Stanley Uche Godfrey

Backend Programming in a Cloud Environment for Imagine Cup 2014

Helsinki Metropolia University of Applied Sciences

Bachelors of engineering

Information Technology

Thesis

Helsinki Metropolia University of Applied Sciences

Author(s)	Stanley Uche Godfrey
Title	**Backend programming in cloud environment for imagine cup 2014**
Degree	Bachelors of engineering
Degree Programme	Information Technology
Specialisation option	Software
Instructor(s)	Kari Aaltonen, Senior Lecturer

The goal of the project was to develop a web application accessible through desktop computers and mobile devices for Microsoft Imagine Cup competition. The application will enable professionals to help out registered members who are in need of their professional help but cannot afford it. For the project implementations, tasks were divided between a team of two students. The tasks were the user interface (UI) design, server side and mobile backend programming.

The user interface was designed with the Mobile First Responsive design, a technology used to make web application users' interface layouts responsive to different electronic gadget screen sizes. The task of mobile backend programming was accomplished with the Microsoft Azure. The server side development was implemented using the three-tier architectural layers which separate the Presentation, Business Logic and the Data-Access layers. The database was designed using the ADO.NET Entity Framework's code-first approach. A database with eight tables was created, when the application attempted to access data for the first time, Among the tables created were the ASP.NET Identity and Microsoft OWIN tables used for users' Authentication and Authorization. The three-tier architectural layer makes the application maintenance easy . A user can register depending on the category the user chooses to register on and can communicate with other users through the application messaging services anywhere and at any time when the users are linked.

	ASP.NET, Imagine Cup, SQL-Server, Microsoft Azure

Contents

Abbreviations and Terms

ADO.NET	A set of software that helps programmers access data and data services
Active Server Page (ASP)	Microsoft dynamic web application technology
Class	A blue print on which software objects are created
Hypertext Mark Up Language (HTML)	A standard mark up language used for creating web pages
Hypertext Transfer Protocol (HTTP)	The foundation of World Wide Web data communication
Imagine Cup	Microsoft annual best application competition for technical students
Integrated Development Environment (IDE)	A software application that provides comprehensive facilities for software development
JavaScript object Object Notation (JSON)	A format which uses human readable text to transport data

Language-Integrated Query (LINQ)	A set of Visual Studios features that extend powerful database query capabilities to C# and Visual basic programming languages syntax
Model View Controller (MVC)	A software design pattern for developing web applications
Plain Old XML (POX)	A web service used to send message
Representational State Transfer (REST)	A web service that enables exchange of data using core internet technologies
Rich Site Summary (RSS)	A format for delivery regularly changing web content
Server	A computer system that responds to across computer net works request
Structure Query Language (SQL or Sql)	Designed for management data store in relational database management system

1 Introduction

This project was carried out to participate in Microsoft Imagine Cup 2014, a competition in which students studying information technology or related courses compete for the most outstanding project in categories such as games, innovation and world citizenship. In order to take part in the competition, my team developed a web application with Microsoft tools.

The applications aim to enable any professional anywhere in the world who is willing to reach out to individuals or group of individuals who genuinely need his/her skills but cannot afford them. The application should also make it possible for a group of professionals to carry out voluntary work of interest. Less privileged people who are genuinely in need are in every part of the world, for instance a low income earner in a city, a peasant farmer in a village or a child in an orphanage. Any of these individuals with life-threatening diseases could be linked to medical practitioners willing to offer free services to these individuals.

This application aims to provide a platform for the potential mentioned in the paragraph above to be achieved. There are two categories of users, the professionals and the beneficiaries. The professionals are those users who would be willing to use their professional skills to assist the other category of users the beneficiaries, who are genuinely in need of a particular service but cannot afford that service. Both users can register; the professionals can browse through all the beneficiaries, read their profiles and if he/she can meet any of the beneficiary need, he/she notifies the website administrators which are my team members and I. The team will link the professional and the beneficiary after making necessary investigations. The beneficiary does not have access to the professionals. He/she cannot browse through the list of professionals or contact them.

My role is to write the back-end programming which will make users to interact with each other and the application administrators with the Microsoft technology, design the database for temporary storage of this application data using Microsoft sql-server database management system and move the data to the cloud where they will be store permanently.

Having worked for a couple of years as an assistant ASP.NET web developer back in my home country and passing internet software engineering courses such Data Management with good grade equips me with skills needed for my role in this project. In this paper, I will discuss the technologies used, how I carried out my part of the project and the result obtained in this project.

2 Microsoft Active Server Page Programming

The Microsoft Active Server Page was introduced in December 1996. It was developed to make dynamic web application creation less difficult and interaction between websites and users possible. Both server-side and client-side scripting languages are used in ASP web development but the common scripting language used was Visual Basic Script. When a user requests an ASP web page with any browser, the server converts the content of the page to HTML format before sending it to the browser which only interprets HTML and present the HTML interpretation to the user. Unlike HTML web pages ASP web pages have an ASP extension instead of the html. Between 1996 and 2000 four versions of ASP were released.ASP version 1 in 1996, ASP version 2 in 1997 and ASP version 3 in 2000. In 2002 ASP.NET was introduced due to low performance, difficulty to detect error during coding, lengthy scripts and hidden settings configurations associated with the ASP technology. [2]

ASP translated into ASP.NET, which makes use of the Microsoft .NET Framework. The .NET Framework incorporates technologies for developing web applications, web services, and Windows applications and makes it possible for these applications to be developed with different programming languages. Over the years several versions of ASP.NET have been released by Microsoft as a result of an upgrade of the .NET Framework by Microsoft. As stated in the immediate paragraph above, ASP.NET introduced in 2002 was ASP.NET 1.0. It provides object orientated programming features and boosts safety and early binding. The next version, ASP.NET 1.1, released in 2003, added features such as automatic validation of input data as well as mobile controls. In November 2005, ASP.NET version 2.0 was released. This version has new features such as Form View data control, Grid View data control, Details View data control, master pages and other customization features. ASP.NET version 3.0 released in 2006, introduced features such as Windows communication and presentation foundation, Windows card space and workflow foundation. In November 2007 ASP.NET

version 3.5 was released and it has the List View control usage capability, Data Pager usage capability, syndicate feeds and HTTP pipeline features as well as JSON, RSS, POX and Partial Trust. In August 2008 ASP.NET 3.5 service packs 1 was released. This version made the dynamic data incorporation for Ajax browser history management more efficient, and introduced System.web.routing and System.web.abstractions namespaces.[3,4]

ASP.NET version 4.0 released in 2010 introduced core services, which made it possible for developers to indicate their targeting framework in the web configuration xml file, extensible request validation, improvements on web forms controls, three different project templates and MVC. The latest, which is the ASP.NET 4.5 released in October 2013, improved existing parallel computing features and introduced new ones and also introduced new programming interface for HTTP applications, and improved on the Windows presentation, communication and workflow foundations features. Though there are several similar server-side technologies competing with ASP.NET, yet ASP.NET stands out with the following peculiar characteristics. [3, 4]

2.1 ASP.NET Features

As part of the Microsoft .NET Framework ASP.NET makes the most of object-orientated programming features available in the .NET framework. Its web pages are objects with different characteristics. For instance the web form which is ASP.NET. The most used development method has on any of its pages programmable objects and events. Figure 1 show a typical model of the ASP.NET web form.

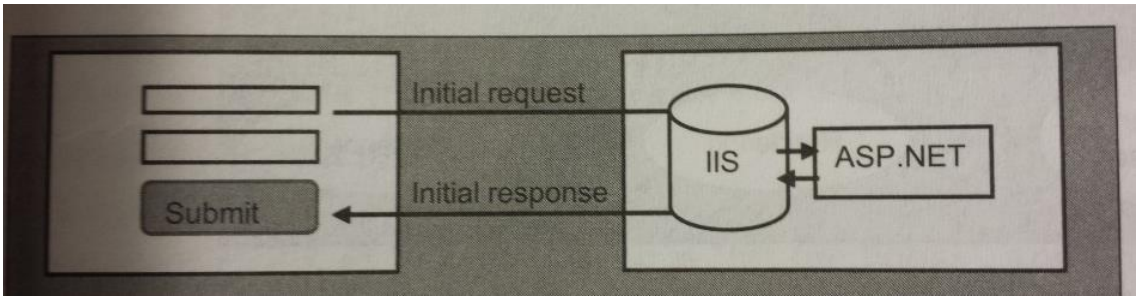

Figure 1 ASP.NET web form with button and text boxes. Data modified from Bochicchio,Mostarda and De Sanctis (2011). [4,5]

Figure 1 shows a web form with a submit button which is programmed so that when it is clicked by a user, the button's click event is caught just as it is done in desktop application development. [4, 5]

An ASP.NET page has server controls which are so called because they are the server-side programmable part of a page. The page is made up of two parts the front end, design or mark-up and the backend or the page with code. A server control can be added using the mark-up or programmatically at the code-behind page. To execute an ASP.NET web form or any of its applications, browser is needed for rendering. What happens is that HTML is generated and a web server runs it. Figure 2 illustrates request and response actions associated with the ASP.NET web application.

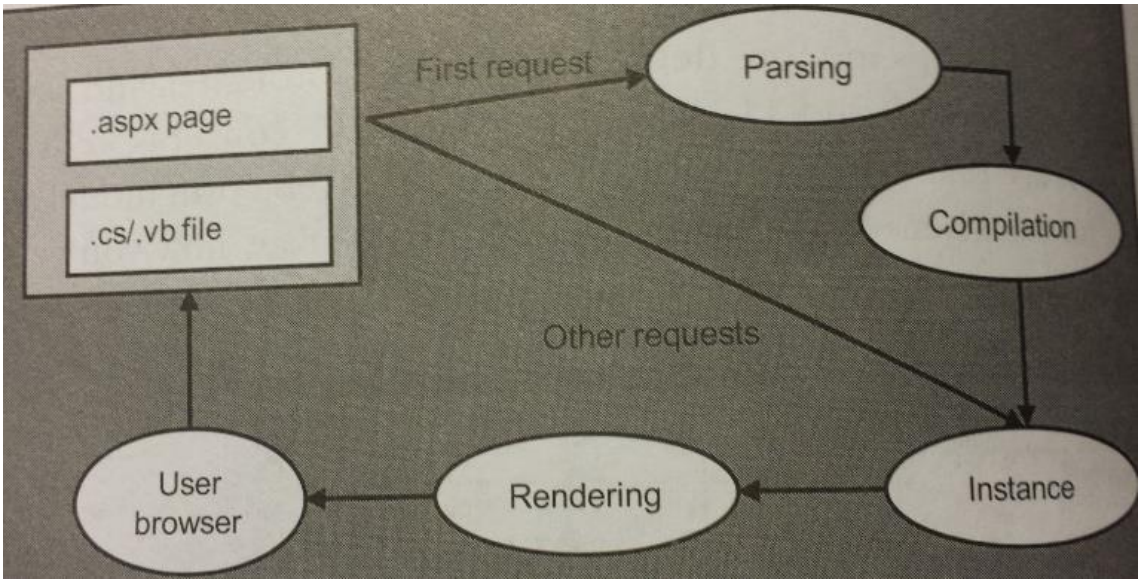

Figure 2 Request and response workflow of the ASP.NET web application. Data gathered from Bochicchio,Mostarda and De Sanctis (2011). [4, 6]

Figure 2 shows a typical ASP.NET web application request and response process. From figure 2, it is easily seen that ASP.NET offers a transparent page compilation mechanism. When a web page is requested, if it is its first request, the web application is first compiled to a disk, then instantiated, rendered and served in a browser as HTML. The developer is not aware of the whole process and if there is any modification ASP.NET automatically updates the web application. [4, 6-7]

ASP.NET application architecture

An object oriented feature of ASP.NET makes building reusable components possible. It also makes avoiding redundancy achievable. To begin ASP.NET development, a developer needs to understand how the Object-Oriented Programming (OOP) concept and architectural patterns are utilized. Architecture is vital and a good one must be provided for robust ASP.NET application development. In the ASP.NET web application, the three-layer architecture is usually used. For example, the web application my team is developing, connects people from different places, one of the actions this application has is to show the list of people in need of help a particular professional can provide, another is to give details about a particular beneficiary. To show the results of these actions a web page that extracts the data from data storage system is needed. Then the storage system is made to be in synchronization with the code behind. For this example, the web application should be separated into three layers. The first layer retrieves the data from data storage system. The second layer represents data in an object-orientated fashion and acts as interface between the first and the third layer displays data from the storage system which, is the SQL server management system for this application. Figure 3 illustrates this three-layer architectural model. The first layer is known as Data Access Layer, the second layer is called the Business Logic Layer, while the third layer is referred to as the Presentation Layer.

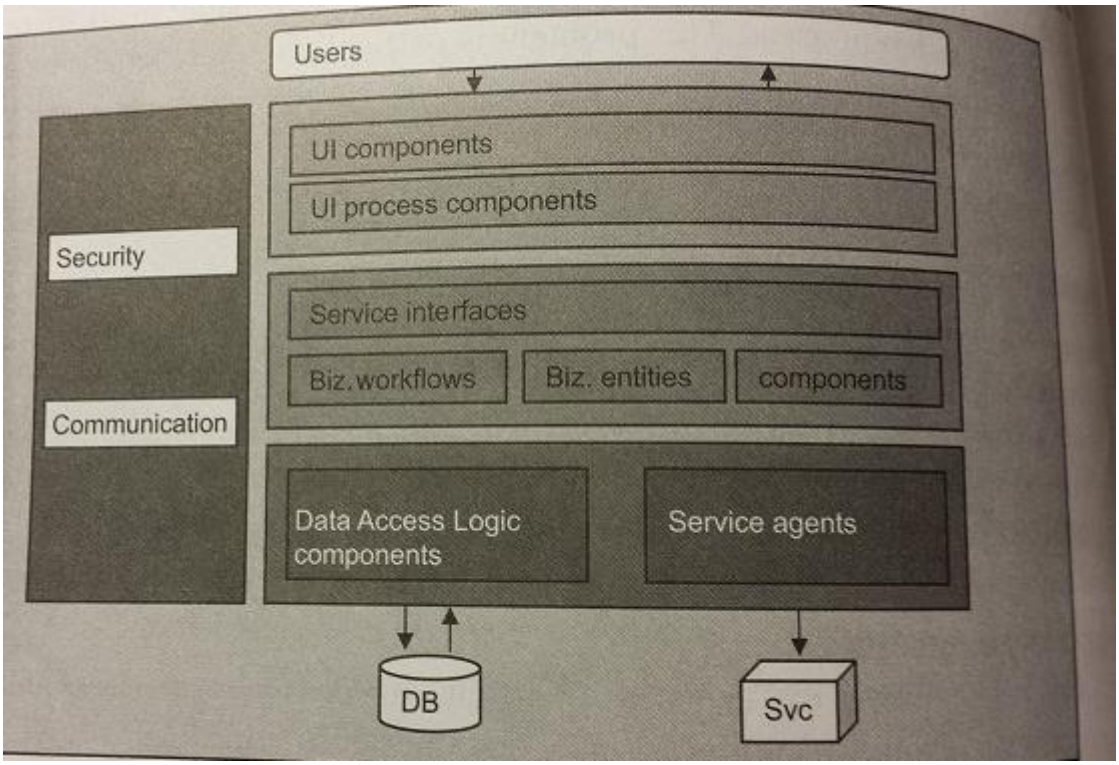

Figure 3 ASP.NET three layer application schema. Data gathered from Bochicchio,Mostarda and De Sanctis (2011). [4, 8]

Figure 3 shows that the Data Access layer interacts with the database storage system. The Business Logic represented by the second frame contains rules to be implemented in order to get the application's business needs met. The presentation layer represented by the first frame interacts with users of the application.

The three-layer architecture must not be used in all ASP.NET web applications but it is the most common architecture used for standard ASP.NET web application development because of its simplicity. However there are situations in which a two-layer architecture is preferred, and for complicated ASP.NET web application an n-layer architecture could be implemented. In a typical multilayer layer architecture, objects are exchanged between layers. [4, 8-9]

2.2 .NET Framework 4.0

The Framework is a collection of software which offers the general functions needed by a programmer to develop software with the desired functionalities. Sharing the same background is an important key to consider when choosing a framework and ASP.NET makes the most of other technologies in the .NET Framework. The .NET Framework provides consistency across all applications, from web applications to services, Windows application and even mobile applications. Figure 4 shows different technologies that can leverage on the .NET Framework 4.0

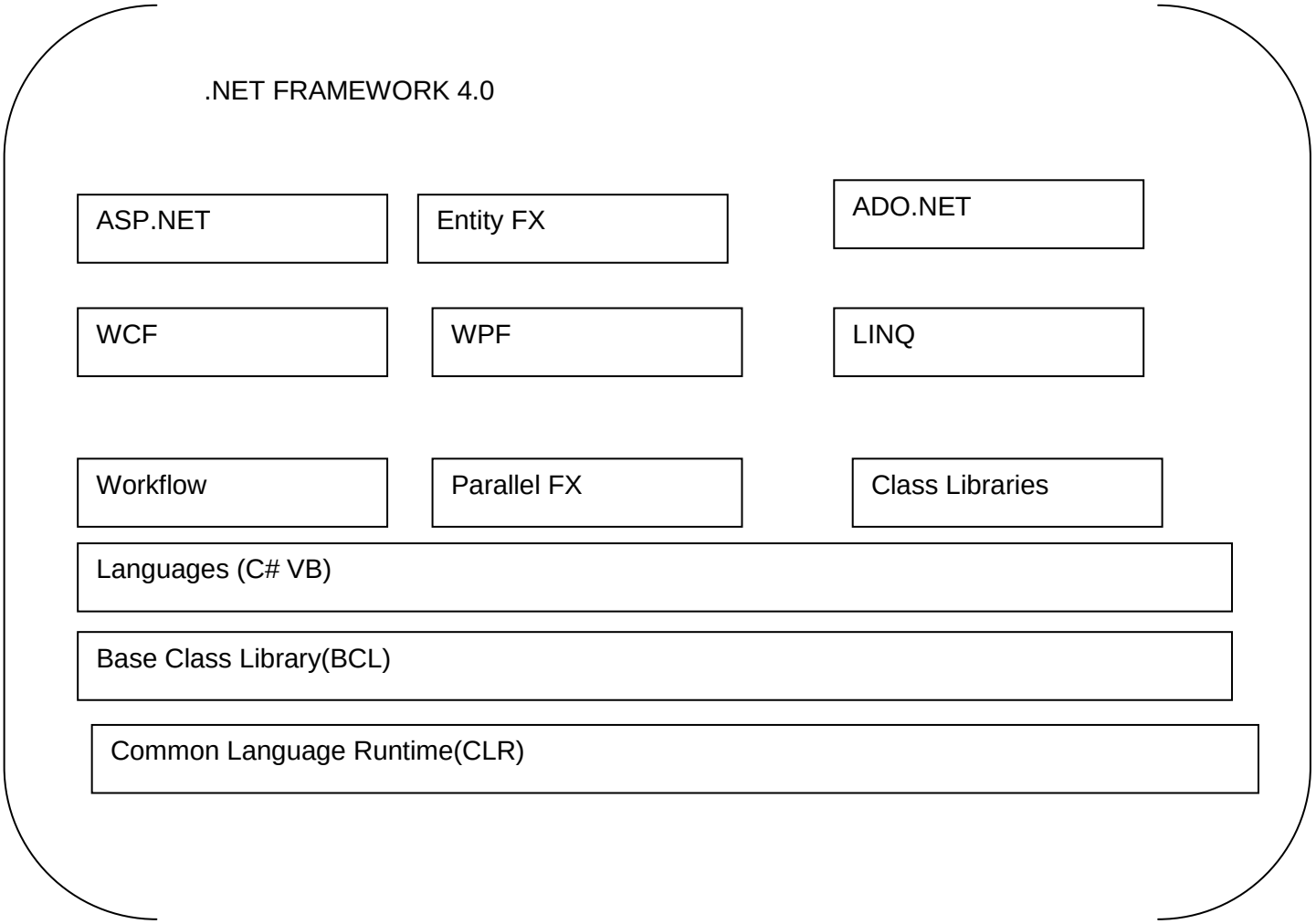

Figure 4. Technologies available in the .NET Framework. Data gathered from Bochicchio,Mostarda and De Sanctis (2011). [4, 18]

Figure 4 clearly shows that ASP.NET is part of .NET Framework; it also shows other technologies that are part of .NET Framework. Developers can leverage on one of the technologies while developing an application of another technology. For instance this project is an ASP.NET web forms but Entity Framework and LINQ technologies were incorporated into it for creating database, accessing data from database, and for the overall management of the database. There are possibilities to make the most of other technologies in the .NET Framework when developing an application of one of its technologies because the underlying compliers, runtime and class library share same components. The .NET Framework is a reusable software platform for developing reusable software and abstraction level in its environment is high. Because of .NET Framework

reusability and high abstraction level features, developers turn to it to minimize the length of code written during application development and shorten the period of the application development as well. [4, 17-18]

Features of the .NET Framework include Cross-Language Integration (CLI) which enables an application written in one language to use the functionalities of another application written in a different programming language. For instance an application written in C# (one of the .NET Framework's programming languages) can call functionalities of one of .NET Framework's libraries written in Visual Basic (another .NET Framework's programming language). Common Language Runtime (CLR) is a .NET Framework feature that manages application memory, handles exception and collects garbage automatically. It also contains the Just-in-Time complier for compiling and execution of programs. Portability is a .NET Framework feature which makes a program compiled with CLR able to run in another machine different from the one that compiled it. Garbage collection is a .NET Framework feature which is responsible for reclaiming memory space occupied by unused objects. Code Verification is another feature of the .NET Framework that enforces security by verifying the code before execution. Assemblies are one of the prominent features of .NET Framework; they are what make deployment, reusability, versioning and security possible. Assemblies consist of metadata which contains identity information, culture information, type, dependencies and security information. [4, 18]

.NET Framework 4.0 has many programming languages developers can choose from when developing an application with it. It also has a Dynamic Language Runtime (DLR) which calls dynamic languages such as Iron Python and Ruby from managed code. For ASP.NET applications C# ("pronounced c sharp") and visual basic (VB) programming languages are used. The VB has similar functionalities to C#. C# which was used for coding the back-end of this thesis project, is an object-oriented programming language which means its components can be reused. It also executes programs at runtime, and supports multiline statements as do Java programming and VB programming and most C-based programming languages. The syntax of C# is much like that of Java though it inherits from C++ another object oriented programming language. To run an application written in C#, .NET Framework uses its Common Language Runtime (CLR) to manage memory, perform garbage collection, and handles exceptions. It also converts the program to machine language with the assistance of the Intermediate Language (IL) and the just-in-time complier. Many objects are available for used by C# programmer in the

.NET Framework class libraries. Regardless of the programming language a developer prefers, the entire features of the .NET Framework are still accessed and performance is not compromised in anyway. [4, 18]

2.3 Data Access in ASP.NET

.NET Framework provides ADO.NET technologies which are used to manage data access and represent data in the ASP.NET application and other .NET Framework applications. Developers have the options offered by ADO.NET when it comes to data access and representation. One of the choices a .NET developer has is to use ADO.NET objects, such as connections, adapters, readers and dataset. This approach is straight forward and easy to understand and it enables .NET beginner developers to start writing ADO.NET codes instantly. The major drawback of this approach is that developers write lengthy code to achieve their objective. Another option available in ADO.NET technologies is the Entity framework which is used for data access and representation in this project and also for database creation in SQL server database management studio as well. The latter option uses ADO.NET classes to communicate with database and then create classes to represent data in the application. Understanding this latter option is not easy but if well understood, in the long run its maintainability is easier than the former option. [4,30-32]

The latter option uses Object Relation Mapping (ORM) which makes complexity of using ADO.NET classes obscures and makes developers work with their application objects. This ORM tools includes the features of the former option due to the fact that it provides immediate and sustained productivity. The Entity framework is Microsoft's ORM; it was announced to be Microsoft's best method of data access. The Entity framework is used for data binding, authentication and performance. In order to use the Entity Framework, developers need to create an internal network of classes caledl the object model that stores data fetched from database when the database is queried. These classes can also be used to update the database. The object model classes have the same characteristics as the data in the database. The object model classes make the complexity of the structure of the database invisible from the business Logic Layer code. The model classes allow the Business Logic Layer code to interact with them and any Data Access layer that is responsible for database interaction. The Business Logic Layer is not aware of the database but communicates with the model classes only. With structure as the one described above in place the Business Logic Layer,

Data Access Layer and Database become what is known as the Business Logic Layer database. Figure 5 illustrates the design of such a database. [4, 30-32]

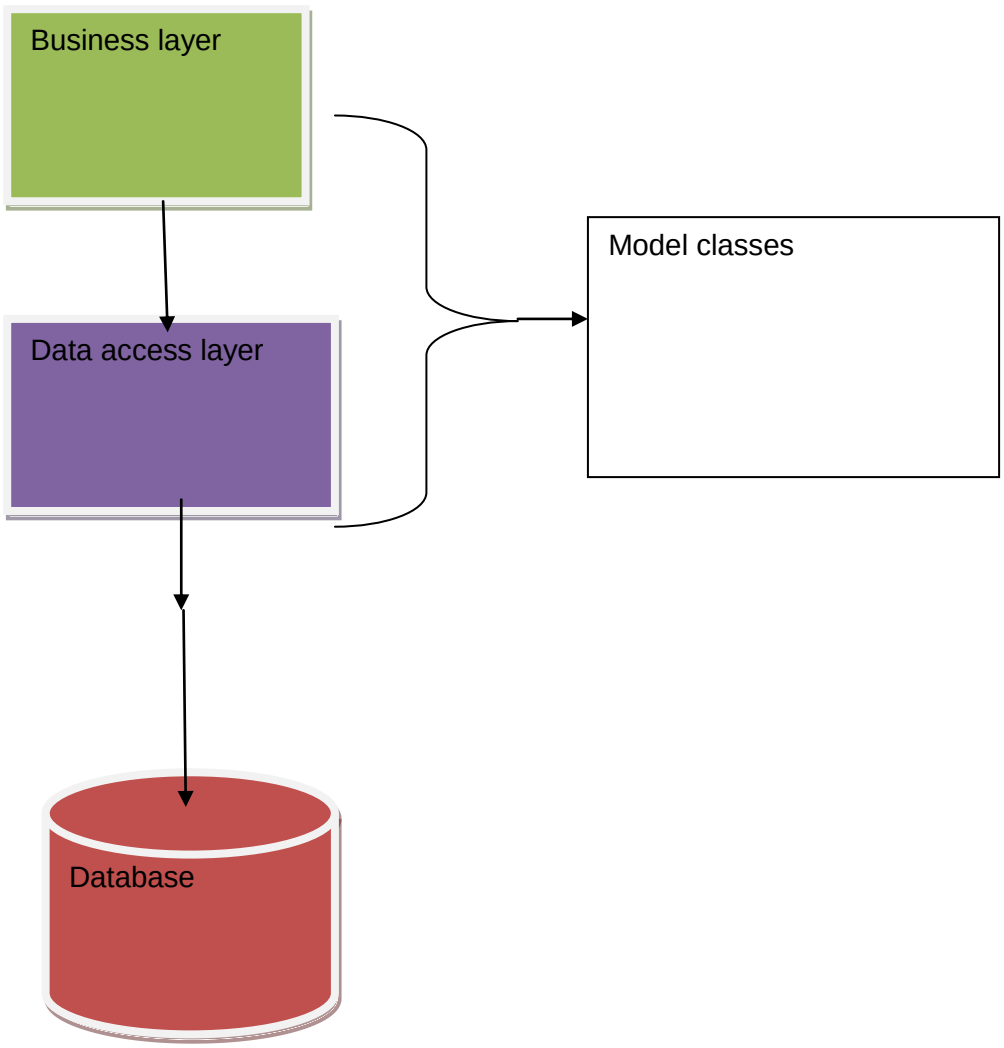

Figure 5. Business Logic Layer database design.

Figure 5 illustrates that the Business Logic Layer makes use of the Model classes but does not directly communicate with the database. Separating the code in different layers is a means of ensuring that application development is faster and easier to maintain which is crucial. [4, 30-32]

The Object Model

The object model is a collection of classes containing data which are fetched from database and have not many characteristics. Some of the features of an object model includes the following.

Data Validation

One of the characteristics of the object model is the identity property, because it is what makes the class unique, the identity property of a model cannot be null or empty. It is liken to the primary key of the table represented by the object model in the database.

Property Joining

Object model class has a joining property which internally joins related properties and returns them without programmers' explicitly writing code to retrieve such related properties.

Connecting Classes to Each Other

Object model classes are not standalone; they connect to one another just as tables are connected by foreign keys column in databases. However using a property that behaves the foreign key column of the database is not efficient. Developers can directly reference another class with a property of the referenced class type. For instance in this application, there is a Professional class and a Beneficiary class. The Professional class has a property of a type Beneficiary when Beneficiary is to be referenced in the Professional class it can be done with the property of the professional class which is of type Beneficiary. More details about this will be given in chapter six.[4,32].

3 Cloud Computing

Cloud computing is a way of computing which delivers IT-enabled capabilities that are scalable as well as elastic. It could also be referred to as a standardized IT capability which includes services, software or infrastructure delivered via Internet technologies in a pay per use, self-service way. Concisely, cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning of scalable and measured resources. The term “cloud computing” became a commercially used term in 2006, that same year Amazon launched its Elastic Compute Cloud (EC2) services that organizations can lease for computing capabilities and processing power to run their enterprise applications. Google Apps started making available browser-based enterprise application that same year and three years later Google App Engine emerged to be another historic achievement. Some of the reasons why cloud computing came into being are explained below. [5, 26-28]

Basic Concepts and Terminology

The terms in cloud computing are explained in this section are:

The Cloud

A cloud is used to refer to the various IT environments put in place to provide scalable and measured IT resources. The term originates from the description of the Internet as being a network of networks which makes remote access to decentralized IT resources possible. Before cloud computing came into being as a formal arm of the IT industry, the symbol of a cloud was mostly used to represent the Internet in documentations of many varieties and specifications. The same symbol is now used to represent the cloud computing environment, as figure 6 shows.

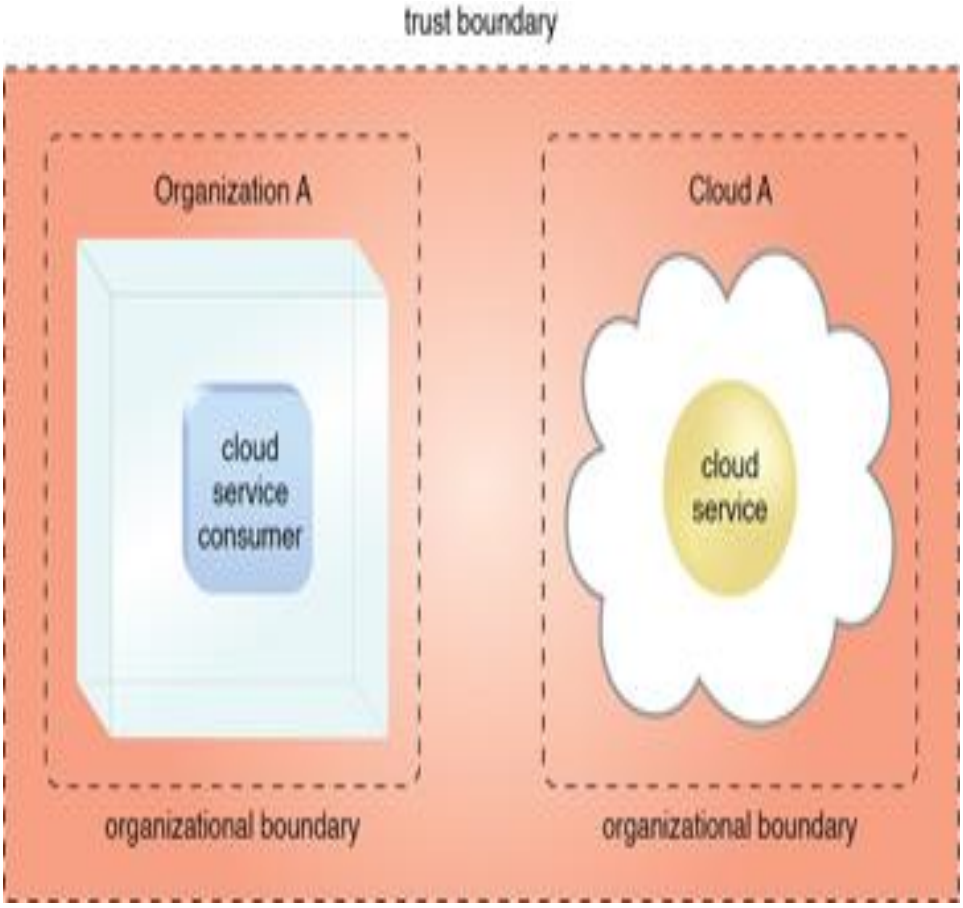

Figure 6. Symbol used to represent cloud computing boundary. Copied from Zaigham Mahmood and Ricardo Puttini [5, 33]

It is crucial to differentiate the term “cloud” and the cloud symbol from the Internet as an environment for providing remote access to IT resources, a cloud has set boundaries of which many are accessible through the Internet. While the Internet provides access to numerous web-based IT resources. It should be noted that a cloud is usually privately owned, providing measured access to IT resources. [5, 33-34]

Moreover, a large portion of the Internet is for accessing content-based IT resources published through the World Wide Web. In contrast, IT resources provided by the cloud environment offer back-end processing functionalities and user-based capabilities to access these functionalities. Though cloud computing is based on Internet protocols (a means by which computers in a certain structure interact with each other in pre-defined manner) and technologies, yet they are not web-based but are based on any protocol that supports remote access to IT resources. [5, 34]

3.1 IT RESOURCE

IT Resources are hardware-based, software-based or virtual IT-related manmade objects such as servers and network devices. Figure 7 illustrates the various symbols representing some IT resources used in cloud computing.

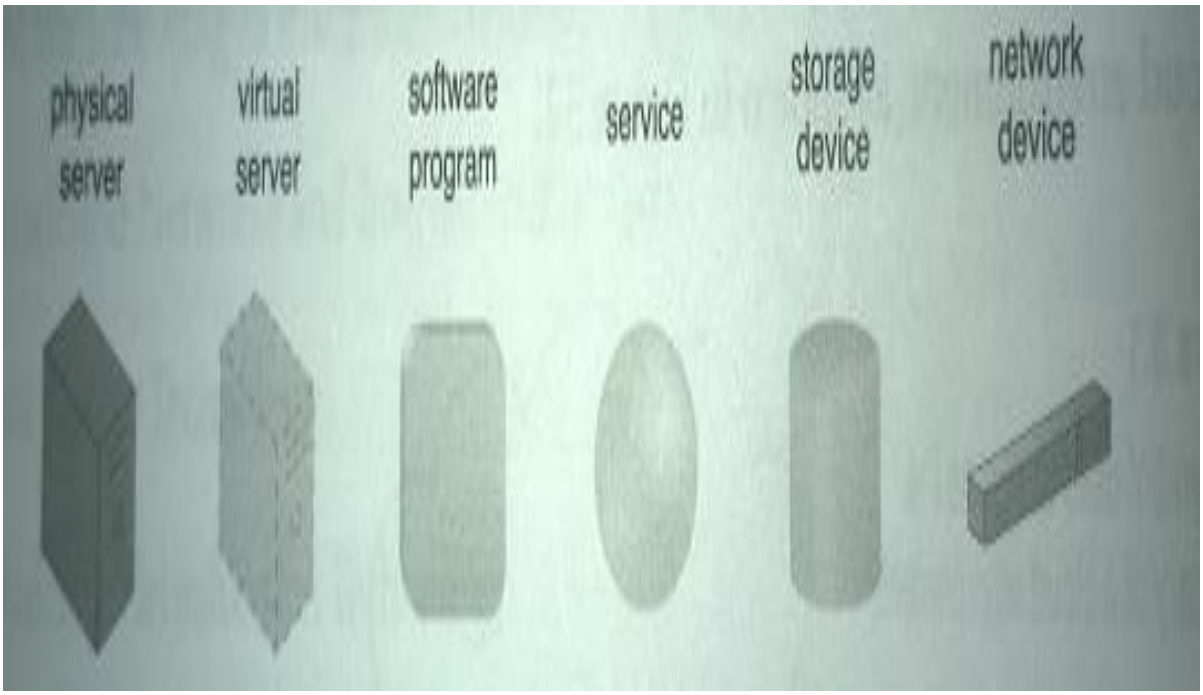

Figure 7. IT resources and their corresponding symbols. Copied from Zaigham Mahmood and Puttini [5, 34].

Figure 7 shows various symbols representing IT resources used in a cloud computing environment. A cloud-based environment which host and provides a set of IT resources can be defined as shown in figure 8.

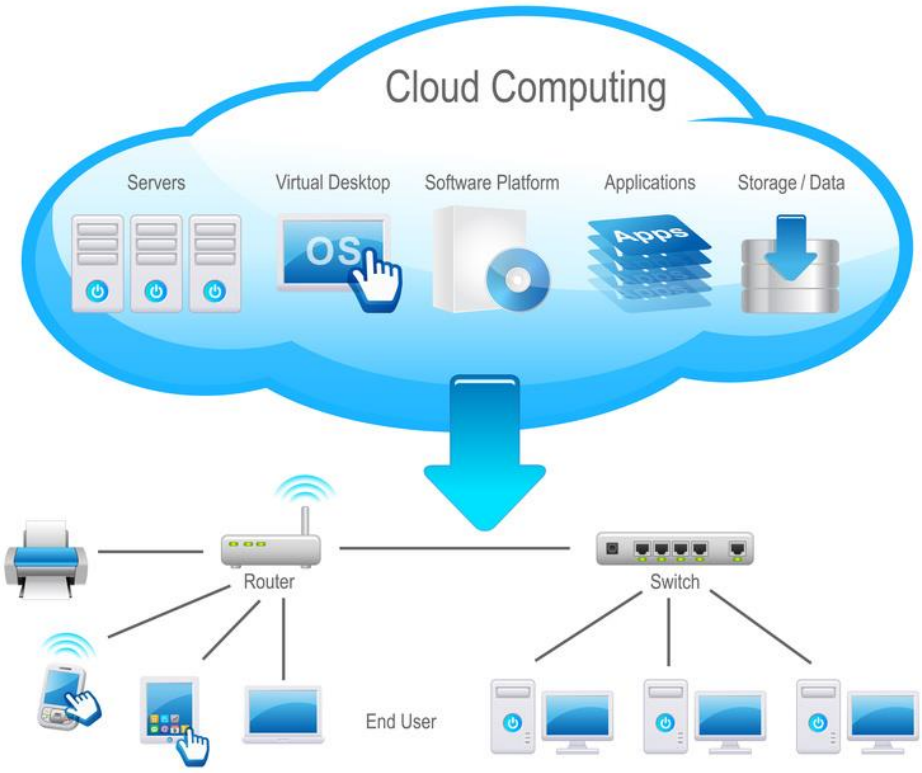

Figure 8. A cloud environment hosting various IT resources. Copied from Zaigham Mahmood and Puttini [5, 35]

Figure 8 shows a cloud hosting various IT resources which are accessed via the internet. The shown IT resources are referred to as cloud-based IT resources. Technology architectures and numerous communication occurrences which IT resources engaged in are also shown in figure 8.[5,35]

Cloud Consumers and Cloud Providers

The cloud providers are responsible for providing cloud-based IT resources while the cloud consumers use the IT resources provided by cloud providers. The terms are used to refer to roles assumed by organizations as regards cloud, cloud provision and contracts. [5,36]

3.2 Scaling

The ability of the IT resources to manage increase or decrease in the usage demands of the IT resources is called scaling. It is categorized into Horizontal Scaling and Vertical Scaling. The allocating and realising of IT resources that are of the same type is

known as horizontal scaling. The horizontal allocation of resources is called scaling out while horizontal releasing of resources is called scaling in [5,37].

Figure 9 shows a horizontally scaled out resources.

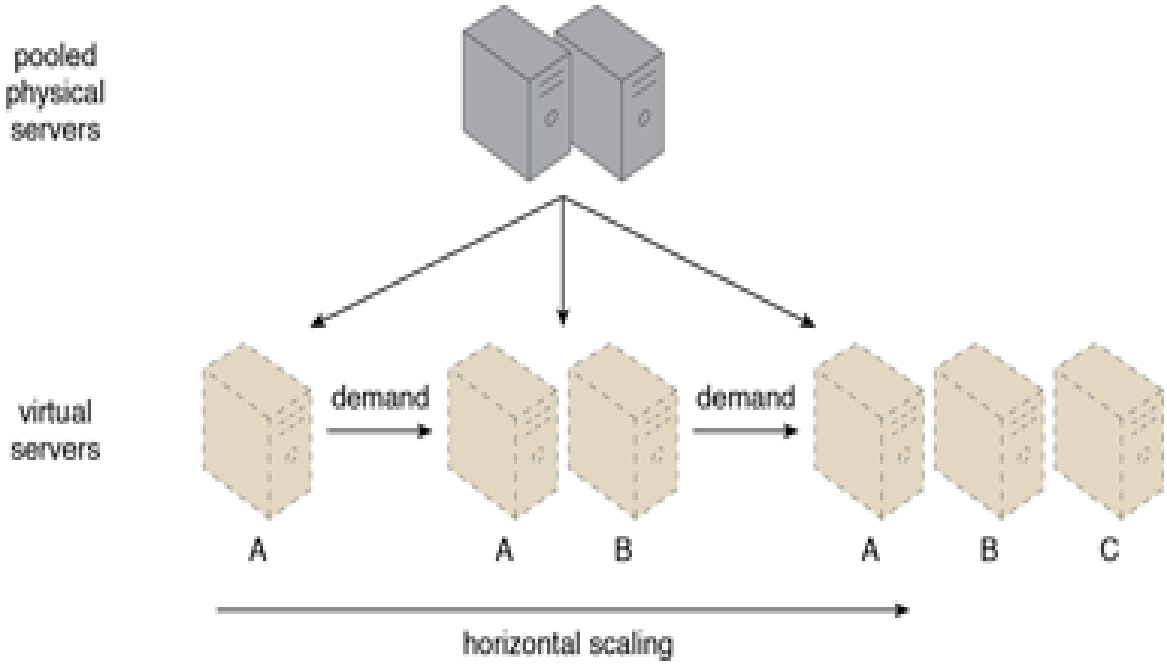

Figure 9. IT resources, Virtual Server A scaled out by adding Virtual server B and Virtual server C. Copied from Zaigham ,Mahmood and Puttini [5, 37]

Figure 9 shows how IT resources can undergo horizontal out scaling in a cloud environment.

Vertical scaling occurs when an existing IT resources is replaced by another of higher or lower capacity. The replacement of IT resources with another of higher capacity is known as scaling up and when an existing IT resource is replaced by another of lower capacity it is referred to as scaling down. Figure 10 illustrate an example of up-scaling IT resources in a cloud environment.

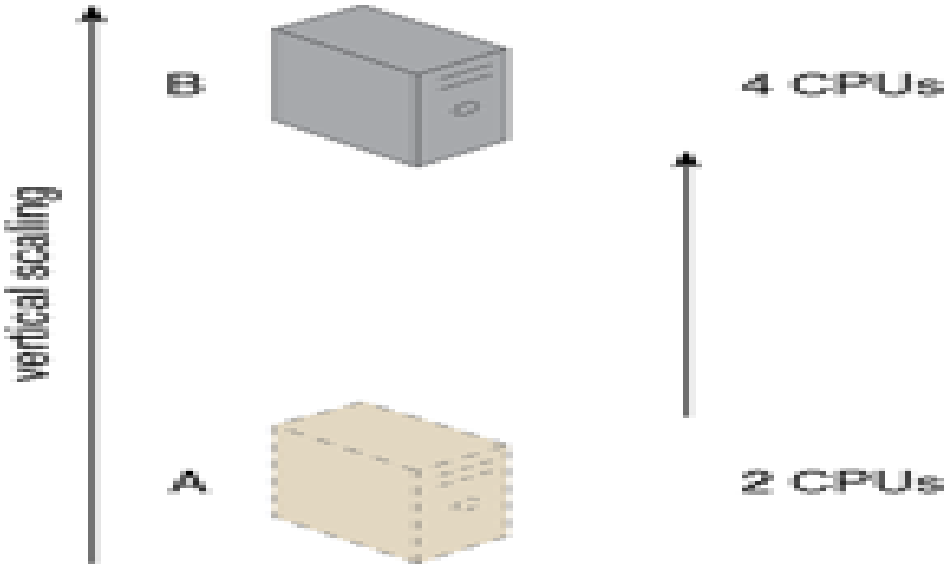

Figure 10. IT resources virtual server with two CPUs are scaled up by replacing them with a Physical server of four CPU. Copied from Zaigham Mahmood and Puttini [5, 38]

Figure 10 illustrates an example of vertical scaling up of IT resources in a cloud environment.

The advantages and disadvantages of the two types of scaling IT resources in a cloud environment are summarized in table 1 below.

Table 1. A summary of advantages and disadvantages of horizontal and vertical scaling. Data gathered from Mahmood and Ricardo Puttini [5, 38]

Horizontal Scaling	Vertical Scaling
Cost less, achieved via commodity hardware components	Cost more
IT resources are available immediately	IT resources are also available instantly
Resource replication and automated scaling	Additional setup is required always
Additional IT resources is mandatory	No addition IT resources required
Cannot be restricted by hardware capacity	Limited by maximum hardware capacities

Table 1 illustrates a summary of comparison between the horizontal scaling and vertical scaling of IT resources in cloud environments.

Capacity Planning

Capacity planning is a method of predicting the future needs of an organization's IT resources, products and services and making sure that these needs are met. This is the maximum amount of work which IT resources can do in a specific period. Differences between the capacity of an IT resource and its demand can make the system either not to work to its full capacity or unable to meet user needs. Capacity planning reduces differences of this kind to accomplish efficiency and performance that are predictable. [5, 28]

Cost Reduction

To ascertain the cost of IT infrastructure and the performance of a business is not easy. The optimal utility needs are sometimes accessed to determine the growth of an IT environment thus leading to the support of increase in investment on scalable and new business automation. The needed investment is put into expanding infrastructure because of the limitation of a given automation solution in terms of usage which is due to its processing power strength. [5, 29].

However the cost of putting new infrastructure in place and that of the one already available need to be accounted for, as well as the cost of operational overhead of the infrastructure which is a significant share of IT budgets. the Operational overhead could be technical staff needed to make the surroundings operational, upgrades and patches that leads to more testing and deployment cycles, expenses on power and cooling bills, security and access control measure needed to maintain and protect infrastructure resources and salaries of personnel needed to see to licensing and support of technical staff. Maintaining an organisation's technology infrastructure and ownership comes with the huge responsibility that affects significantly the budget of cooperate organizations. The IT arm of organizations are most times a serious financial problem affecting the profitability and overall growth of the organization. [5, 29-30].

Organizational Agility

Businesses should be flexible and respond to changes positively. For IT companies, responding to a business change is by expanding the IT infrastructure to be more than what was estimated initially. However there could be limitations in the infrastructure which makes it impossible for organizations to respond to such change if the initial ca-

pacity was not implemented due to inadequate finances. Furthermore, altering business requirements or priorities might require an increase in IT resources and efficiency. Even when an organization can handle an increase in IT resources and efficiency demands, the way which the IT infrastructure is utilized could cause exceptions that could bring the operations of the infrastructure to a halt. Because of lack of reliability controls within the infrastructure, the customers and clients may react in such a way that continuity of the organization is threatened. Generally, up-front investments and infrastructure ownership costs needed for a new or expanded business automation solution could be unaffordable to some businesses making them go for inferior IT infrastructure which decreases the business chances of meeting real-world requirements. [5, 30]

In another case, businesses do not go for an automation solution altogether when upon review of their infrastructure budgets it is noticed that the cost is prohibitive. It is a situation that can make organizations not to keep abreast of the current market demands, competitive pressure and sometimes set business goals. [5, 30]

Cloud Service

Cloud is no doubt a remotely accessed environment. Nevertheless some IT resources in a cloud environment cannot be accessed remotely. For instance a database or server in a cloud environment could only be accessed by other IT resources within that same cloud environment. For the purpose of accessibility of IT resources in cloud environments a software program and some API (Application Programming Interface) may be deployed to make sure that IT resource in a cloud environment remote access is not hindered. Such a software programme is called cloud service. [5, 39]

However the term cloud within the cloud computing context is broad, a cloud service could mean a web-based software program with a technical interface brought about through the use of a messaging protocol. [5, 39]

The idea of cloud computing is to make available IT resources as services that comprise other IT resources providing functionalities for clients to use and take advantage of remotely. Numerous models for generic types of cloud services have come into existence and majority of them have the “as-a-service” suffix. Among them is “Mobile as-a-backend-service” a cloud service this project utilised for accessing data when it this project was run with mobile devices. Chapter five concentrates on “Mobile back-end as a service”. [5, 39].

4 Mobile Back-end as a Service

Mobile Backend as-a-service (MBaaS) is one of the cloud computing technology services. It is a special type of Platform as-a-service (PaaS) which is a cloud service in which providers not only provide the infrastructure but configuration platform also. Mobile Backend as-a-Service could be defined as a clouding computing model enabling mobile and web application developers to store their application data into the cloud, manage users, interact with social network services and push notifications. MBaaS resolves problems posed by the server side infrastructure management, mobile and web application boom and the need to develop mobile and web application within a short period of time. [6]

MBaaS Services

The two general services MBaaS providers provide are cloud-based data storage facility and custom Software Development Kit (SDK) used for making Create, Read, Update and Delete (CRUD) operations on application data. The outstanding characteristic that makes MBaaS stand out among other cloud services is its ability to connect mobile and web applications to the cloud services with a unified connection. It saves developers time, effort and knowledge needed for creating web service backend database and middleware. [6]

Most recent mobile and web applications are socially integrated. This integration plays a vital role in virtually in everyday mobile and web applications. MBaas providers ought to handle this integration else the daunting task of maintaining a different number of sets of API to perform a particular task in a web or mobile application will fall on the developers lap. A good example is the provision of the Open Standard for authorization (OAUth) by MBaas providers which enables developers to give users the option of using their Facebook or Twitter login credential for authentication without compromising security.[6]

One of the important features of a mobile application that makes it a success is the push notification. It is a fact that mobile and web applications have been developed and distributed across different platforms recently. With this in mind, a central engine that

sends push notifications to application across a multiplatform is leverage to developers. Otherwise such push notifications can only be achieved by writing different blocks of code for the different platforms. However MbaaS provides a single unified mechanism which uses the push notification services to send push notification to any application regardless of the platform.[6]

MBaaS also offers the user analytics, used for application tracking to know how many hits have been made to an API and the region which these hits are made from. When access to information of this kind is made accurately and timely it will help the developers in the developers' duty of making sure the application is successful and keeping track of the way the application is used. [6]

Feeding data into a system could be a daunting and overwhelming task, sometimes leading to application processing time to slow down and making data importation and exportation an issue. Nevertheless, MBaaS provides a way of data importation and exportation from data sources which has no effect on the processing speed of an application. [6]

Apart from these basic features, MBaaS providers are differentiated by services they render. Mind maps from QuickBlox which is a major player in MbaaS at the beginning of year the 2014 announced some sets of services render by some MbaaS provider. From the announcement Mind-maps made, one can deduce that other services offered by different MBaaS providers range from messaging server , interface design, server side custom logic, integration email, chart functionality and shopping cart. [6]

MBaaS Ecosystem

The MBaaS ecosystem according to Kinvey, one of the MBaaS major players, is increasing. Kinvey released an ecosystem map in January 2013. The map is concise yet the right representation of the the MBaaS ecosystem. Analysts believe that having 40 or more MBaaS providers means that there is no unique service that differentiates one provider from another considering that the basic service a developer needs is the cloud storage capability, custom SDK for multiple platforms and automatic RESTful API generation and every MBaaS provider renders these three services. Analysts also believe that it will take some time for MBaas providers to be distinguished based on

their services since the technology is a relatively new one and it is still growing and maturing. [6]

Choosing the Right MBaaS Provider

With over 40 providers offering similar services it is confusing to choose one MBaaS provider from a set of MBaaS providers. However if a thorough analysis is made, a way of making a good choice can be figured out. [6]

An on-premises set-up is ideal for an enterprise with sensitive data and servers for the purpose of storing their data. In this case an MBaaS provider offering an on-premises service could be contracted for on-premises set up. For start-ups, cloud-based offering will be advisable. Many enterprises are reluctant to patronize MBaaS providers it is usually the start-ups and independent developers that are MBaaS major clients. [6]

Analysts advise that the choice of an MBaaS provider should be made, by considering which of the sets of MBaaS providers allow developers to make the most of their creativity without worry much about server-side management. [6]

5 Programming Phases of the Application Back-End

There are different stages of backend programming of an ASP.NET web application. Visual studios 2013, which is the IDE (Integrated Development Environment) used for the entire project and the application programming processes are accomplished in the following phases

Database and data objects creation

Code-first approach of the Entity framework database was adopted, KHDBContext class which extends Database Context, a class used for communicating with data as an object was first created as listing 1 shows.

```
public class KHDBContext:DbContext
   {
        public KHDBContext(): base("CaringHearts")
        {

        }
        public DbSet<Beneficiary> Beneficiaries { get; set; }
        public DbSet<Donor> Donors { get; set; }



    }
```

.

Listing 1. CaringHeart Context class.

The class KHDBContext represents the project database in an object form while the KHDBContext properties Beneficiaries and Donors represent the database Donor table and Beneficiary table and the project Donor and Beneficiary objects. The Donor and Beneficiary objects and their fields which represent the Donor and Beneficiary tables and their columns in the Sql-Server 2012 Management studio, were created using listing 2 and listing 3 below.

```
public class Donor
    {
        [ScaffoldColumn(false)]

        public Guid ID { get; set; }
        [Required, StringLength(50)]
        public string FirstName { get; set; }
        [Required, StringLength(50)]
        public string LastName { get; set; }
        [Required, StringLength(50)]
        public string UserName { get; set; }
        [Required, StringLength(128)]
        public string Password { get; set; }
        [Required, StringLength(50)]
        public string Category { get; set; }
        [Required, StringLength(50)]
        public string Country { get; set; }
        [Required, StringLength(50)]
        public string State { get; set; }
        [Required, StringLength(50)]
        public string StreeAddress { get; set; }
        [Required, StringLength(50)]
        public string PostCode { get; set; }


        [Required, StringLength(50)]
        public string Sex { get; set; }
        [Required]
        public string DOB { get; set; }
        [Required, StringLength(50)]
        public string Profession { get; set; }
        [Required, StringLength(250), DataType(DataType.MultilineText)]
        public string AboutYourSelf { get; set; }
        [Required, StringLength(250), DataType(DataType.MultilineText)]
        public string Motivation { get; set; }
        [Required, StringLength(50)]
        public string Term { get; set; }

        [Required, StringLength(50)]
        public string Email { get; set; }
        [Required, StringLength(50)]
        public string MobileNumber { get; set; }
        [Required]
        public Byte[] Pix { get; set; }

    }
```

Listing 2. Donor object class which representing the Donor table in the caringHearts database.

```
public class Beneficiary
    {
         [ScaffoldColumn(false)]

        public Guid ID { get; set; }
        [Required, StringLength(50)]
        public string FirstName { get; set; }
        [Required, StringLength(50)]
        public string LastName { get; set; }
        [Required, StringLength(50)]
        public string UserName { get; set; }
        [Required, StringLength(128)]
        public string Password { get; set; }
        [Required, StringLength(50)]
        public string Category { get; set; }
        [Required, StringLength(50)]
        public string Country { get; set; }
        [Required, StringLength(50)]
        public string State { get; set; }
        [Required, StringLength(50)]
        public string StreeAddress { get; set; }
        [Required, StringLength(50)]
        public string PostCode { get; set; }
        [Required, StringLength(50)]
        public string Sex { get; set; }
        [Required]
        public string DOB { get; set; }

        [Required, StringLength(250), DataType(DataType.MultilineText)]
        public string AboutYourSelf { get; set; }
        [Required, StringLength(250), DataType(DataType.MultilineText)]
        public string Challenge { get; set; }

        [Required, StringLength(50)]
        public string Email { get; set; }
        [Required, StringLength(50)]
        public string MobileNumber { get; set; }
         [Required]

        public Byte[] Pix { get; set; }
```

Listing 3. Beneficiary object code snippet.

The authorization and the authentication tables were created at runtime. These tables include MigrationHistory, ASP.NET Roles, ASP.NETUserClaims, ASP.NETUserLogins, ASP.NETUserRoles and ASP,NETUserRoles tables. The creation of these tables were made possible by the inclusion of listing 4 in the configuration set tag of the application's the web.config file.

```
<connectionStrings>

      <clear />
```

```
      <add name="DefaultConnection" providerName="System.Data.SqlClient" connec-
tionString="Server=UJAY-TOSH; Database=CaringHearts;User id
=sa;Password=****************" />
      <add name="CaringHearts" providerName="System.Data.SqlClient" connection-
String="Server=UJAY-TOSH; Database=CaringHearts;User id =sa;Password=stanley" />


  </connectionStrings>
```

Listing 4. The application's configuration file connection strings.

The authentication and authorization tables, together with the Donor and Beneficiary tables were created at the runtime in the Sql-Server management studio. Figure 11 illustrates the outcome of listing 2, listing 3 and listing 4 in the Sql-Server 2012 Management studio.

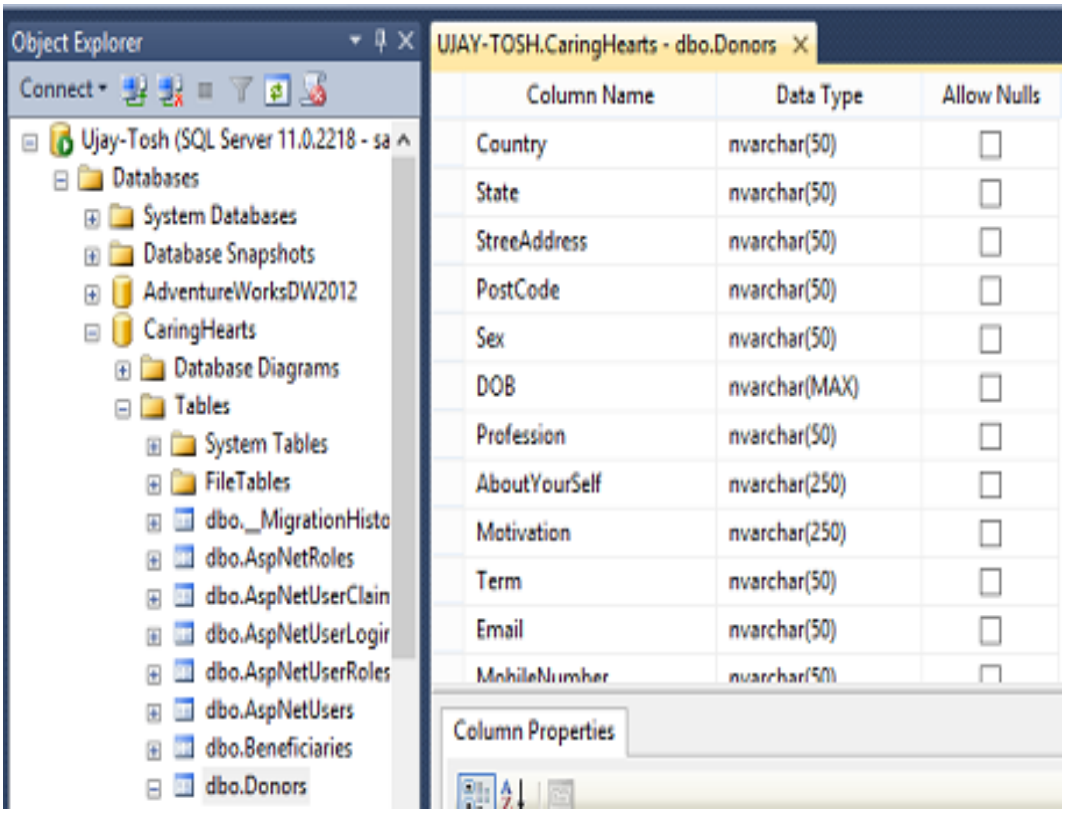

Figure 11. Sql-Server 2012 Management studios showing Caring Hearts Database and its tables.

5.1 Data Access Layer Programming

Besides the KHDBContext, the Beneficiary and the Donor object classes, there are also the BeneficiaryData object class, the DonorData object class and Linq_Statement object class.

The BeneficiaryData Object Class

The BeneficiaryData object class was used to push data gathered from the BeneficiaryLogic to the Sql-Server database. In this object class there is the AddBeneficiary method which takes the BeneficiaryLogic object as a parameter. The KHDBContext object class is initialized in this method as well as the Beneficiary object class. Then each field of the BeneficiaryLogic object class is assigned to the corresponding field of the Beneficiary object class. Then the Add method of the KHDContext.Beneficiaries is called passing the Beneficiary object data class as its parameter and saved into the Sql-Server database by calling the KHDContext's SaveChanges method, as Listing 5 illustrates.

```
public class BeneficiaryData{
          public void AddBeneficiary(BeneficiaryLogic bl) {
            KHDBContext kContext = new KHDBContext();

            Beneficiary d = new Beneficiary();
            String dob = bl.Day + "/" + bl.Month + "/" + bl.Year;
            d.ID = bl.ID;
            d.FirstName = bl.FirstName;
            d.LastName = bl.LastName;
            d.UserName = bl.UserName;
            d.Password = bl.Password;
            d.Category = bl.Category;
            d.Country = bl.Country;
            d.State = bl.State;
            d.StreeAddress = bl.StreeAddress;
            d.PostCode = bl.PostCode;
            d.Sex = bl.Sex;
            d.DOB = dob;

            d.AboutYourSelf = bl.AboutYourSelf;
            d.Challenge = bl.Challenge;



            d.Email = bl.Email;
            d.MobileNumber = bl.MobileNumber;
            d.Pix = bl.Pix;

            kContext.Beneficiaries.Add(d);
            kContext.SaveChanges();
```

```
        }
    }
```

Listing 5. BeneficiaryData object class code snippets.

Listing 5 shows the Beneficiary class object and its fields being assigned the values of the BeneficiaryLogic object class fields.

5.2 The DonorData Object Class

The DonorData class like the BeneficiaryData class is used to push the data input by a user registering on the website as Donor to the Sql-Server database. In the class there is the AddDonor method where KHDContext and the Donor class are initialized. The AddDonor method takes the DonorLogic object class as its parameter and assigns the values of the DonorLogic fields to the corresponding fields of the Donor class object. The KHDBContext's Add method is called passing the Donor object to it as a parameter and saved into the Sql-Server database with the SaveChanges method of the KHDContext. Listing 6 shows the code snippet of the DonorData class object.

```
public class DonorData
    {
        public void AddDonor(DonorLogic dl)
        {
            KHDBContext kContext = new KHDBContext();

            Donor d = new Donor();
            String dob = dl.Day + "/" + dl.Month + "/" + dl.Year;
            d.ID = dl.ID;
            d.FirstName = dl.FirstName;
            d.LastName = dl.LastName;
            d.UserName = dl.UserName;
            d.Password = dl.Password;
            d.Category = dl.Category;
            d.Country = dl.Country;
            d.State = dl.State;
            d.StreeAddress = dl.StreeAddress;
            d.PostCode = dl.PostCode;
            d.Sex = dl.Sex;
            d.DOB = dob;
            d.Profession = dl.Profession;
            d.AboutYourSelf = dl.AboutYourSelf;
            d.Motivation = dl.Movtivation;


            d.Term = dl.Term;
```

```
            d.Email = dl.Email;
            d.MobileNumber = dl.MobileNumber;
            d.Pix = dl.Pix;

            kContext.Donors.Add(d);
            kContext.SaveChanges();


        }
```

Listing 6. Code snippets of the DonorData object class.

Listing 6 illustrates the DonorData object class and its AddDonor method which takes DonorLogic class object as parameter and assigns the values of the DonorLogic fields to its corresponding Donor object class field.

5.3 The Linq_Statement object class

The Linq_Statement object class is where users' data are retrieved from the Sql-server database and stored in a DataSet object. In the Linq_Statement object class, the KHDContext class is initialized while the Beneficiary and the Donor objects are declared. The Linq_Statement object class has GetUserData which returns a DataSet with two tables; the helper DataTable which has exact columns as the CaringHearts database table Donor. And the Beneficiary DataTable which has exact columns as the CaringHearts Database Beneficiary table in the SQL-Server database as shown in appendix 1.

The Linq_Statement object class has another method called SelectDonor. The SelectDonor method takes the user's name as a string parameter and uses it to fetch the Donor data in the Sql-Server database as listing 8 illustrates.

```
public Donor selectDonor(string username)
        {

            var dd = (from dr in context.Donors
                                    select dr).FirstOrDefault();
            Donor da =dd;

            return da;
        }
```

Listing 8. Linq_Statement selectDonor method code snippets

Listing 8 shows the code for the selectDonor method which returns a Donor object. There is a similar method called SelectBeneficiary for fetching the Beneficiary data from the database.

The Linq_Statement object class has two methods the BeneficiaryDetails and the DonorDetails and each returns a DataSet. In each of the these methods the data of Beneficiary and Donor objects are selected from the Beneficiary and Donor tables of the CaringHearts database using LINQ to EntityFramework selected statement. The result of the select statement is returned as an IEnumerable object and converted to an array of Beneficiary or Donor object depending on the method called. Each object of the array is added to the DataSet object created with listing 7 and the DataSet is returned by the methods as listing 9 shows.

```
   public  DataSet BeneficiaryDetails()
        {
            DataSet dset = GetUserData("Beneficiary");
            DataTable table = dset.Tables["Needy"];
            ResizeImage im = new ResizeImage();
            IEnumerable<Beneficiary> dd = from dr in context.Beneficiaries
                                          select dr;
            Beneficiary[] da = dd.ToArray();
            Bitmap[] bitmap = new Bitmap[da.Length];
            for (int k = 0; k < da.Length; k++)
            {
                bitmap[k] = im.PostImage(da[k].Pix);
                table.Rows.Add(da[k].ID, da[k].FirstName, da[k].UserName,
da[k].Category, da[k].Country,
                       da[k].State, da[k].StreeAddress, da[k].PostCode,
da[k].Sex, da[k].DOB, da[k].AboutYourSelf, da[k].Challenge,
                       da[k].Email, da[k].MobileNumber, bitmap[k]);
            }



            return dset;
        }
```

Listing 9. C# and LINQ to Entity Framework code of the BeneficiaryDetails method.

The Business logic layer serves as an interface between the Presentation and the Data Access layer. The Presentation layer does not interact directly with the Data Access layer and vice versa in a three-tier web application architecture. For that reason the Business logic layer is where data received by the Presentation layer and going to the

Data Access layer are sent to. In the Business logic layer of the application there the BeneficiaryLogic, DonorLogic, PasswordHash and ResizeImage object classes.

5.4 The BeneficiaryLogic object class

The BeneficiaryLogic class object has C# properties representing data input by a user registering as Beneficiary on the web application. It also has methods for getting data across to the Data Access layer and for retrieving data from the Data Access layer, as listing 10 shows.

```
public class BeneficiaryLogic
  {

        public Guid ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }
        public string Category { get; set; }
        public string Country { get; set; }
        public string State { get; set; }
        public string StreeAddress { get; set; }
        public string PostCode { get; set; }
        public string Sex { get; set; }
        public string Day { get; set; }
        public string Month { get; set; }
        public string Year { get; set; }

        public string AboutYourSelf { get; set; }
        public string Challenge { get; set; }

        public string Email { get; set; }
        public string MobileNumber { get; set; }
        public Byte[] Pix { get; set; }
        public void AddBeneficiary(BeneficiaryLogic bl)
        {
            BeneficiaryData bd = new BeneficiaryData();
            bd.AddBeneficiary(bl);
```

```
        }
        public Beneficiary[] GetBeneficiaries()
        {
            Beneficiary[] ben =new Beneficiary[34];

            return ben;
        }
        public Beneficiary GetBeneficiary(string username) {
            Linq_Statments ls = new Linq_Statments();
            return ls.selectBeneficiary(username);
        }


    }
```

Listing 10. C# code for The BeneficiaryLogic class object

From listing 10, one can see that objects from the Data Access layer such as BeneficiaryData and Linq_Statement are initialized and their different methods called in the relevant method of the BeneficiaryLogic class object. Similar class object called the DonorLogic exist in the Business logic layer of the application.

PasswordHash object class

PasswordHash object class is used not only to hash a user's passwords, but also to salt the password making password cracking by hackers difficult. Password inputs by users of the website are turned into any amount of fix-length fingerprint data that cannot be reversed before being stored in the database. The fingerprint data are randomized, to ensure that same fingerprint data is not got from the same input data every time. Listing 11 illustrates the C# code used for hashing and salting users' password.

```
public class PasswordHash
    {
        private const int PBKDF2IterCount = 1000; // default for
Rfc2898DeriveBytes
        private const int PBKDF2SubkeyLength = 256 / 8; // 256 bits
        private const int SaltSize = 128 / 8; // 128 bits
        public static string HashPassword(string password)
        {
            byte[] salt;
            byte[] subkey;
            using (var deriveBytes = new Rfc2898DeriveBytes(password, SaltSize,
PBKDF2IterCount))
            {
                salt = deriveBytes.Salt;
                subkey = deriveBytes.GetBytes(PBKDF2SubkeyLength);
            }
```

```
            byte[] outputBytes = new byte[1 + SaltSize + PBKDF2SubkeyLength];
            Buffer.BlockCopy(salt, 0, outputBytes, 1, SaltSize);
            Buffer.BlockCopy(subkey, 0, outputBytes, 1 + SaltSize,
PBKDF2SubkeyLength);
            return Convert.ToBase64String(outputBytes);
        }

        }
```

Listing 11. Password hashing and salting C# code.

ResizeImage Object Class

Pictures uploaded by users are not stored in the original format. The pictures are in the database. Because storing the pictures in their original format in the database will consume much space they are converted to binary format before getting them across to the Data Access layer and finally to the database. When the image is retrieved for the Data Access layer the process will be reversed. Listing 12 shows the ResizeImage object class code and its PrepareImageForUpload and PostImage methods used for Image conversion to and from binary respectively.

```
public class ResizeImage
    {
        public ResizeImage()
        {
            //
            // TODO: Add constructor logic here
            //
        }


        public byte[] PrepImageForUpload(Stream FileData)
        {
            using (Bitmap origImage = new Bitmap(FileData))
            {
                int maxWidth = 165;

                int newWidth = origImage.Width;
                int newHeight = origImage.Height;
                if (origImage.Width < newWidth) //Force to max width
                {
                    newWidth = maxWidth;
                    newHeight = origImage.Height * maxWidth / origImage.Width;
                }

                using (Bitmap newImage = new Bitmap(newWidth, newHeight))
                {
                    using (Graphics gr = Graphics.FromImage(newImage))
                    {
                        gr.SmoothingMode = SmoothingMode.AntiAlias;
                        gr.InterpolationMode = Interpola-
tionMode.HighQualityBicubic;
                        gr.PixelOffsetMode = PixelOffsetMode.HighQuality;
```

```
                        gr.DrawImage(origImage, new Rectangle(0, 0, newWidth,
newHeight));

                        MemoryStream ms = new MemoryStream();
                        newImage.Save(ms, Sys-
tem.Drawing.Imaging.ImageFormat.Jpeg);
                        return ms.ToArray();
                    }
                }
            }

        }
        public Bitmap PostImage(byte[] img)
        {
            MemoryStream ms = new MemoryStream(img);
            System.Drawing.Bitmap imgdraw = (Bit-
map)System.Drawing.Bitmap.FromStream(ms);
            System.Drawing.Bitmap imgdrawing = (Bit-
map)imgdraw.GetThumbnailImage(100, 100, null, new System.IntPtr());
            imgdrawing.Save(System.Web.HttpContext.Current.Response.OutputStream,
System.Drawing.Imaging.ImageFormat.Bmp);
            return imgdrawing;

        }
    }
```

Listing 12. ResizeImage class object code

5.5 The Presentation Layer programming

The presentation layer deals with fetching data keyed in by users. In this phase, a user's interface control unique identifier (ID) attribute is used to identify each control. Since all data are keyed in as a string object, appropriate data type conversion is done in this phase by referencing the value of the ID of the particular control. All this is done in the button event method. The HTML button has onserverclick attribute which automatically will create an event method at the backend side once the onserverclick attribute is add to the button's HTML code. This event method enables programmers to program events that happen when the button is clicked. In the onserverclick event method of the Donor and Beneficiary registration pages, data keyed in on these page controls are converted to appropriate data type and assigned to appropriate BeneficiaryLogic or DonorLogic objects, as appendix 2 illustrate.

Authentication

The login page is used to authenticate users before granting them access to certain web pages of the application. The ASP.NET Identity and Microsoft OWIN are used for authentication in this application. In the login button event, the userManager object find method is used to determine if the user's username and password are stored in the database. If that is the case, the user is authenticated and redirected to the user's profile page as listing 15 shows.

```
  public partial class Login : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                if (User.Identity.IsAuthenticated)
                {
                    DataSet ds = new DataSet();
                    msg.Value = string.Format("Hello {0}!!", Us-
er.Identity.GetUserName());
                    msg.Visible = true;
                }
            }
        }

        protected void btnSubmit_ServerClick(object sender, EventArgs e)
        {

            var userStore = new UserStore<IdentityUser>();
            var userManager = new UserManager<IdentityUser>(userStore);
            var user = userManager.Find(loginID.Value, loginPassword.Value);
            if (user != null)
            {
               // Session["UserDetails"] =
Linq_Statments.DonorDetails(user.UserName);

                var authenticationManager = HttpCon-
text.Current.GetOwinContext().Authentication;
                var userIdentity = userManager.CreateIdentity(user, DefaultAu-
thenticationTypes.ApplicationCookie);

                authenticationManager.SignIn(new AuthenticationProperties() {
IsPersistent = false }, userIdentity);
                Response.Redirect("~/NeedyInfo/needy_info.aspx");
            }
            else
            {
                msg.Value = "Invalid username or password.";
                msg.Visible = true;
            }


        }
    }
```

Listing 15. Application users' authentication code

As can be seen from listing 15, users are redirected to their profile pages when users login. Authorization gives only login users the right to access their profile pages and restrict certain users from accessing certain pages with sensitive data. This part of the application programming is yet to be implemented when this thesis is being written due to time constrains.

6 Results and Discussion

:The application called Caring Hearts is designed in order to participate in the Microsoft Imagine cup 2014. The user interface of the application was designed with standard HTML 5 using the mobile first responsive design

Mobile first responsive design

Different electronic devices nowadays are used to access web application. Most of these electronic devices are of different types, shapes and screen sizes. For this purpose, it is important to design a web application with user interface layout that responds to any screen size. The considerations to take into account when designing a web application layout is not limited to screen sizes of mobile devices but also the way these mobile devices connect to the Internet and interact with web content. For instance recent mobile devices have touch screens which is a new way to interact with web contents. [1]

To develop a website in a mobile context and not only for different screen sizes, many challenges of mobile development have to be resolved upfront. The need of mobile context makes what content is important and how to present that content as fast as possible vital to Caring Hearts web application and web applications in general. [1]

Caring Hearts Application Features

More features will be added to the application as the need arises but for now the application has the following features.

User Registration Pages

The user registration pages are of two types. The pages are different as the professionals are required to give a couple of different piece of information for instance the professional's motivation, profession and how he/she intends to help. In case of the beneficiary, he/she is required to give information on what challenge he/she is undergoing during registration.

User Profiles

Each user will have a profile page where the user profile pictures and information are displayed. The user's profile page is access when that user has been authenticated. Every professional is authorized to view the profile of a beneficiary but a needy can only be authorized to view a professional profile if that professional has agreed to render any sort of help to that particular beneficiary.

A messaging feature will be included in the profile page to make users interact with one another when they have been linked. which means that users would not be allowed to communicate with one another unless they are linked. A professional when authenticated can browse through the needy list; go to the beneficiary's profile page to know more about him/her and if the professional thinks he/she can render any help to the needy the professional contacts the application administrator via the messaging feature. The professional and the needy will be linked after the authenticities of both of them are proved beyond reasonable doubt.

Monitoring Feature

The progress of linked users will be monitored by the site administrators if users are comfortable with it to ensure everything is going on smoothly. There are also features to alert the site administrators and block the beneficiary if the professional suspects anything fraudulent and vice versa.

Programming and developing an ASP.NET application of this project magnitude takes longer time than available and is progressive. At the time of writing this thesis, the working part of the application includes

Registration Page For Users

The beneficiary partial class object and the helper partial class object for registering users as a beneficiary and helper has been fully developed as planned.

Authentication For Users

Registered users can be authenticated and given access to certain content of the web-site depending on whether a user registered as a Helper/Donor or a Beneficiary.
At the moment, all the pages needed to make the application fully functional are not yet designed and programmed due to the assumption that the application development and the Thesis writing duration will last till the end of May 2014.

Discussion

The two important users of the application are the Beneficiary and the Donor. These users can be registered, log on to the website and access certain resources of the website..The users' passwords are well secured using the hashing and salting technologies. If the website is hacked, users' sensitive data such as passwords will not be compromised.

HTML as the front end of the website will make it fit in all mobile devices. However, as an ASP.NET programmer developing an ASP.NET application with the HTML front end for the first time, it has been quite a challenge and has elongated the duration of the project development. For instance presenting data from database to users can be easily done by using the grid view controls and assigning the query results of the Data Access Layer's Linq_Statement to the DataSource property of the GridView object. However with standard HTML front end controls, tables are used to present data to users, and how to populate tables with the LINQ query result has to be figured out before proceeding with the application programming. Challenges like this have delayed the project development processes.

The email service is one of the yet to programmed features of the project. Though this aspect of programming in ASP.NET is not completely new to me, it is the first time I

am going to develop any application with this kind of feature which means more time will be needed for testing the email services.

The application data will be stored in the cloud using the Microsoft Azure cloud Mobile Backend as-a-service. Cloud computing is completely new to me, I have to do some self-studying before programming this feature of the project, which will ensure efficient data presentation to users with mobile devices.

Though the set goals for the application have not all been achieved, we are progressing and we will add the yet to be added features as soon as possible.

7 Conclusion

The goal of the project was to develop an ASP.NET web application in order to participate in the Microsoft Imagine Cup 2014. The application will have two categories of users the professionals called Donors/helper and the Beneficiaries or those in need of professional help but cannot afford such help.

At the time of writing this thesis, both professional and beneficiary users can register and log on securely to the website. Most of the object classes needed for storing data to and retrieving data from the Sql-Server 2012 database have been programmed using the 3-tier web development architecture.

In the Presentation Layer, data input by users is converted to appropriate data type and assigned to appropriate objects in the Business Logic Layer and data retrieved from the database pass the Data Access Layer and the Business Logic Layer before they are presented to users in the Presentation Layer. The database querying was done with LINQ to Entity Framework in the Data Access Layer and the Business Logic Layer is the interface for the Presentation Layer and Data Access Layer communication.

Though there are features of the application that are not yet included, the team will continue to work on the project till all the features we decided on the onset to add to the project are added. However the fundamental features needed to implement other features of the application have been implemented.

All in all, I would say the project has been a success and I have learned a great deal as the backend programmer of this ASP.NET web application which has a standard HTML front end instead of the usual ASP.NET front end. When the application is fully developed and deployed, it will help bring professional who are willing to use their professional skills to help out individuals who need their professionals help but cannot afford with distance not being a barrier.

References

1. Brad Frost. Creating a mobile-First Responsive Web Design [online]. April 16 ,2012.
   URL: http://www.html5rocks.com/en/mobile/responsivedesign/. Accessed April 27, 2014

2. Sudhirmangh. Beginners Introduction to ASP.NET [online].
   Oct 20, 2004.
   URL:http://www.codeproject.com/Articles/4468/Beginners-Introduction-to-%20%20ASP-NET.
   Accessed February 23, 2014.

3. Darie,Wyatt and Possey. Build your own ASP.NET 4 website using C#&VB [online].United state; Site point; September 2011.
   URL: http://edu.ercess.co.in/ebooks/.net/Build-Your-Own-ASP.NET-4-Website-using-C%23-VB.NET.pdf.
   Accessed February 23, 2014.

4. Bochicchio D, Mostarda S, De Sanctis M. ASP.NET 4 in Practice. New York: Manning Publication Co; 2011.

5. Thomas Erl, Zaigham Mahmood and Richardo Puttini.Cloud Computing Concepts,Technology and Architecture. Westford Massachusetts:Person Hall Person Education; May 2013.

6. Manit Kalsi. Demystifying the Mobile Backend as-a-Service [online]. March 21,2014.
   URL: http://www.opensourceforu.com/2014/03/demystifying-mobile-backend-service/ . Accessed April 5,2014.

7. Pranav Rastogi Introduction to ASP.NET Identity [online]. October 17,2013.

   URL:http://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity. Accessed April 15,2014.

GetUserData method code snippets of the Linq_Statement object class.

```
public  DataSet GetUserData(string user_type)
        {
            DataSet ds = new DataSet(user_type);

            DataTable helper = ds.Tables.Add("Donor");
            DataColumn DonorId = helper.Columns.Add("id", typeof(Guid));
            DataColumn DonorFName = helper.Columns.Add("FirstName",
typeof(string));
            DataColumn DonorLName = helper.Columns.Add("LastName",
typeof(string));
            DataColumn DonorUName = helper.Columns.Add("UserName",
typeof(string));
            DataColumn DonorCategory= helper.Columns.Add("Category",
typeof(string));
            DataColumn DonorCountry= helper.Columns.Add("Country",
typeof(string));
            DataColumn DonorState = helper.Columns.Add("State", typeof(string));
            DataColumn DonorAddress = helper.Columns.Add("StreetAddress",
typeof(string));
            DataColumn DonorPostCode = helper.Columns.Add("PostCode",
typeof(string));
            DataColumn DonorSex = helper.Columns.Add("Sex", typeof(string));
            DataColumn DonorDOB = helper.Columns.Add("DOB",
typeof(DataSetDateTime));
            DataColumn Profession = helper.Columns.Add("Profession",
typeof(string));
            DataColumn AboutYourself = helper.Columns.Add("AboutYourSelf",
typeof(string));
            DataColumn Movation= helper.Columns.Add("Motivation",
typeof(string));
            DataColumn DonorTerm = helper.Columns.Add("Term", typeof(string));
            DataColumn DonorEmail = helper.Columns.Add("Email", typeof(string));
            DataColumn DonorMobileNumber = helper.Columns.Add("MobileNumber",
typeof(string));
            DataColumn DonorPhoto = helper.Columns.Add("Photo", typeof(Bitmap));
            helper.PrimaryKey = new DataColumn[] { helper.Columns["id"] };

            DataTable needy = ds.Tables.Add("Needy");
            DataColumn NeedyId = needy.Columns.Add("id", typeof(Guid));
            DataColumn NeedyFName = needy.Columns.Add("FirstName",
typeof(string));
            DataColumn NeedyLName = needy.Columns.Add("LastName",
typeof(string));
            DataColumn NeedyUName = needy.Columns.Add("UserName",
typeof(string));
            DataColumn NeedyCategory = needy.Columns.Add("Category",
typeof(string));
            DataColumn NeedyCountry = needy.Columns.Add("Country",
typeof(string));
            DataColumn NeedyState = needy.Columns.Add("State", typeof(string));
            DataColumn NeedyAddress = needy.Columns.Add("StreetAddress",
typeof(string));
            DataColumn NeedyPostCode = needy.Columns.Add("PostCode",
typeof(string));
```

```
            DataColumn NeedySex = needy.Columns.Add("Sex", typeof(string));
            DataColumn NeedyAboutYourself = needy.Columns.Add("AboutYourSelf",
typeof(string));
            DataColumn NeedyDOB = needy.Columns.Add("DOB",
typeof(DataSetDateTime));
            DataColumn Challenge = needy.Columns.Add("Profession",
typeof(string));
            DataColumn Email = needy.Columns.Add("Email", typeof(string));



            DataColumn NeedyMobileNumber = needy.Columns.Add("MobileNumber",
typeof(string));
            DataColumn NeedyPhoto = needy.Columns.Add("Photo", typeof(Bitmap));
            needy.PrimaryKey = new DataColumn[] { needy.Columns["id"] };


            return ds;

        }
    }
```

Helper object class's code

```
public partial class helper : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        private string SexCheckBox()
        {
            string sex;
            if (Field90.Checked)
            {
                Field91.Checked = false;
                sex = "Male";
            }
            else if (Field91.Checked)
            {
                Field90.Checked = false;
                sex = "Female";
            }
            else
            {
                sex = "None";
            }
            return sex;
        }

        protected void btnSubmit_ServerClick1(object sender, EventArgs e)
        {
            DonorLogic dl = new DonorLogic();

            Donor dd;
            Guid id = Guid.NewGuid();
            int index;
            dl.ID = id;
            dl.FirstName = Field81.Value;
            dl.LastName = Field82.Value;
            dl.UserName = Field83.Value;
            dl.Password = PasswordHash.HashPassword(Field84.Value);

            index = ddlCategory.SelectedIndex;
            dl.Category = ddlCategory.Items[index].Text;

            index = ddlCountry.SelectedIndex;
            dl.Country = ddlCountry.Items[index].Text;
            dl.State = Field85.Value;
            dl.StreeAddress = Field87.Value;
            dl.PostCode = Field88.Value;
            index = ddlDay.SelectedIndex;
            dl.Day = ddlDay.Items[index].Text;
            index = ddlMonth.SelectedIndex;
            dl.Month = ddlMonth.Items[index].Text;
            index = ddlYear.SelectedIndex;
            dl.Year = ddlYear.Items[index].Text;
            dl.Sex = SexCheckBox();
            dl.AboutYourSelf = Field93.Value;
            dl.Movtivation = Field94.Value;
```

```
            dl.Email = Field95.Value;
            dl.MobileNumber = Field96.Value;
            dl.Profession = Field98.Value;
            dl.Term = ddlTerm.Value;

            if (Field9.PostedFile != null)
            {
                Stream st = Field9.PostedFile.InputStream;

                ResizeImage ri = new ResizeImage();
                dl.Pix = ri.PrepImageForUpload(st);


            }
            if ((dd = dl.GetDonor(Field83.Value)) == null)
            {
                try
                {

                    dl.AddDonor(dl);
                    msg.Visible = true;
                    msg.Value = dl.createProfile(Field83.Value, Field84.Value);

                    Response.Redirect("helper.aspx");


                }
                catch (Exception ex)
                {
                    msg.Value = ex.Message;
                }


            }
            else
                msg.Value = "Sorry username already exist !";


        }

    }
```

Needy object class code

```
public partial class needy : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        private string SexCheckBox()
        {
            string sex;
            if (Field90.Checked)
```

```
            {
                Field91.Checked = false;
                sex = "Male";
            }
            else if (Field91.Checked)
            {
                Field90.Checked = false;
                sex = "Female";
            }
            else
            {
                sex = "None";
            }
            return sex;
        }

        protected void btnSubmit_ServerClick1(object sender, EventArgs e)
        {
            BeneficiaryLogic bl = new BeneficiaryLogic();
            DonorLogic dl = new DonorLogic();

            Beneficiary bf;
            Guid id = Guid.NewGuid();
            int index;
            bl.ID = id;
            bl.FirstName = Field81.Value;
            bl.LastName = Field82.Value;
            bl.UserName = Field83.Value;
            bl.Password = PasswordHash.HashPassword(Field84.Value);

            index = ddlCategory.SelectedIndex;
            bl.Category = ddlCategory.Items[index].Text;

            index = ddlCountry.SelectedIndex;
            bl.Country = ddlCountry.Items[index].Text;
            bl.State = Field85.Value;
            bl.StreeAddress = Field87.Value;
            bl.PostCode = Field88.Value;
            index = ddlDay.SelectedIndex;
            bl.Day = ddlDay.Items[index].Text;
            index = ddlMonth.SelectedIndex;
            bl.Month = ddlMonth.Items[index].Text;
            index = ddlYear.SelectedIndex;
            bl.Year = ddlYear.Items[index].Text;
            bl.Sex = SexCheckBox();
            bl.AboutYourSelf = Field93.Value;
            bl.Challenge = Field94.Value;
            bl.Email = Field95.Value;
            bl.MobileNumber = Field96.Value;


            if (Field9.PostedFile != null)
            {
                Stream st = Field9.PostedFile.InputStream;

                ResizeImage ri = new ResizeImage();
                bl.Pix = ri.PrepImageForUpload(st);


            }
```

```
            if ((bf = bl.GetBeneficiary(Field83.Value)) == null)
            {
                try
                {

                    bl.AddBeneficiary(bl);
                    msg.Visible = true;
                    msg.Value = dl.createProfile(bl.UserName, Field84.Value);

                    Response.Redirect("~/Default.aspx");



                }
                catch (Exception ex)
                {
                    msg.Value = ex.Message;
                }

            }
            else
                msg.Value = "Sorry username already exist!";


        }

    }
```