact details for the Contact and/or Account, and what they may be potentially interested in i.e. the technology or services that they might want to purchase.

The screenshot shows the Microsoft Dynamics CRM interface for a 'Lead' entity named 'Test Lead'. The top navigation bar includes 'SALES', 'Leads', and 'Test Lead'. The main header shows the lead's status as 'Warm' and 'New', with a 'Test Stage' button. The 'Summary' section on the left lists contact details: Name (Test Lead), Job Title (Manager), Business Phone (01234 567890), Mobile Phone (09876543210), and Email (test@dev.co.uk). The 'COMPANY' section lists Test Company, Website (http://test@dev.co.uk), and Address. A world map is displayed below the company information. The right pane shows the 'STAKEHOLDERS' section with a table listing Test Contact as a Stakeholder. The 'POSTS' section shows a single post by Test Lead on 'TestLead' wall.

From there the final step was to configure the Opportunity entity. The Opportunity form is where the information is stored about the opportunity that the sales rep has engaged with the account for. From here it is developed and progressed through various stages, with details such as the current situation, the timeframe that the customer will make the purchase, the budget the customer has set out, the proposed solution to the customer, who the stakeholders are and their relevant roles and contact details.

The screenshot shows the Microsoft Dynamics CRM interface for an 'Opportunity' entity named 'Presentation Test'. The top navigation bar includes 'NEW', 'CLOSE AS WON', 'CLOSE AS LOST', 'RECALCULATE OPPORTUNITY', 'ASSIGN', 'EMAIL A LINK', and 'DELETE'. The main header shows the opportunity's status as 'In Progress' with an estimated revenue of £10,000.00. The 'Summary' section on the left lists opportunity details: Topic (Presentation Test), Contact (Test Contact), Account (Test Account), Purchase Timeframe (This Quarter), Currency (Euro), Budget Amount (£10,000.00), Purchase Process (Individual), and Description (Description text for presentation). The right pane shows the 'STAKEHOLDERS' section with a table listing Test Contact as a Stakeholder. The 'SALES TEAM' section shows Stephen Tynell as a Sales Professional. The 'Current Situation' section shows 'Test' and the 'Proposed Solution' section shows 'Very good proposed solution'.

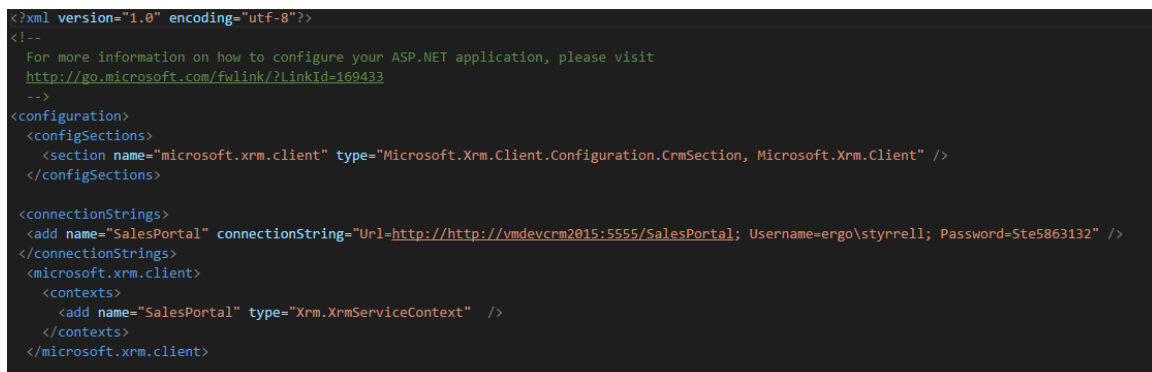
The Portal will be implemented using an ASP.Net Application hosted in Azure.

All GUIs and Views will be implemented using Web Forms, to keep the feel as simplistic and user friendly as possible.

To create the connection to Microsoft Dynamics CRM a combination of declaring the connection strings and using the Microsoft Dynamics SDK to configure the references, alongside using the CrmSvcUtil command prompt to generate Early Bound entity classes, i.e. to create and specify the connection to the entities needed in CRM.

The connection strings are then encrypted using the ASP.NET IIS Registration Tool, which encrypts that section of the Web.config file, by storing the connection strings in an RsaProtectedConfigurationProvider key, which means that only the authorised users can view the connection strings.

This is done as the connection strings contain authentication details including credentials such as a username and password of an Administrator user for the CRM Environment.

A screenshot of a WebConfig file in a code editor. The XML is in UTF-8 encoding. It contains a comment about configuring the ASP.NET application. The configuration section includes a CRM section. The connection strings section contains a single entry for 'SalesPortal' with a URL, username, and password. The Microsoft.Xrm.client section includes a context for 'SalesPortal'.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <configSections>
    <section name="microsoft.xrm.client" type="Microsoft.Xrm.Client.Configuration.CrmSection, Microsoft.Xrm.Client" />
  </configSections>

  <connectionStrings>
    <add name="SalesPortal" connectionString="Url=http://http://vmdevcrm2015:5555/SalesPortal; Username=ergo\styrrell; Password=Ste5863132" />
  </connectionStrings>
  <microsoft.xrm.client>
    <contexts>
      <add name="SalesPortal" type="Xrm.XrmServiceContext" />
    </contexts>
  </microsoft.xrm.client>
</configuration>
```

Screenshot of the WebConfig file to show the connection strings that are initially put in place.


```

<?xml version="1.0" encoding="utf-8"?>
<!--
For more information on how to configure your ASP.NET application, please visit
http://go.microsoft.com/fwlink/?linkid=109411
-->
<configuration>
  <configSections>
    <section name="microsoft.xrm.client" type="Microsoft.Xrm.Client.Configuration.CrmSection, Microsoft.Xrm.Client" />
  </configSections>

  <connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">
    <encryptedData type="http://www.w3.org/2001/04/xmlenc#element"
      xmlns="http://www.w3.org/2001/04/xmlenc#"
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#"
          <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#"
            <KeyName Rsa Key:/KeyName>
            </KeyInfo>
            <CipherData>
              <CipherValue>1AwVHJ1f7Mk6g8gu2HvPI+NaGL+2Ys6N85Zp6zColDnp/APPMKXatFUR8Ifm/ISveXBHvUsFukbqD2PQSiFhsIHLs3pSASm6tJH5CSVo+ppbQ1u0mJ3aGcORbtet1Qby9H8/B9mkH7pPo10INRGBd2B8d04MFY745339UHNUnXktYtAe1AIE3ZroBnIRVVEhy1Y90
              </CipherValue>
            </CipherData>
            <EncryptedKey>
              <KeyInfo>
                <CipherData>
                  <CipherValue>Ntot7sP87edcsBpkdb3nB3s22E1t1Iapt5K8B91lyzXOVdOF+16PyO+ERID8cyERZigh8T5K11tue55Gqx8V7d+SPNACLfdclwBtBurh1BYnh3JgR2sFw6cRztk18P64814g681+YyziL3h/av32B0P/Ba-SVFhm/wJ]CakQo4eDoJ2k1hzQ3AK+H7b6TFDv3wssyph4m29PFL1
                  </CipherValue>
                </CipherData>
              </KeyInfo>
            </EncryptedKey>
          </KeyInfo>
        </CipherData>
      </EncryptedData>
    </connectionStrings>
    <microsoft.xrm.client>
      <contexts>
        <add name="SalesPortal" type="Xrm.XrmServiceContext" />
      </contexts>
    </microsoft.xrm.client>
  </connectionStrings>
</configuration>

```

Screenshot of the WebConfig file to show the connection strings encrypted using the RSA key.

The authentication method will be provided using the existing Active Directory login that the user has been granted by their organisation.

The main GUIs will be implemented using Web Forms, comprising of a combination of text fields as well as lookup fields/drop downs that will replicate the forms within the Dynamics CRM to capture the required fields that the system necessitates to create a record.

For the mobile port, the main method used will be Xamarin Portable Class Library, where all of our C# classes will be analysed to determine the level of mobilisation they are capable, and then re-written accordingly. The main port will be to Android, however depending on the time constraints, iOS and Windows may also be looked at.

In regards to authentication, authentication will be done using form-based authentication against a custom password field within the dynamics CRM implementation to allow for users to view the records. This is done by creating a custom field for password and username on the form, and implementing an authentication mode within the Web.config file.

```

<authentication mode="Forms">
  <forms name=".ASPXAUTH" loginUrl="Login.aspx"
protection="All" timeout="30" path="/"
requireSSL="false" slidingExpiration="true"
cookieless="UseDeviceProfile" domain=""
enableCrossAppRedirects="false">
  </forms>
</authentication>
<authorization>
  <deny users="?" />
  <allow users="*" />
</authorization>
<location path="CreateOpp.aspx">
  <system.web>
    <authorization>
      <allow users="*" />
    </authorization>
  </system.web>
</location> -

```

2.3.1.1 Technologies Used

The main technology used in the project is Microsoft Dynamics CRM. Microsoft Dynamics CRM is one of the world's leading Customer Relationship Management software that helps sales, marketing and customer service professions track customers that they have been in contact with, including storing them in a customer database with all the relevant information such as when they were in contact, how they were in contact, and the topic they spoke about.

It is an IIS-based web application that communicates with a SQL backend server using extensive web service interfaces. It also uses JavaScript libraries primarily for functions as event handlers on entity form events.

All development for the CRM platform is done using Visual Studio 2015 Community Edition, and also employing the use of the Microsoft Dynamics CRM Software Development Kit tool. This tool provides developers with many different applications they can use to further the development of their CRM environment, such as the CrmSvcUtil tool, which generates CRM metadata to be used with external applications and services, for web configuration files to allow connections through secure keys and connections strings. Also the use of the metadata generator tool can be used to generate entity model diagrams which are used to display the intercommunication

between all of the entities in CRM as a whole, and also between specified entities, to give a detailed overview of how they interact with each other.

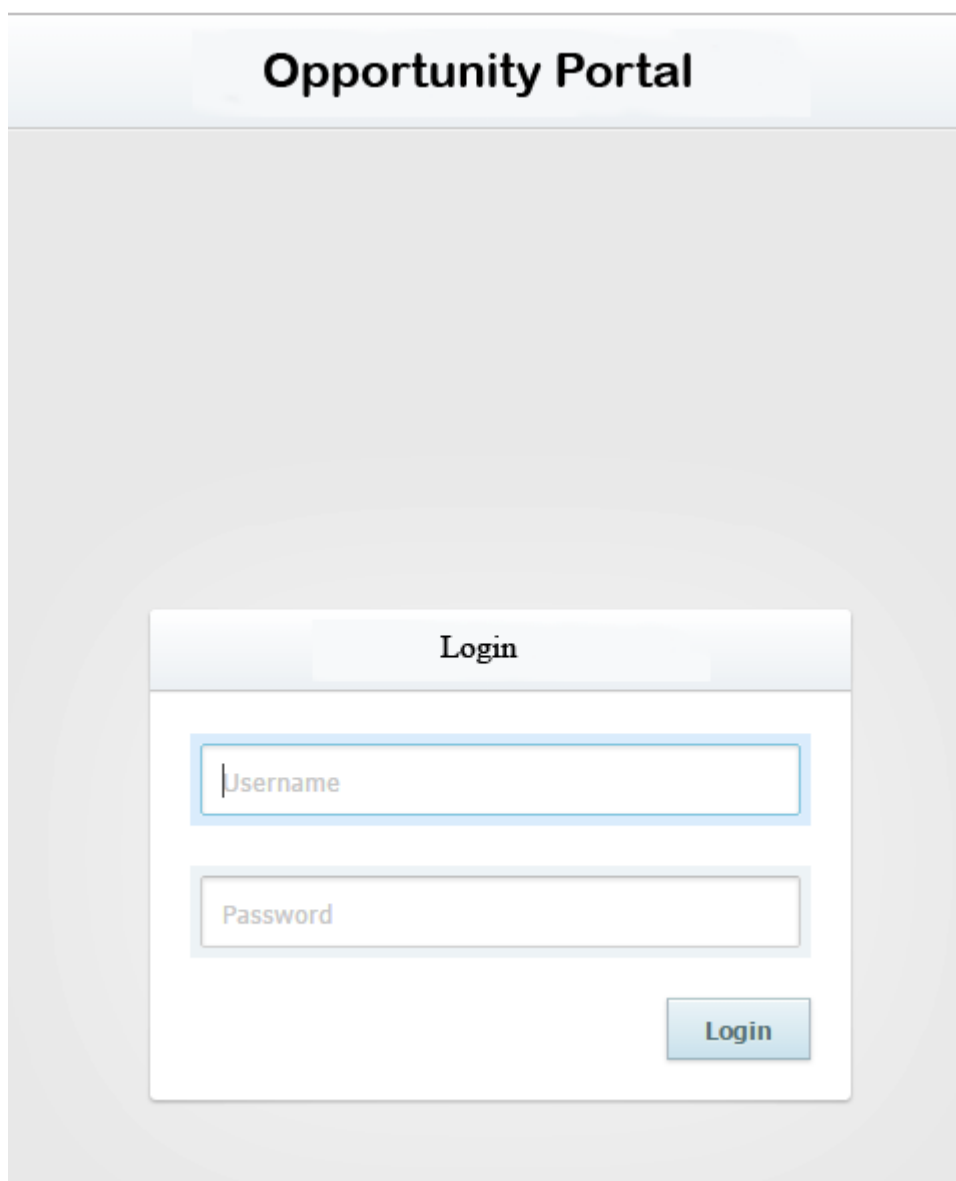
The SDK also provides a set of developer extensions that simplify the development of applications that interact with the Dynamics CRM environment. It extends the core Microsoft Dynamics CRM platform using the `OrganizationServiceContext` class. It provides a simplified connection to CRM servers provided by the `CrmConnection` class, code generation of strong types provided by customisations to the code generation tool, `app.config` and `web.config` configuration options to enable custom extensions, as well as performance enhancements by caching service results that are provided by the `CachedOrganizationService` class.

Also part of the SDK tool is the Portal development tools. Included in these, to facilitate for portals built to extend and communicate with a Dynamics CRM environment are Security Management through the `Microsoft.Xrm.Portal` reference, Cache Management also through the `Microsoft.Xrm.Portal` reference, and also Content Management through the `Microsoft.Xrm.Portal` reference, `Microsoft.Xrm.Portal`. Files reference, and the `WebSiteCopy.exe` tool.

The portal I have built is an ASP.Net application configured with Web Forms that communicates with the Microsoft Dynamics CRM platform using the Microsoft Developer extension tools and the CRM web service interface. Using JavaScript on top of the .Net platform for form events also provides a sleek and interactive user experience. Hosted in Azure, the cloud application means it is always available and secure. The reason Azure is chosen is due to Microsoft's data centres are an industry standard and are known for minimal down time or exposure to hacks and leaks. Also the use of Platform as a Service means that further development and integration with emerging technologies is made simple and allows for cross platform development due to the outstanding Azure service bus that Microsoft has developed for use with any Azure Web Service applications.

FetchXML is a proprietary query language that was developed for Microsoft Dynamics CRM alone. It was based on a schema that describes its capabilities. FetchXML supports similar query abilities as query expressions. Also, it's used as a serialised form of query, used to save a query as a saved view for the users under the userquery entity in CRM, and also at an organisation level for saved views that all users can access. A FetchXML query can be executed using the `IorganizationService.RetrieveMultiple` method from CRM, and you can also convert a FetchXML query to a query expression using the `FetchXmlToQueryExpressionRequest` message.

2.3.2 Graphical User Interface (GUI)



The image shows a web application interface for an "Opportunity Portal". At the top, there is a header bar with the text "Opportunity Portal". Below this, the main content area is a light gray rectangle. Centered within this area is a white login form. The form has a title "Login" at the top. It contains two input fields: the first is labeled "Username" and the second is labeled "Password". Both fields have a light blue border. Below the password field is a blue button with the text "Login".

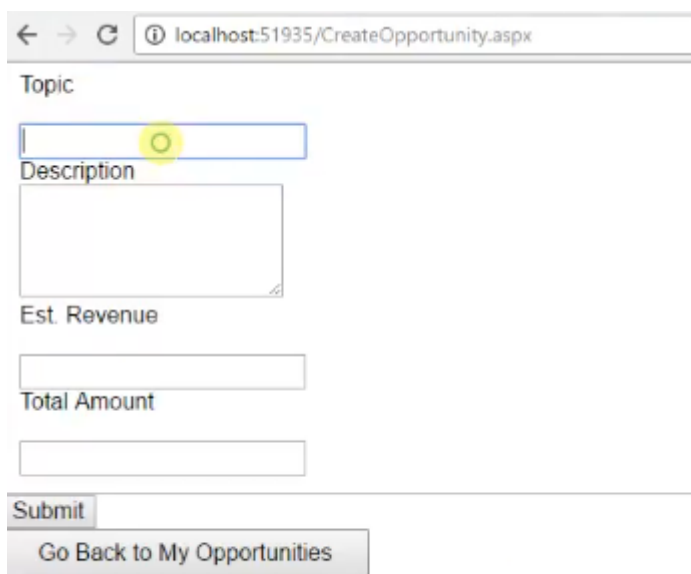
A simple login page that the customer will login using their Active Directory credentials



name	estimatedclosedate	estimatedvalue	parentcontactid	parentaccountid	closeprobability	opportunityratingcode
Presentation Test		€10,000.00				Warm
Test Logging in Portal		€10,000.00				Warm
Step		€0.00				Warm
sss		€500,000.00				Warm
Test		€0.00				Warm
Test		€40,000.00				Warm

[Create New Opportunity](#)
[View My Won Opportunities](#)

Here is the main view that is visible to all users, where they can view their existing open opportunities that they have created, and also can navigate to create new opportunities and view their won opportunities.



localhost:51935/CreateOpportunity.aspx

Topic

Description

Est. Revenue

Total Amount

[Submit](#)
[Go Back to My Opportunities](#)

The main create form, where users will create records. The lookup fields will consist of constrains, so that the user can only select records they own/have permission to view, for example they can only log opportunities against customers they manage, and then the field for contact will be filtered for contacts for that customer.

2.4 Testing and Evaluation

2.4.1 System Function Testing

For testing of the Microsoft Dynamics CRM environment, I chose to employ the use of the Microsoft diagnostics tools. These tools measure the Latency and Bandwidth performance of the Microsoft Dynamics CRM environment, as well as the various JavaScript benchmarks.

The results of these tests will indicate how the environment performs under light load, whether the system is slow to load and return information, and how well optimised the JavaScript libraries are.

=== Latency Test Info ===

Number of times run: 20

Run 1 time: 27 ms

Run 2 time: 36 ms

Run 3 time: 25 ms

Run 4 time: 33 ms

Run 5 time: 25 ms

Run 6 time: 25 ms

Run 7 time: 29 ms

Run 8 time: 32 ms

Run 9 time: 26 ms

Run 10 time: 27 ms

Run 11 time: 24 ms

Run 12 time: 25 ms

Run 13 time: 27 ms

Run 14 time: 35 ms

Run 15 time: 31 ms

Run 16 time: 28 ms

Run 17 time: 30 ms

Run 18 time: 31 ms

Run 19 time: 26 ms

Run 20 time: 25 ms

Average latency: 28 ms

Client Time: Sun, 07 May 2017 12:33:46 GMT

=== Bandwidth Test Info ===

Run 1

Time: 55 ms

Blob Size: 15180 bytes

Speed: 269 KB/sec

Run 2

Time: 68 ms

Blob Size: 15180 bytes

Speed: 218 KB/sec

Run 3

Time: 45 ms

Blob Size: 15180 bytes

Speed: 329 KB/sec

Run 4

Time: 45 ms

Blob Size: 15180 bytes

Speed: 329 KB/sec

Run 5

Time: 41 ms

Blob Size: 15180 bytes

Speed: 361 KB/sec

Run 6

Time: 45 ms

Blob Size: 15180 bytes

Speed: 329 KB/sec

Run 7

Time: 41 ms

Blob Size: 15180 bytes

Speed: 361 KB/sec

Run 8

Time: 46 ms

Blob Size: 15180 bytes

Speed: 322 KB/sec

Run 9

Time: 41 ms

Blob Size: 15180 bytes

Speed: 361 KB/sec

Run 10

Time: 47 ms

Blob Size: 15180 bytes

Speed: 315 KB/sec

Max Download speed: 361 KB/sec

Client Time: Sun, 07 May 2017 12:33:47 GMT

=== Browser Info ===

Browser CodeName: Mozilla

Browser Name: Netscape

Browser Version: 5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36

Cookies Enabled: true

Platform: Win32

User-agent header: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36

Client Time: Sun, 07 May 2017 12:33:47 GMT

=== Machine Info ===

Client IP Address: 172.24.24.47

Client Time: Sun, 07 May 2017 12:33:47 GMT

=== Array Manipultaion Benchmark ===

Time: 562 ms

Client Time: Sun, 07 May 2017 12:33:47 GMT

=== Morph Benchmark ===

Time: 84 ms

Client Time: Sun, 07 May 2017 12:33:48 GMT

=== Base 64 Benchmark ===

Time: 6 ms

Client Time: Sun, 07 May 2017 12:33:48 GMT

=== DOM Benchmark ===

Total Time: 77 ms

Breakdown:

Append: 23ms

Prepend: 20ms

Index: 2ms

Insert: 17ms

Remove: 15ms

Client Time: Sun, 07 May 2017 12:33:48 GMT

=== Organization Info ===

Organization name: SalesPortal

Is Live: False

Server time: 07/05/2017 12:33:33 UTC

Url: <http://vmdevcrm2015:5555/SalesPortal/tools/diagnostics/diag.aspx>

Client Time: Sun, 07 May 2017 12:33:48 GMT

From

For the Microsoft Dynamics CRM environment, I will also utilise Microsoft's CRM Performance Toolkit. This will analyse the performance of the front end and backend, testing the latency in the system, how long it takes operations to run such as forms loading or actions being triggered. We will also conduct load testing, to make sure that there is no detriment in performance with a large number of users logged into the system or trying to access it and perform operations at the same time. This is impertinent as the system is likely to be used in larger organisations consisting of large numbers of employees.

To test the security of the servers and infrastructure, the Windows Server 2012 Security Configuration and Analysis Tool will be deployed on each server. Windows Server 2012 is secure out of the box, however there can be certain features that can leave unwarranted gaps in firewalls and links between other servers. We will analyse the results and make the corresponding adjustments to either the server itself or the firewall that is deployed on the server, depending on what is discovered from the testing.

For Stress Testing/Load testing of the portal, we will use two separate applications, the first being the Open Source application Netling. Netling is a load tester client for easy web testing. It is extremely fast while using little CPU or memory. The reason load testing is important is because the application has to be able to handle multiple resources trying to access it at the same time. Below is the testing results.

```
netling http://localhost/CRMPortal -t 8 -p 30 -a
```

Running 10s test @ http://localhost/CRMPortal

Threads: 8

Pipelining: 30

Thread affinity: ON

26880 requests in 10.08s

Requests/sec: 2666

Bandwidth: 9 mbit

Errors: 0

Latency

Median: 106.211 ms

StdDev: 67.039 ms

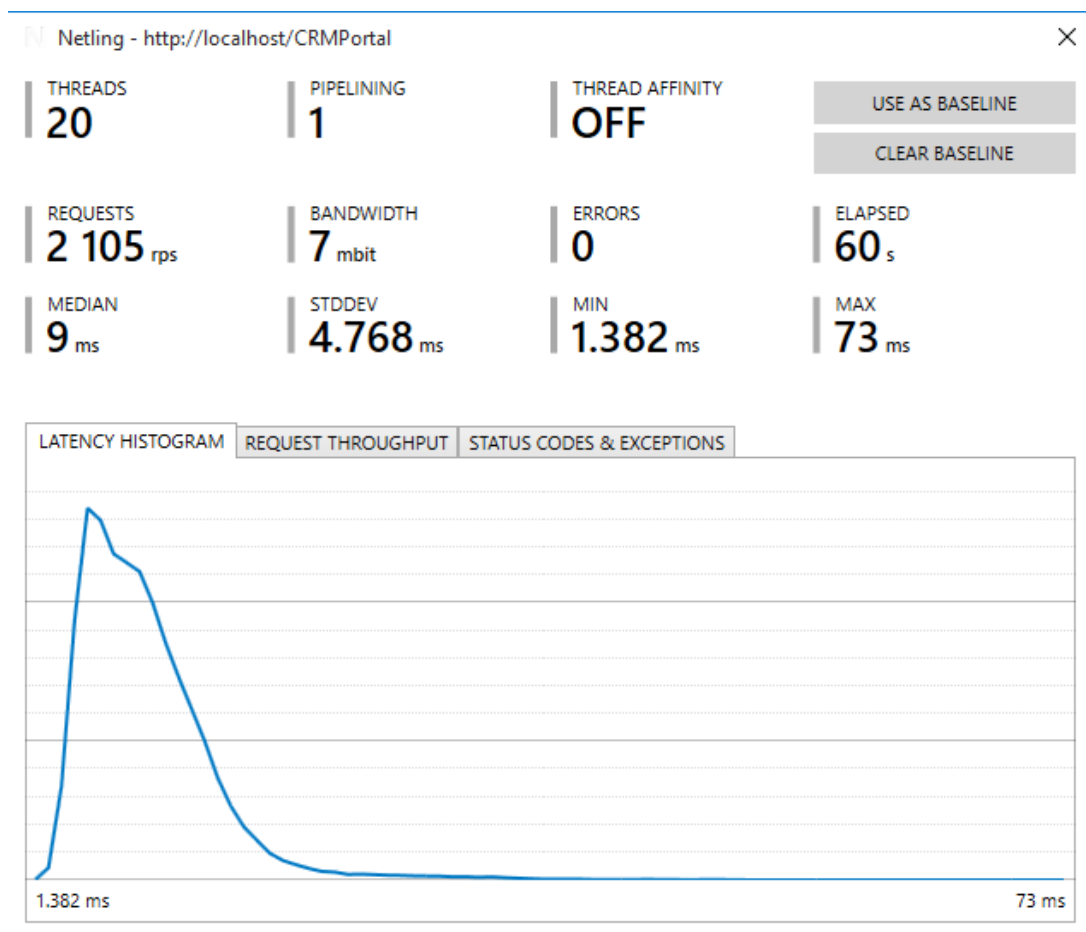
Min: 6.246 ms

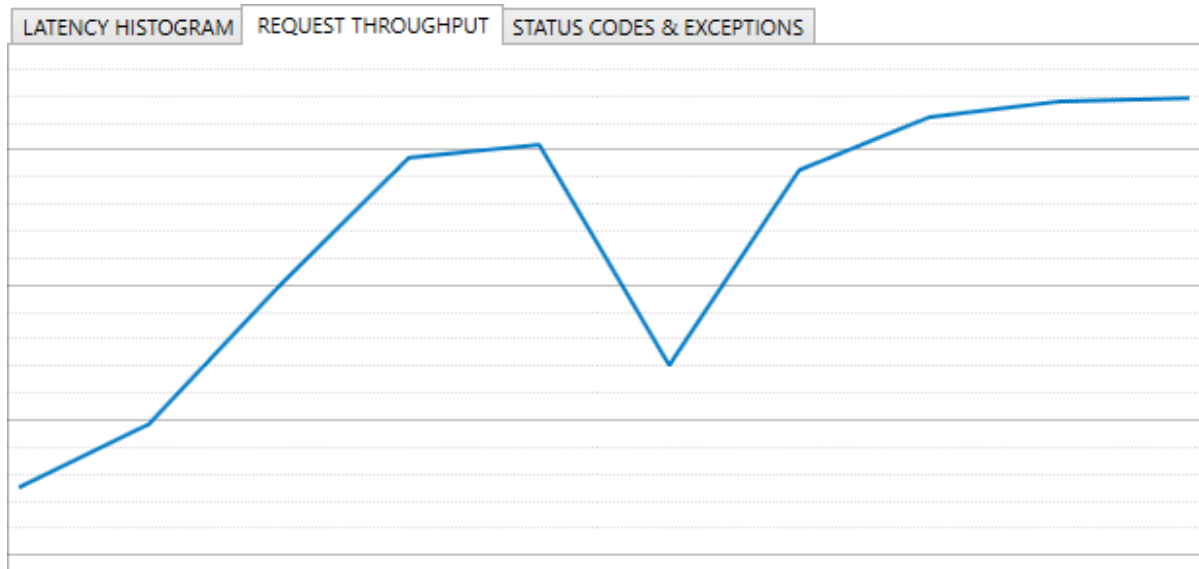
Max: 477.453 ms



6.246ms=====

477.453 ms





As we can see from the above results, the system holds well under short bursts of activity from multiple users, with no errors occurring throughout the process. However, where we may run into issues is when there are multiple users concurrently running long tasks or procedures. As seen from the final graph above, the application comes under significant load when 20 users are concurrently running tasks

Functional testing for the portal will be done using the pre-defined test plans, end-user scenarios drawn out to imitate the steps/processes a user will go through when using the system.

Test Case 1: End user logging into the system.

Testing will make sure the user can only login using valid credentials, and special characters are correctly accepted.

Test Case 2: User security profile

This will cover the requirement of the user only being able to view their own records, depending on their privileges.

Test Case 3: User creating an opportunity

Testing will take into account all configurations/options a user can select when creating an opportunity, and making sure the process flow of creating an opportunity is functioning correctly.

Test Case 4: User updating an opportunity

Testing will take into account the user's options for editing an existing opportunity.

For the regression testing (IE the web application GUI), I will use Selenium WebDriver, a browser-based tool that is used to automate web applications for testing purpose. Selenium will rigorously test each element of the front-end application and produce bug reproduction scripts and also create scripts to aid in automation-aided exploratory testing.

Below is an example of one of the Selenium test scripts that was ran against the environment to test the functionality of the application, and also to test regression.

Test Case

Source

Untitled *

Untitled 2 *

Command	Target	Value
open	/CRMPortal/Login.aspx?ReturnUrl=%2fcrmportal	
type	id=Login1_UserName	styrrell
type	id=Login1_Password	password
click	id=Login1_RememberMe	
clickAndWait	id=Login1_LoginButton	
clickAndWait	id=CreateButton	
type	id=NewOpportunityView_name	Test
type	id=NewOpportunityView_description	TEst
type	id=NewOpportunityView_estimatedvalue	120000
type	id=NewOpportunityView_totalamount	20000
clickAndWait	name=NewOpportunityView\$ctl29	
clickAndWait	id=SubmitButton	
clickAndWait	id=GoTo	
type	id=Login1_UserName	styrrell
type	id=Login1_Password	password
clickAndWait	id=Login1_LoginButton	

Command

type

Target

id=NewOpportunityView_description

Value

TEst

Runs: 1

The above test script allows runs a list of commands against the application at high speed to test for latency and any errors with the application.

Performance testing will consist of several different variables:

- Response time or latency, measured at the server and measured at the client

- Throughput - the number of requests per second that can be managed by the application
- Resource Utilisation – How heavy the application is on a user's machine (CPU, RAM, Network).
- Workload will also be included in the performance testing, however the results will be recycled from the load testing.

To conduct this performance testing, I used WebSurge. WebSurge is an application that spoofs or imitates user connections simultaneously, to see how well the application responds. For the purpose of the project, and given the earlier load testing that had been conducted, the aim of the WebSurge test was to overload the system and push it to its limits. In order to do this, I set the test for 22 threads, meaning that 22 sessions would be simultaneously conducted on all of the portal pages, for 1000 seconds non stop. This equated to nearly 200 requests sent to the application per second, or 12000 requests per minute. In real situations, this load would not be expected, however it is a good measure of future proofing the system as if an organisation was to expand it may need to be able to handle this kind of simultaneous workload. Below is a sample of the responses received from the application through WebSurge.

</ResponseContent>

<StatusCode>200</StatusCode>

<StatusDescription>OK</StatusDescription>

<ResponseLength>4497</ResponseLength>

<TimeTakenMs>79</TimeTakenMs>

<TimeToFirstByteMs>79</TimeToFirstByteMs>

<IsWarmupRequest>>false</IsWarmupRequest>

</HttpRequestData>

<HttpRequestData>

<SortOrder>0</SortOrder>

<Id>4727854023723107060</Id>


```
<Timestamp>2017-05-07T14:16:47.2453917Z</Timestamp>

<IsActive>true</IsActive>

<Url>http://localhost/CRMPortal/CreateOpportunity.aspx</Url>

<HttpVerb>GET</HttpVerb>

<RequestContent />

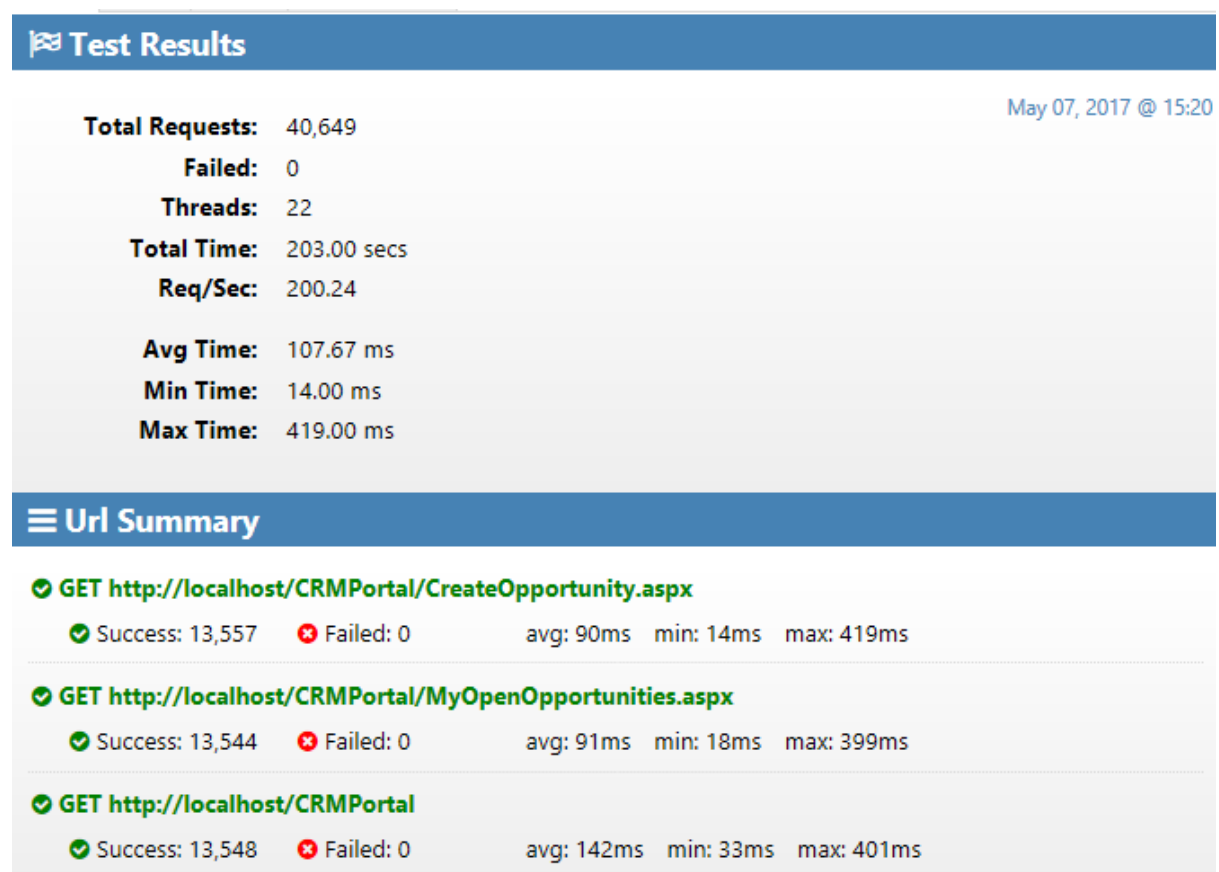
<AuthenticationType>Windows</AuthenticationType>

<Headers>

  <HttpRequestHeader>

    <Name>Accept-Encoding</Name>
```

Below is the overall results from the stress testing:



As you can see, there were a total of 40,000 requests sent to the application over the span of a 3 minute period. This is as before mentioned extreme testing and would not simulate a real life environment. However as you can see from the above graph, the results came back with zero failures which is a superb result as it means the

application withstood the barrage of requests and did not crash. One thing to be noted however, is the difference between the minimum time and the maximum time the request took to complete. From analysing the logs also, it is evident that as the requests went on, the time it took to complete the request increased dramatically. If the tests had continued for a substantial while longer, one can only assume that the application would have failed due to a time out error.

Another aspect to the performance testing is testing of the application across different platforms/browsers. One of the requirements is the testing of the application in different web browsers (Google Chrome, Firefox, Internet Explorer, Edge, etc), as well as across different type of mobile devices to test the scaling for example mobile phones of different screen sizes and operating systems (Android, iOS, Windows) and also the respective tablets (iPads and Android

2.4.2 User Interface Evaluation

User Interface Evaluation will be done by applying Usability Heuristics as the chosen method for UI testing. The reason this was chosen over Empirical Evaluation, was due to the time constraints and the lack of access to suitable users.

We will employ Jakob Nielsen's ten general principles for interaction design, which are the widely used standard for heuristic evaluation:

Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

In terms of the portal, JavaScript has been put in place to notify users of actions taken, such as when the form has been posted for creating a record, to allow them to know that it is okay to proceed back to another page. The latency on these JavaScript libraries is also minimal, so the user will be updated in near realtime.

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

As it is a user-based system, the language used is quite simplistic and speaks well to the intended users as it is mostly sales speak and there is little 'technical speak' as

the system is not targeted towards technically orientated users. This leads for a simplistic and easy user experience as there is no confusion and the users can relate to the system.

User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Undo is supported by the inbuilt browser functions, the application itself has been designed to incorporate this functionality. In regards to 'emergency exits', each page has consistent navigation, including a route for the user navigate back to the home page from all forms.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Conventional wording is employed across all forms, as the labels of the fields are clearly named across the entire system.

Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action

Error prevention is again done through JavaScript, when errors occur, notification boxes are employed through JavaScript to allow the user know the error that has occurred, with a detailed response of the error. This allows the user to know what the error was and how they can prevent repeating the error.

(Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Information is carried across from entity to entity, and preloaded by selecting option sets. For example, when an Account or Contact is selected when creating a lead or opportunity, their contact details are automatically populated on the form. This means users do not have to make a double entry of information.

Flexibility and efficiency of use

Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Efficiency of use is key to the application, as the whole premise of the application is to make the user's workflow more efficient by giving them the access to this lightweight application.

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

As the design is lightweight, users are only prompted with the information that they need in order to complete the functions that are available through the application.

Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Error messages are custom done through JavaScript, and are designed with user-level language to provide a clear explanation of exactly what has happened to cause this error, and how it can be prevented.

Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

A lightweight user manual will be delivered to users of the system, to explain the nuances of the system. However as users will be familiar with the dynamics CRM

system there will be little training involved, merely just getting used to the UI and how it interacts with the system. In terms of technical documentation, a full technical document will be compiled and provided to system administrators to allow for support and maintenance of all issues as the idea behind the system is that once it is in place there is only minor configuration needed for further progression within the organisation.

2.4.3 Customer testing

Customer testing will be done by a selection of end users, where test scripts will be in place to run through scenarios that an end user would complete, to see how they deal with navigating through and using the system. This form of User Acceptance Testing will provide feedback from valid end users and any changes/suggestions will be considered before the final revision/implementation to allow for the application to be geared towards the end user's usability.

User testing will also take place on the user's device in order to test the systems functionality and performance across the various standard devices in use in said businesses

2.4 Conclusion

After gathering and collating the results from testing of both the Microsoft Dynamics CRM environment, and the portal application, we can conclude that security is an issue of concern for the portal. One way this might be improved is by implementing a Single Sign On method of authentication, or possibly even secure the application with endpoint security certifications.

In regards to performance, the application holds up quite well under stress, and is able to handle multiple requests at the same time without failing. There is a trade off with the requests returning slower as the load increases, but this is not majorly noticeable from a user perspective and does not cause any connection timeouts at an application level.

For the dynamics CRM environment, one thing that is noticed from the performance testing is that JavaScript libraries are quite well optimised for the system and the benchmarks show that there is little delay between the form load request and the

libraries loading on the form. This is great, as it means users are not left with hanging loading screens waiting for their information to load and the system is not under straining when attempting to load these pages.

One thing to be noted however is that the latency and bandwidth for the CRM environment are not at what would be considered optimal measures, and there is a minor delay between operations. This is fine in its current use and configuration, however this may cause issues in the future, if the system were to expand with the amount of data within it, from records created by users, and also when the load of users using the system at the same time, could cause the system to be quite laboured and slow to complete requests. One way to combat this is not only to optimise the SQL database using indexing. Indexing is used to improve searches on specific search columns that are used in CRM views or advanced find, to reduce the amount of time spent return the records to the users client.

Another option to improve performance would also be to increase the resources available on the servers themselves. Increased RAM and CPU cores can benefit CRM systems astronomically, as the more resources available to the system, the faster process can be completed and queries returned.

Performance issues could also be a result of a poor network infrastructure. If an enterprise company has only deployed a gigabit network connection between their servers and also between client machines, noticeable lag/delay can be noticed. The same can be said for users who are connected to the internet using WiFi rather than using an ethernet cable. For an enterprise system, a 10 gigabit network connection is the recommended standard for cross-server communication, particularly when the data centre is not isolated from the rest of the office and connected to the rest of the company's network.

3 Conclusion and Future Work

One of the things we can conclude from the deployment of the Dynamics CRM environment, is that one of the main focuses that needs to be emphasised is the standard of the infrastructure put in place. The better the servers specifications that are deployed, such as RAM and CPU core availability, as well as the networking that will connect the entire system together, can play a huge part in how the system will perform not only under load, but in general. Performance is one of the key factors that will turn off prospective customers/users from utilising a system to it's full ability. Nobody wants to use a slow and cumbersome system.

Microsoft Dynamics CRM is extremely customisable to suit every business and organisation, and the possibilities of what can be done is almost endless. One of the main points that I would make is that the system that was deployed as part of the project was not a full customisation as I did not feel there was a certain brand or company for the system to fit or mold to fully. As you can see however, the vanilla crm implementation is still quite user friendly and attractive from a UI perspective, and with further configuration can be made look like a top end system.

One of the further points to make from the conclusion of the project is that little integration pieces feature, other than the Bing Maps API, so it does not demonstrate to the full extent of the extendability and compatibility of Dynamics CRM, from an integration point of view with 3rd party systems and also other systems within the Microsoft ecosystem.

The main advantages of the portal lie in the marketability/usability for existing CRM customers, from the feedback I've received from companies and end users, it's a simple system that can improve the workload and hassle for a lot of salespeople. Also the fact that once the project has been implemented once that it can be expanded upon for different entities as the connection strings and authentication is already in place is a huge factor.

The mobile aspect also has proved to be a major advantage, as the only existing mobile clients (Resco and the Dynamics 365 application for mobile) are extremely resource heavy and demand high end devices, and are also expensive to licence, and are not user friendly. Hence the idea to go with a lightweight user friendly client.

The limits of the portal are that it's not as secure as the standalone Dynamics CRM platform, as it does not incorporate Single Sign on or Multifactor authentication, which means that companies using sensitive data or who are for example, SOC compliant, will not be able to utilise the system.

At this point in the project, the disadvantages and advantages are hard to gauge before the application itself has been fully implemented and configured, however this is a point that will be returned to post completion of the project once end user feedback has been received and once the full performance of the project can be evaluated over a period of time by the developer and possibly a system administrator.

Further development or research

With the introduction of Microsoft's latest release of Dynamics CRM, known as Microsoft Dynamics 365, the scope for further development is increased exponentially. Microsoft Dynamics 365 is a cloud solution of Microsoft CRM, which is built purposely around integrations with Microsoft's other Dynamics line of productions such as Microsoft Dynamics AX which is their ERP system for mid to large enterprises, which will allow for integration between CRM and process management such as procurement, human resources, inventory management and more detailed customer management. Also there is Microsoft Dynamics NAV which is Microsoft's alternative ERP system that centres around finance and analytics for small and medium enterprises. The integration with NAV allows for scope of invoicing of customers and projects through one core system, which also provides for excellent reporting as having all of the information correlated and accessible through one point of access means products like Power BI (which can also be integrated with CRM) can provide in-depth, detailed and also expandable reporting at the click of a button, allowing businesses to see their information in real time.

One of the other key selling points for the Dynamics 365 platform is the seamless integration with Office 365. Outlook, excel and word are the core staple applications of almost every business. With Office 365 and Dynamics 365 integrated, users can now create/read/update files within their browser, using tools such as Excel Online and Word Online. This is a huge turnaround from typically having to download the files, edit them in the desktop version of these applications and upload again. This will not

only save time and energy for businesses, but also storage space on local machines and also on the servers as it eliminates different revisions having to be uploaded.

As Dynamics 365 is a cloud based application, and Microsoft's API endpoint is now accessible through the cloud, integration with other Microsoft systems is also now extremely accessible. Integrations with services such as HubSpot as a marketing tool, SharePoint as a document library, Skype For Business as a VOIP system for messaging and calls, Yammer for a business communication platform, LinkedIn for contact management, and as before mentioned PowerBI are all extremely easy to configure to give the organisation a true integrated and easy workflow.

With the ongoing expansion of Microsoft Azure also, the Azure Service Bus can now be utilised to integrate other 3rd party systems that are outside of the Microsoft domain with ease, such as ServiceNow as a service desk tool, third party booking systems such as EventBrite, other ERP systems such as SAGE50 which is used for sales/purchases/payments and other accounting purposes,

With more resources, the application can be expanded/replicated for other areas that the Microsoft Dynamics CRM system encompasses, for example a service portal where customers can log support tickets, view updates/resolutions on the cases that they have logged, and can also close cases if they have resolved them on their side. The same can be said for a Marketing portal, where for examples leads could be created from contacts the marketing team has engaged with, as well as marketing campaigns, or even a portal for resources to log their timesheets for the week/month.

Another element that could be considered is storing records offline which would then be synced when the user goes online. This would be beneficial in cases with remote employees or employees who are travelling, where a stable internet connection is unavailable. Users could create records offline without an internet connection, and then once they connect their device to the internet their records are synchronised to their Dynamics CRM environment.

In regards to research, I conducted my research into the project by consulting senior engineers/developers within my own organisation to scope the plausibility/implementation plan to see how feasible the project was.

To gauge the usability/demand for the project, I created a survey using SurveyMonkey and sent it to 3 different organisations/companies that currently employ a Microsoft Dynamics CRM system to manage their business cycle, to scope/gather their interest and opinions on using such a application.

Across two organisations, the majority felt like such an application would help improve their workload and time, (78% and 86% of people within the organisations responded that the application would be something they would be interested to incorporate into their jobs.) The final organisation was not interested as they were concerned around the security of the application due to their nature of business being one that holds a lot of sensitive customer information.

In regards to the mobile ports, as previously mentioned Android will be the main implementation, however given more time the port for Windows phones and iOS devices would also be feasible, however given the timeline, this may not be possible to complete within this revision of the project.

Another potential development aspect that could be looked at in the future would also be the implementation of Multi-Factor authentication, or single sign on. These would both be using sign in credentials from an Active Directory account using Microsoft ADFS, or possibly even Microsoft Office 365. At present, the knowledge to investigate/research this option is not available to myself, however it is something to consider in future implementations.

References

Msdn.microsoft.com. (2016). Chapter 16 - Testing .NET Application Performance. [online] Available at: https://msdn.microsoft.com/en-us/library/ff647788.aspx#scalenetchapt16_topic9 [Accessed 11 Dec. 2016].

Msdn.microsoft.com. (2016). Portal developer guide for Microsoft Dynamics CRM. [online] Available at: [https://msdn.microsoft.com/en-us/library/gg695814\(v=crm.7\).aspx](https://msdn.microsoft.com/en-us/library/gg695814(v=crm.7).aspx) [Accessed 11 Dec. 2016].

Msdn.microsoft.com. (2016). Prepare for portal development. [online] Available at: <https://msdn.microsoft.com/en-us/library/gg695813.aspx> [Accessed 11 Dec. 2016].

CANITPRO. (2017). Step-By-Step: Setting up AD FS and Enabling Single Sign-On to Office 365. [online] Available at: <https://blogs.technet.microsoft.com/canitpro/2015/09/11/step-by-step-setting-up-ad-fs-and-enabling-single-sign-on-to-office-365/> [Accessed 7 May 2017].

Dynamics CRM in the Field. (2017). CRM Performance Toolkit. [online] Available at: <https://blogs.msdn.microsoft.com/crminthefield/2010/09/09/crm-performance-toolkit/> [Accessed 7 May 2017].

Hanselman.com. (2017). Two tools for quick and easy web application load testing during development - Scott Hanselman. [online] Available at: <http://www.hanselman.com/blog/TwoToolsForQuickAndEasyWebApplicationLoadTestingDuringDevelopment.aspx> [Accessed 7 May 2017].

Msdn.microsoft.com. (2017). Software Development Kit for Microsoft Dynamics 365 (online) and Dynamics 365 (on-premises). [online] Available at: <https://msdn.microsoft.com/en-us/library/hh547453.aspx> [Accessed 7 May 2017].

Nngroup.com. (2017). 10 Heuristics for User Interface Design: Article by Jakob Nielsen. [online] Available at: <https://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed 7 May 2017].

Support.simplifiedns.com. (2017). Virtual hosting with IIS (Internet Information Services). [online] Available at: <http://support.simplifiedns.com/kb/a82/virtual-hosting-with-iis-internet-information-services.aspx> [Accessed 7 May 2017].

Technet.microsoft.com. (2017). Planning your deployment of Microsoft Dynamics 365. [online] Available at: <https://technet.microsoft.com/en-us/library/hh699722.aspx> [Accessed 7 May 2017].

Technet.microsoft.com. (2017). Security and Protection. [online] Available at: [https://technet.microsoft.com/en-us/library/hh831778\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/hh831778(v=ws.11).aspx) [Accessed 7 May 2017].

Weblog.west-wind.com. (2017). Announcing West Wind Web Surge 1.0. [online] Available at: <https://weblog.west-wind.com/posts/2015/sep/02/announcing-west-wind-web-surge-10> [Accessed 7 May 2017].

Anon, (2017). [online] Available at: <https://www.linkedin.com/pulse/installing-ms-dynamics-crm-2015-windows-server-2012-neeraj-kumar> [Accessed 9 May 2017].

xRM Wordpress. (2017). Microsoft Dynamics CRM 2015 System Requirements | xRM. [online] Available at: <http://xrm.com/reference/microsoft-dynamics-crm-2015/system-requirements/> [Accessed 9 May 2017].

Appendix

Appendix 1 - Project Plan

Project Proposal

Microsoft Dynamics CRM Salesperson Portal

Stephen Tyrrell, x13451692, Stephen.Tyrrell@student.ncirl.ie

BSc (Hons) in Computing Evening

Software Development Specialisation

15/10/16

1. Objectives

The purpose of this document is to present the proposal for the development of a hosted web portal for salespeople to log opportunities to CRM without having to be on an internal network (for non IFD) and also be a simplified version of the Microsoft Dynamics CRM for Mobile form.

The intended customers are existing Microsoft Dynamics CRM customers using CRM as a sales tracking/sales management system, or new customers interested in implementing the system.

2. Background

3. The background to this project comes from my personal experience in working with and implementing Microsoft Dynamics CRM solutions for customers through my job as a CRM Developer.
4. Microsoft Dynamics CRM is one of the market's leading Customer Relationship Management systems, and is used by companies of all sizes, across the globe. It is renowned for its sales processing capabilities, for tracking leads and opportunities as well as a marketing tool which can be integrated with several 3rd party systems such as HubSpot and EventBrite.
5. However, typically the web interface for Dynamics CRM is bulky and slow, due to the amount of functionality and resources used for the different parts of the system (Marketing, Project, Service, Sales etc.), and frequently we see concerns raised by users in regards to slow loading times, browsers crashing on low powered machines due to the heavy client side scripting on the pages due to JavaScript functionality. To combat this, my project also looks to deploy an ASP.Net application that serves as a 'portal' to allow Sales reps and other users to access their CRM records and create new records for just the Sales area of the system, particularly on lower powered systems.
6. My experience with implementing solutions using a combination of ASP.Net applications using Web Forms and deploying within IIS from personal projects and at a business level will benefit me greatly I feel, as I'm aware of the challenges and strategies that have to be undertaken in order to complete this project correctly and on time with my project plan. This will also be the first time I've implemented a solution such as this, which I feel will be a great learning curve to push my development skills as well as my personal experience with the system, as

implementing the full application lifestyle and then extending beyond the system's capability/function as standard.

7. Technical Approach

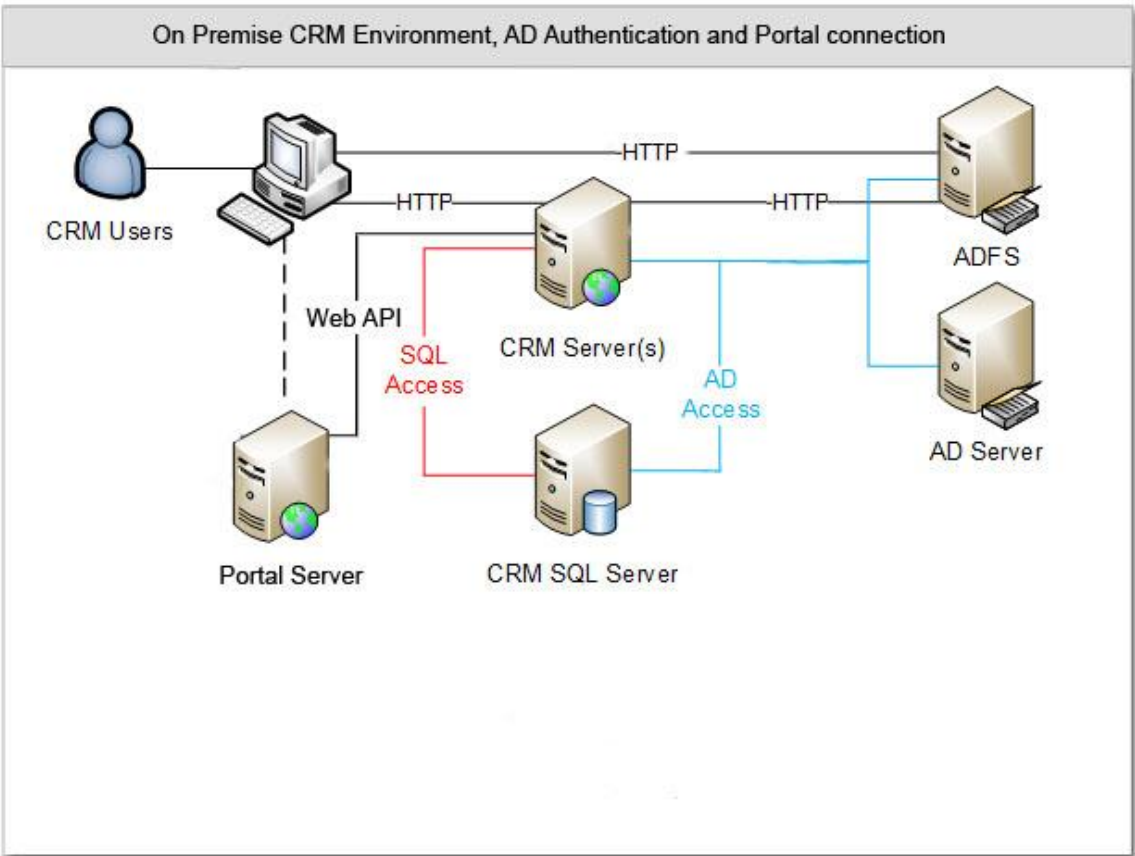
Technologies

The Microsoft Dynamics CRM infrastructure consists of the Application itself which will be deployed on a stand alone Application Server, and will be available through an IIS website. It will communicate with a SQL server which will store the database and allow for read and write functions. There will also be an AD server to store user profiles and credentials.

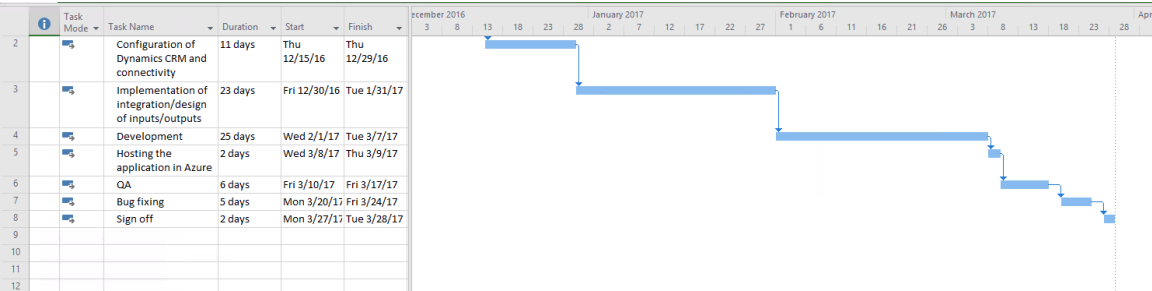
The project also consists of a 'portal' which is an ASP.NET web application using Web Forms, that will be hosted within azure, that connects to a Microsoft Dynamics CRM organisation. As mentioned before, all the front-end components will be Web Forms, and the overall application will be hosted in Azure. The project will be implemented using C#, and the Dynamics CRM organisation that will be used will be standard out of the box functionality, bar some minor adjustments to the contact form to allow for user authentication.

For the mobile port, Xamarin will be used to transform the completed Web Application into a mobile friendly version that will be able to run locally. This will primarily be focused on Android devices due to Android being a common choice amongst businesses, however Windows and iOS ports will also be considering depending on user feedback and time constraints.

Below is the overall view of the application structure and infrastructure.



8. Project Plan



9. Technical Details

ASP.NET application implemented in C#, using Web Forms for the front-end components. Will be hosted in Microsoft Azure.

The authentication will be configured through Dynamics CRM, using the user's profile to provide/update login details as well as security roles.

The Dynamics CRM instance will be hosted on-premise, hosted on a Windows Server running Server 2012R, with a SQL database on the same server.

All mobile development/portability scaling will be done through Xamarin, essentially providing a scaled version of the Web Application that will have the capability to run locally on any mobile device.

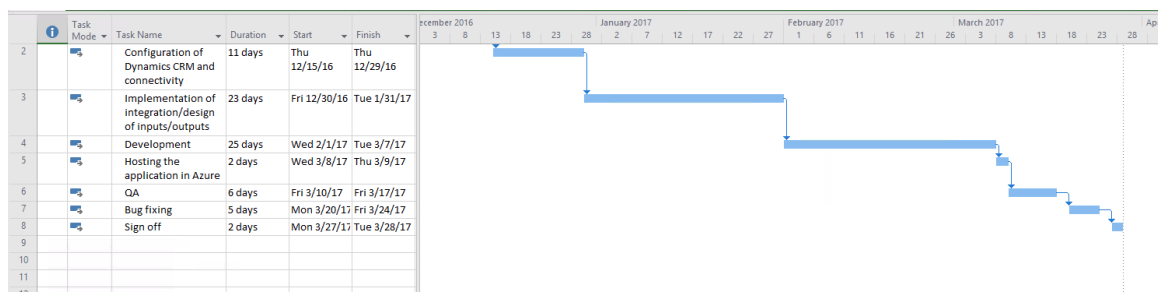
10. Evaluation

The testing routine will consist of running through each of the various end user scenarios, making sure each functionality performs correctly. This includes creating and editing opportunities from the portal, and also updating opportunities from within CRM itself and making sure that the updates are visible within the portal.

The authentication will also be tested, making sure that no other users will be able to view opportunities other than their own, unless given elevated privileges.

Signature of student and date

Project Plan



Appendix 2 - Monthly Journals

September Monthly Reflective Journal

Stephen Tyrrell x13451692

This month was spent researching the viability and uniqueness of my project. I had a good grasp on the technologies needed to build the integration, but the fact it hadn't been done before left me questioning whether it was feasible or not. Using resources such as Microsoft Developer Network, consulting with senior developers within my job who have more experience with server and client side integration, I realised that although tricky and a long process, I believe it will be possible.

I also had to prepare my project pitch, which was quite difficult to explain exactly what I wanted to do without it sounding simple. On the face of it, my project literally sounds like uploading photos to a system, however it's a lot more complex than it seems.

The only way to prove the complexity of the project was to start to map out the requirements and flow of the project, to get an idea of exactly what was required for the development and implementation of the systems. Several brainstorming sessions in front of a large whiteboard took place to achieve this, and towards the end of the month I felt like I had compiled a significant display/argument to show how the project would not only be implemented, but how it can fulfil the requirements set for us as students for the final project. Once I had compiled everything together into a legible document, I sent this to my supervisor and Eamon for review, as their input would be needed.

October Reflective Journal

This month I met with my project supervisor Catherine, to discuss my project idea and to plot out the use cases and flow. As discussed in the meeting, the main point was that there was only two actors, and one of the things I needed to do throughout the requirements specification was clarify and state that despite building upon existing infrastructure and programs, the code I will be creating will not be an improvement or modification of an existing codeset, but rather a bespoke codeset that has not been done before.

This month I also documented my use cases and constructed the process flows as part of my requirement specification document

After meeting with my project supervisor at the end of the month, we came to the conclusion that my initial project idea (a Dynamics CRM integration with Active Directory) would not fulfill the criteria needed for a final year project. Luckily I had a couple of other project ideas I had toyed around with, and after discussing it with Catherine and Eamonn Nolan, I decided to revisit my idea of implement an ASP.Net portal that would allow salespeople to create/view/edit opportunities in CRM. The task ahead of me to re-write my project proposal/requirements specification seemed daunting but I managed to get through it all.

November Reflective Journal

Having met with Catherine after submitting my requirements, I set out on conducting further research/discovery for my project idea. This involved reaching out to contacts within the industry I worked in, customers that had Microsoft Dynamics CRM systems in place and were using the system in a way that they would benefit from the portal. I also consulted other senior developers to gain their perspective on what way the project should be implemented and how they felt the project should be structured. This proved to be extremely helpful as it offered me some guidance in areas I was not fully convinced would be possible to implement, particularly around authentication, as I had not seen any examples of the same sort of system implemented with strong authentication factors.

December Reflective Journal

December would prove to be the hardest month of this semester, with other uploads and projects conflicting with the time needed to do my Technical Report, and also my prototype.

I started the development on my prototype, mainly around implementing the Microsoft Dynamics CRM system, which I deployed on a server and configured the database, so that I would have an environment to work off. From there the next step was to create

the web application and host it in Azure, a simple step using Microsoft Imagine and Visual Studio Community 2015.

The next point in the development of the application itself, was the configuration of the connection to CRM. This was done through the Webconfig file of the project, and the early bound classes were created using the CrmSvcUtil file to assist in generating the connection code to the entities needed in CRM. From there my next point of action was to configure the authentication, for which I would be using form based authentication. I created a username and password field on the form in the CRM environment, and configured the projects Webconfig file to take this into account, and set the permissions based on the valid login credentials used against the form. Another step may be to look at form based authentication on the user form also as opposed to the opportunity form, and this will be a decision I will look at further down the line.

January Reflective Journal

After reviewing the feedback given during the midpoint presentation, there were two factors that I felt needed to be addressed with the development of my application. One was the security. The connection I implemented between the Portal and the CRM deployment was merely just a simple connection string that contained login details (user name and password) for a super user. This of course is not secure as any compromise to the back end of the application would render these credentials to any prospective breach.

To combat this, I implemented the Microsoft recommended ASP.NET IIS Registration Tool to encrypt my web configuration file. This meant that the credentials used to authenticate the application with the CRM environment would be stored as a Cipher key and not plain text credentials, which would secure the application immensely and address any security concerns.

It was also noted that the GUI needed to be improved from a user's perspective. As my prototype was mostly focused on function over form, I only implemented basic web forms. I plan to style these pages using CSS to give the user a friendly and easy on the eye basis to work off of.

February Reflective Journal

February was a tricky month, as we had a lot of other assignments for different modules after our exams, so balancing the amount of college assignments I had, alongside work, was quite tricky, and I felt like I made little progress this month. One of the main things I did was research into different ways I could develop the portal. One of the main things that stood out was, for displaying lists of items/records, I could use fetchxml queries. Fetchxml is commonly used in CRM systems as a way of querying the data on the client side through the browser/interface, as opposed to doing SQL queries at a DB level, or even just building basic advanced finds. This means the exact information the user requests or needs to see can be accurately displayed.

Another possibility is introducing form based controls. This will allow dynamically updated fields to also be updated in the portal in almost real time, making sure there are no delays between say create and update.

In regards to the documentation, throughout February I reviewed the existing use cases that I had designed for the system, and several changes were made to reflect the direction of the project and the way it would be implemented going forward. For me personally, I felt it was important to update this as I progressed rather than at the end of the project, as many things could be missed throughout the process and would not be reflected correctly.

February also saw the month where the portal was deployed as a temporary measure as an IIS website. As I could not correctly deploy to Azure at this stage, I chose to host the website through the Default Website App Pool in IIS, with a localhost domain. This was to allow me to see the website work off of a static domain as opposed to localhost with just a port number, as sometimes there can be complications and mistakes when deploying.

March Reflective Journal

As the project deadlines started to approach, as well as the end of the semester and my final exams as well as the remaining projects for my other modules, I found that time management would be tough between the assignments, work and my project.

This would prove evident as little development occurred over the final two weeks of the month due to exam preparation and other projects that were due took precedence.

In regards to the project, one of the avenues I decided to further research and entertain as a possibility of inclusion in my development of the portal, was also including the Leads entity from CRM as part of the portal, offering users the ability to view Leads they have created, as well as create leads through the portal. As of the writing of this journal I have not decided whether to pursue this or not, as adding further development changes this close to the deadline could prove costly, and could skew the rest of my project. However, using the Agile methodology, I will approach the existing project as the first sprint, and if there is remaining time before the deadline, I will add an additional sprint to include the development of the Lead entity within the portal. This will also have to include writing of the particular requirements needed, use cases and an update of the project document as well as sufficient testing to make sure that it does not have any negative impact on the system.

This is where version control is very important, as being able to track each check-in or change to the code and revert to previous versions in the case of malfunction or errors can save a project from the brink of disaster, by pulling down a previous version of the code. However, this also only works if the check-ins are well documented to explain the changes, as if not it could lead to a long time spent trawling through code differences to ensure you have the correct version, and compiling several different versions could prove to be an extreme waste of time.

In regards to my documentation which from speaking with my supervisor and discussing the importance of the documentation

S, March was spent filling out the existing requirements and specifications of my Technical Report, as well as updating the existing Requirement Specifications which was included as part of the appendix to reflect changes in the project as well as what I planned to do.

Appendix 3 – Other Materials Used

Here is a screenshot of the SurveyMonkey sent out to customers/end users to scope the interest/usability of a portal within their businesses/organisations. This proved useful in regards to gauging interest and any features they would desire, and also any potential concerns they might have/predict. User feedback is extremely important

when implementing a project that is aimed at end users, particularly one that is targeting to be integrated into their every day use.

Microsoft Dynamics CRM Salesperson Portal

1. Would you feel that a lightweight portal (that is mobile based also) would help you log opportunities?

☐ Yes

☐ No


☐ Unsure

2. What would you look for in such a portal?

3. What would be your concerns (if applicable)

Done

Powered by

 **SurveyMonkey**

See how easy it is to [create a survey](#).