Iteration 3 Review

Contents

[Implemented User Stories 2](#_Toc352770423)

[Story ID: 4 - Table Exports 2](#_Toc352770424)

[Story ID: 8 - Mobile Forms 2](#_Toc352770425)

[Story ID: 7 – Public URLs 2](#_Toc352770426)

[Story ID: 12 - Configure LAN Server 2](#_Toc352770427)

[Related Progress 3](#_Toc352770428)

[Class Diagram 3](#_Toc352770429)

[Database Schema 5](#_Toc352770430)

[Entity Framework 5](#_Toc352770431)

[SMO Database 6](#_Toc352770432)

[Revised Release Plan 7](#_Toc352770433)

[Difficulties and Learning Outcomes 8](#_Toc352770434)

[Story ID: 4 - Table Exports 8](#_Toc352770435)

[Story ID: 8 - Mobile Forms 8](#_Toc352770436)

[Validation 8](#_Toc352770437)

[Story ID: 7 – Public URLs 8](#_Toc352770438)

[State diagram 9](#_Toc352770439)

[Interaction diagram 10](#_Toc352770440)

# Implemented User Stories

### Story ID: 4 - Table Exports

User can export tables in .CSV format to a location of their choosing. Values are separated by commas and return lines for each row.

A table must first be submitted to the SQL database before it can be exported to a .CSV file. The option to export the table is not visible until the table has been submitted. Tables with no data in them are exported as a row of column headers, we may limit exports to tables that have data entries in a future iteration.

### Story ID: 8 - Mobile Forms

A user view has been created for data input into each submitted form. This view is only currently formatted for desktop browsing. The option to enter this ‘user view’ is only visible after the table has been submitted as a form to the SQL database.

Each user view is dynamically generated based on the columns present in the specified SQL Form table. There are a limited number of default layouts dependent on the input field type (as selected by the user when setting up form) in question. For example radio buttons are presented in the user view when a Boolean value (bit value) is stored in the database.

All user input fields are validated based on the expected type for the database, e.g. Numeric values only for Number fields.

### Story ID: 7 – Public URLs

The user can provide their clients with a public URL to access and fill in their forms. Clients do not need to be account holders to access this page. We see this being useful for businesses that would like to implement these forms across their social networking sites and also to advertise at events for users to access on their own devices. Currently the user must copy the URL from the provided textbox to paste where they like, however, we hope to integrate the site with Facebook/Twitter/Google in the future and provide direct sharing for the user.

### Story ID: 12 - Configure LAN Server

To allow remote access to our local server from another computer on the network we made changes to the file at IISExpress/config/applicationhost.config. These changes specified to allow remote connections via port 80 to be bound to the local IP address and connect to the server running locally. This has been tested and fully functional within the college network.

# Related Progress

# Class Diagram



# Database Schema

### Entity Framework



### SMO Database



# Macintosh HD:Users:shanemurphy:Desktop:Screen Shot 2013-03-14 at 12.41.11.pngRevised Release Plan

# Difficulties and Learning Outcomes

## Story ID: 4 - Table Exports

Some time spent seeking a code solution to separating .CSV file values into separate columns in MS Excel. Ultimately turned out to be a problem with settings in the Excel application itself.

## Story ID: 8 - Mobile Forms

Our main focus when dealing with mobile forms was dealing with user input into the system, and protecting our application from malicious or bad input. The Razor engine implemented in MVC4 automatically protects against cross site scripting (XSS) by rendering all input collected using the razor ‘@’ tag as plain text.

Cross site request forgery attacks (CSRF) can be protected against using the Html.AntiForgeryToke() method. This method issues a random token to the user to protect against this; however, it requires the use of cookies. We have not implemented this as yet, but may do so in a future iteration.

### Validation

We are using The Data Annotations Extensions package available from the NuGet installer to validate certain user inputs, e.g. the Email field required for registration. Other user input is validated using JQuery Validation. A number of different examples and methods are recommended online, leading to much time spent experimenting and trying out a number of different approaches.

## Story ID: 7 – Public URLs

In order to generate a custom URL such as localhost/magic/5, we needed to create a separate method to accept a username and retrieve the corresponding userId before passing to our original method. The problem with this is that as the user is not logged in, the SimpleMembership database has not been initialized at this point and we could not gain access to the database of users to find the UserId. We used the InitialiseSimpleMembershipAttribute(), which is a type of singleton design pattern to ensure only one instance of this connection is made.

We were unable to create a route such that the user could access the input for the tables through an URL similar to localhost/magic/5, the routing table seems to require some (any) values separated by a ‘/’ to replace as the specified default controller and action.

We settled to leave routing to localhost/public/form/username}/{tableId} for this iteration, allowing the user to access forms through localhost/public/form/shane/5 for example. We may return to this at a later iteration, but are content with results for now.

# State diagram



# Interaction diagram

