

Technical Specification

Softball Signups

|  |  |
| --- | --- |
| **Filename:** | ITLC\_Technical\_Spec\_Template\_V2.0 |
| **Last Save Date:** | Wednesday, March 05, 2014 |
| **Author(s):** | First Last (alias) |
| **File Location:** | [\\server\folder](mailto:ITLCHelp@microsoft.com) |
| **Project Information Classification:** | HBI, MBI, LBI, PII, or HSPII [select information classification] |
| **Microsoft Confidential:** This document must be handled in accordance with its assigned information classification. Refer to *InfoSec #2.0 Information Classification & Handling Standard* for more information. | |

Table of Contents

[1 Introduction 3](#_Toc381780057)

[2 System Architecture 3](#_Toc381780058)

[2.1 Distributed System View 3](#_Toc381780059)

[2.1.1 Application Context Diagram 3](#_Toc381780060)

[2.1.2 Functional Components 3](#_Toc381780061)

[2.2 Deployment View 4](#_Toc381780062)

[2.2.1 Logical Server Definition 4](#_Toc381780063)

[2.3 System Quality Attributes 4](#_Toc381780064)

[2.3.1 Performance 4](#_Toc381780065)

[2.3.2 Dependability 4](#_Toc381780066)

[2.3.3 Security 5](#_Toc381780067)

[2.3.4 <Use Case Name Goes Here> 5](#_Toc381780068)

[2.4 Technology Dependencies 5](#_Toc381780069)

[2.4.1 Server software dependencies 5](#_Toc381780070)

[3 Front End 6](#_Toc381780071)

[3.1 Windows Application Front End 6](#_Toc381780072)

[3.1.1 Screen Navigation map 6](#_Toc381780073)

[3.1.2 Localization 10](#_Toc381780074)

[3.1.3 Configurations 10](#_Toc381780075)

[3.1.4 Error Handling 10](#_Toc381780076)

[3.1.5 Security 10](#_Toc381780077)

[4 Database Objects 10](#_Toc381780078)

[4.1 Entity Relationship Diagram 11](#_Toc381780079)

[4.1.1 Tables used in Application 11](#_Toc381780080)

[4.1.2 Meta Data Table Structure 11](#_Toc381780081)

[4.2 Tables 12](#_Toc381780082)

[4.3 Views 13](#_Toc381780083)

[4.4 Indexes 13](#_Toc381780084)

[4.5 Triggers 13](#_Toc381780085)

[4.6 User Defined Functions 13](#_Toc381780086)

[4.7 Stored Procedures 13](#_Toc381780087)

[4.8 Scripts 13](#_Toc381780088)

[5 Registry Settings/INI files/.config files 13](#_Toc381780089)

[6 Internationalization 14](#_Toc381780090)

[Appendix A Glossary/ Definitions 15](#_Toc381780091)

[Appendix B Related Documents/References 16](#_Toc381780092)

[Appendix C Environments 17](#_Toc381780093)

[Appendix D Reviewers 18](#_Toc381780094)

# Introduction

This is the development design specification for the MSSA Baseball roster application for the MSSA class project. It is designed for the class to become familiar with the development cycle within many of the Microsoft teams. The document will outline many of the technical details that will make up the project.

This project is a front end solution that will allow users to enter themselves into a baseball roster. When they enter themselves into the roster, the information will be persisted into an Azure database.

This project is not designed to be a shipping product. There are bugs within not only the application, but also the specification here in the hopes that the students will search out these issues and report them in the manner setup within the class.

# System Architecture

## Distributed System View

### Application Context Diagram

This application is a simple roster management tool. It uses a front in written in Win32/.Net4.5 which interfaces with a back in database. It will allow the user to view/add/edit characteristics associated with users and teams. The current implementation uses an Azure SQL database as the backend repository of information so that a user can manage the information from any machine that has internet connection.



### Functional Components

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Responsibility** | **Execution Environment** |
| .NET UI Front End |  |  |
| Azure Database |  |  |
|  |  |  |

* + - 1. Functional Component Description

.NET UI Front End – this component will have several views for each for each of the management characteristics of the tool. It will include:

1. View/Edit Players
2. Add Players
3. View/Edit Teams
4. Add Team

The UI will connect to an Azure SQL database. There are views and tables. All joins/queries will be done using LINQ to SQL data object. There will be a Visual Studio SQL Database Project which will manage the schema between the UI front end and the Azure SQL database.

## Deployment View

There are several connections within the application that one needs to be aware of that will be points of interest. These can be issues with connection problems, security issue and configuration of the tool.



### Logical Server Definition

There is only one database connection which the tool uses as defined below in the table.

|  |  |  |
| --- | --- | --- |
| **Server Logical Name** | **Software environment** | **Security Environment** |
| Data Source=e0emvc1t9z.database.windows.net;  Initial Catalog=dbBaseBallRoster | As defined by Microsoft for the Azure connection point. | Microsoft to supply the security environment per Azure standards. |

## System Quality Attributes

### Performance

The performance of the application is defined by the performance of the internet connection between the UI and the Azure SQL database, and the Azure SQL database itself. Both of these are outside of the control of the tool. Given the performance of the Azure components, it is an acceptable criteria.

### Dependability

The dependability of the tool is defined by the dependability of the Azure components. At this time this is an acceptable criteria for the tool.

* + - 1. Availability

The uptime or availability is defined by the Azure components. At this time it is within acceptable limits for the usage of the tool.

* + - 1. Reliability

The reliability is defined by the Azure components. At this time it is within acceptable limits for the usage of the tool

* + - 1. Safety

Azure components are backed up regularly. This does not mean that the actual database is backed up in the case of lost data. Due to the level of importance of the data we feel that this is an OK tradeoff.

* + - 1. Maintainability

The front end and the data are separate components. There is also a database project which will keep the database schema and the front end tools in sync. Since the tool uses LINQ to SQL, any changes/edits/enhancements will find any out of sync issues and not compile quickly identifying any issues. This will allow easy maintainability.

### Security

* + - 1. Confidentiality

There is one user characteristics (email address) that defines private user information. This will only be shown when an administrator is logged in. Normal users will not see the email addresses of other users. All other data will be exposed to all other users.

* + - 1. Integrity

The integrity of the data in the Azure SQL database is controlled by the use of foreign keys. This will keep orphan data from the database.

In this diagram, the ‘sunny day flow’ is illustrated. The user provides sufficient and correct information in order to retrieve a list of agreements, and the list is correctly returned to the front end.

### <Use Case Name Goes Here>

<<place text that gives a brief overview of the use case.>>

<<place the dynamic model diagram here. Use the EMF format to keep the diagram size small.>>

<<place text here that describes which choices from the use case are illustrated in the model.>>

## Technology Dependencies

### Server software dependencies

Since the application is hosted in Azure, SQL Server & IIS are not required on the server. No server is required.

# Front End

## Windows Application Front End

### Screen Navigation map

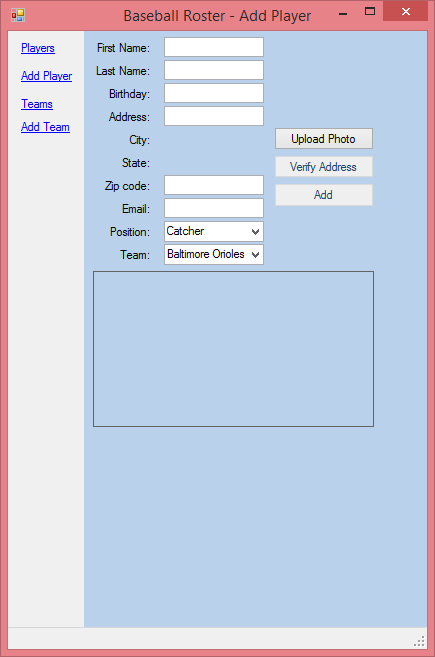
User presses Players: This will list all players.

* When the user presses the individual player in the grid, the player’s details will be displayed in the lower pane.
* When the user presses Upload Photo, a dialog will be displayed that will allow the user to select a JPG image of the player. When the user selects the image, the user’s image will be updated on the form.
* When the user enters the Address and Zip code, the Verify Address button is enabled.
* Pressing the Verify Address will use a web service to identify the City and State and fill in the form with this information.
* When the address is verified and found, a web service is called that will return an aerial image of the players address.
* When the user has added all required fields, the Apply button is enabled.



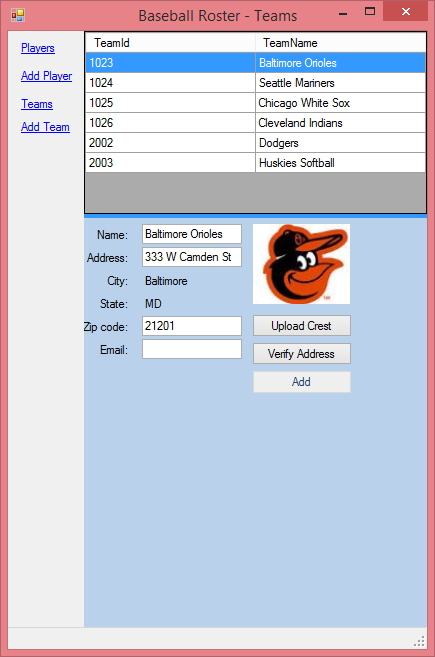
User presses Add Player – this allows the user to add a new player to the database.

* When the user presses Add, the information is saved to the database.
* When the user enters the Address and Zip code, the Verify Address button is enabled.
* Pressing the Verify Address will use a web service to identify the City and State and fill in the form with this information.
* When the address is verified and found, a web service is called that will return an aerial image of the players address.
* Pressing Upload Photo will show a dialog to the user that will allow them to select a JPG image associated with the player.
* When the user has added all required fields, the Add button is enabled.



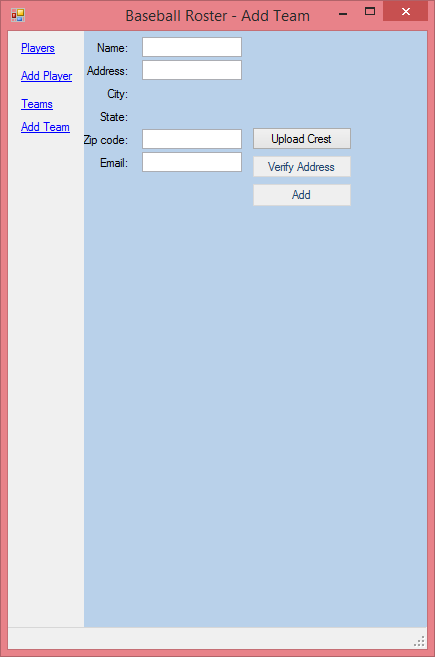
The user presses Teams – this will allow a user to select a specific team which will show the details of the team and also allow the user to edit the team.

* When the user selects a team in the grid, the team information is updated in the lower pane.
* When the user enters the Address and Zip code, the Verify Address button is enabled.
* Pressing the Verify Address will use a web service to identify the City and State and fill in the form with this information.
* When the address is verified and found, a web service is called that will return an aerial image of the players address.
* When the user presses Upload Crest, a dialog will be displayed which the user can then select a JPG image that is associated with the team.
* When a team image is selected, the team image is updated.
* When the user presses Add, the data is stored in the database.
* When the user has added all required fields, the Add button is enabled.



User presses Add Team - this allows the user to add a new team to the database.

* When the user enters the Address and Zip code, the Verify Address button is enabled.
* Pressing the Verify Address will use a web service to identify the City and State and fill in the form with this information.
* When the user presses Upload Crest, a dialog will be displayed which the user can then select a JPG image that is associated with the team.
* When a team image is selected, the team image is updated.
* When the user has added all required fields, the Add button is enabled.
* When the user presses Add, the data is entered into the database.



* + - 1. Screen or Control definition

Name of controls will following a naming convention as follows:

1. [ViewName]{ControlType][Action]
2. Sample controls:

playerButtonUpdatePlayerApply;

playerComboBoxPositions;

playerLabelAddress;

### Localization

All strings that are shown on the form will be localized using the resource file.

### Configurations

The following information will be stored in the configuration file:

* Database connection string.
* ISearchService Web API settings
* IGeoCodeService Web API settings.
* Different control settings for the form so that when the user opens the form again, they will get the same look and feel they had when they last closed the form.

### Error Handling

For most methods, there is a common error handler which will:

* Give the user a similar UI if there are any errors.
* Be easily updated in one place for all methods.
* All UI event methods will wrap the event with an exception handler so that the application will never crash.

### Security

* + - 1. Authentication

Only the database administrator can log into the database and see the individual schema and data. The authentication process will be controlled by the normal database mechanisms.

* + - 1. Authorization

Again this is controlled by the Azure SQL server.

* + - 1. Encryption

There is no encryption used in this product.

# Database Objects

This is a very simple database composed of three tables. All ‘querying’ of the data is done from within the code through LINQ to SQL objects.

## Entity Relationship Diagram

### Tables used in Application

[A data schema (ER diagram) may be included here or may be referenced as a separate document. The sample below is a subset of the RIO ERwin diagram. On maintenance projects where the changes to the database are minimal, this diagram is not required. This ER diagram would contain the list of tables and relationships pertaining to the application]

<<Begin text here>>



### Meta Data Table Structure

[This section requires the ER diagram for the tables that are associated with Meta Data where applicable. Metadata tables are tables that drive business rules for the use of the application without ever being displayed in any report or user interface component. Examples include tables with permissions for screens, controls or page objects by user group]



## Tables

1. TABLE [dbo].[Player]

[PlayerId] INT IDENTITY (1, 1) NOT NULL,

[FirstName] NVARCHAR (50) NOT NULL,

[LastName] NVARCHAR (50) NOT NULL,

[Email] NVARCHAR (50) NOT NULL,

[CreatedDateTime] DATETIME2 (7) NOT NULL,

[ModifiedDateTime] DATETIME2 (7) NOT NULL,

[PositionId] INT NULL,

[TeamId] INT NULL,

[Birthdate] DATETIME2 (7) NULL,

[Image] VARBINARY (MAX) NULL,

[City] VARCHAR (50) NULL,

[State] VARCHAR (2) NULL,

[Zip] VARCHAR (7) NULL,

[Address] VARCHAR (50) NULL,

[AddressURI] VARCHAR (256) NULL,

[Longitude] FLOAT (53) NULL,

[Latitude] FLOAT (53) NULL,

CONSTRAINT [PK\_PlayerId] PRIMARY KEY CLUSTERED ([PlayerId] ASC),

CONSTRAINT [FK\_Player\_Position] FOREIGN KEY ([PositionId]) REFERENCES [dbo].[Position] ([PositionId]),

CONSTRAINT [FK\_Player\_Team] FOREIGN KEY ([TeamId]) REFERENCES [dbo].[Team] ([TeamId])

NONCLUSTERED INDEX [IX\_Player\_\_ID]

ON [dbo].[Player]([PlayerId] ASC

1. TABLE [dbo].[Position]

[PositionId] INT NOT NULL,

[PositionName] NVARCHAR (20) NOT NULL,

PRIMARY KEY CLUSTERED ([PositionId] ASC)

NONCLUSTERED INDEX [IX\_Position\_PositionId]

ON [dbo].[Position]([PositionId] ASC

1. TABLE [dbo].[Team]

[TeamId] INT IDENTITY (1, 1) NOT NULL,

[TeamName] NVARCHAR (50) NOT NULL,

[Address] VARCHAR (50) NULL,

[City] VARCHAR (50) NULL,

[ZipCode] VARCHAR (10) NULL,

[Email] VARCHAR (50) NULL,

[Image] VARBINARY (MAX) NULL,

[State] VARCHAR (2) NULL,

CONSTRAINT [PK\_TeamId] PRIMARY KEY CLUSTERED ([TeamId] ASC)

## Views

There are no views. All ‘views’ are done with LINQ within the code.

## Indexes

NONCLUSTERED INDEX [IX\_Player\_\_ID] ON [dbo].[Player]([PlayerId] ASC

CONSTRAINT [PK\_TeamId] PRIMARY KEY CLUSTERED ([TeamId] ASC)

CONSTRAINT [PK\_PlayerId] PRIMARY KEY CLUSTERED ([PlayerId] ASC),

CONSTRAINT [FK\_Player\_Position] FOREIGN KEY ([PositionId]) REFERENCES [dbo].[Position] ([PositionId]),

CONSTRAINT [FK\_Player\_Team] FOREIGN KEY ([TeamId]) REFERENCES [dbo].[Team] ([TeamId])

NONCLUSTERED INDEX [IX\_Player\_\_ID] ON [dbo].[Player]([PlayerId] ASC

## Triggers

There are no triggers.

## User Defined Functions

There are no user defined functions.

## Stored Procedures

There are no stored procedures.

## Scripts

There is a solution (MSAADatabase) to manage changes to the database. You can manage changes to objects and settings on a database server with the project. There is a wizard within Visual Studio projecv that will help deploy the database.

# Registry Settings/INI files/.config files

| AppSettings Key | Value | Purpose |
| --- | --- | --- |
| dbBaseBallRosterConnectionString |  | This is the connection string to connect to the Azure database. |
| SplitterDistanceTeam | 260 | This is a persisted sizing so the formn retains the same look on startup from what it was the last time it was run. |
| SplitterDistanceRoster | 260 | This is a persisted sizing so the formn retains the same look on startup from what it was the last time it was run. |
| SplitterDistanceContainerSurface | 93 | This is a persisted sizing so the formn retains the same look on startup from what it was the last time it was run. |
| SplitterDistancePlayer | 260 | This is a persisted sizing so the formn retains the same look on startup from what it was the last time it was run. |
| SplitterDistanceSurface | 260 | This is a persisted sizing so the formn retains the same look on startup from what it was the last time it was run. |
| BingKey |  | Key used to call into the GoeCache web APIs. |

# Internationalization

*There is no internationalization done with this project other than to allow localizing the strings.*

*For more information about globalization and localization standards and practices within MSIT,*

* *visit* [*http://sharepoint/sites/globalizeIT*](http://sharepoint/sites/globalizeIT)
* *or contact* [*MSIT Globalization-Localization Info*](http://itlc/) *(mailto:locinfo)]*

1. Glossary/ Definitions

Below is a list of common terms and their definitions that are used throughout this document:

| Term | Definition |
| --- | --- |
|  |  |
|  |  |
|  |  |

1. Related Documents/References

| Document | Document Location |
| --- | --- |
| Vision and Scope |  |
| Business Requirements Document |  |
| Functional Specification |  |
|  |  |
|  |  |

1. Environments

[Include in this section known environmental information. At design time test and production environment details may not be known.]

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Development | Test | Production |
| e.g. IIS |  |  |  |
| e.g. Database |  |  |  |
| e.g. Reporting DB |  |  |  |
| e.g. Batch job group |  |  |  |
| e.g. File share |  |  |  |
| e.g. VSS Location |  |  |  |

1. Reviewers

Below is a list of the project team members and required reviewers

|  |  |  |  |
| --- | --- | --- | --- |
| Person | Role | Contact | Reviewed Date |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |