Iteration 1 Review

Contents

[Application Architecture Definition 2](#_Toc349847535)

[MVC 2](#_Toc349847536)

[Revised User Stories 2](#_Toc349847537)

[Log In 2](#_Toc349847538)

[Generate Form 3](#_Toc349847539)

[Select Input type 3](#_Toc349847540)

[Table Exports 4](#_Toc349847541)

[Admin Access 4](#_Toc349847542)

[Account Creation 5](#_Toc349847543)

[Public URLS 5](#_Toc349847544)

[Mobile Forms 6](#_Toc349847545)

[Statistics 6](#_Toc349847546)

[Payment 7](#_Toc349847547)

[Implemented User Stories 7](#_Toc349847548)

[Story ID: 1 – Log In 7](#_Toc349847549)

[Story ID: 2 – Generate Forms 8](#_Toc349847550)

[Story ID: 3 8](#_Toc349847551)

[Story ID: 6 – Account Creation 8](#_Toc349847552)

[Class Diagram 8](#_Toc349847553)

[Database Schema 9](#_Toc349847554)

[Entity Framework 9](#_Toc349847555)

[SMO Database 10](#_Toc349847556)

[Revised Release Plan 11](#_Toc349847557)

[Difficulties and Learning Outcomes 12](#_Toc349847558)

[Story ID: 1 – Log In 12](#_Toc349847559)

[Profile Linking 12](#_Toc349847560)

[Story ID: 2 – Generate Forms 12](#_Toc349847561)

[Unique Table IDs 12](#_Toc349847562)

[Cascade Delete 12](#_Toc349847563)

[Views using Column Names 12](#_Toc349847564)

[Reading data back from SQL Database 12](#_Toc349847565)

[Story ID: 6 – Account Creation 13](#_Toc349847566)

[Small issues 13](#_Toc349847567)

# Application Architecture Definition

## MVC

The Model View Controller architecture is being implemented in this project.

MVC is an Object Orientated development methodology to enable code reuse by separating the application into 3 parts: the Model, View and Controller.

The model is charged with managing fundamental behaviours and the data for the application, typically seen implemented as a database. The view is the layer at which the user may interact with the application. It is the user interface and is responsible for rendering data passed to it from the model. The controller handles user input via the view layer and makes calls to model objects.

# Revised User Stories

### Log In

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 1** | **Story Title: Log In** | **Priority: High** | **Estimate: 2 hours** |
| As a user I want to log in so that I can access my account | | | |

**Confirmation:**

1. **Success** – valid user logged in and referred to personal home page
   1. User Presented with a list of their associated sheets table entries.

Which can be selected to view the individual signatories.

1. **Failure** – display message
   1. Username and hash of password does not match that stored in the database.
   2. Display failed login message, allow retry up to 3 attempts.

### Generate Form

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 2** | **Story Title: Generate Form** | **Priority: High** | **Estimate: 5 hours** |
| As a User I want to generate custom registration forms so that I can capture data from clients | | | |

**Confirmation:**

1. **Success** – Valid table names provided, table created, stored and associated with user profile
   1. User returned to list of tables including newly created one.
2. **Failure** – Invalid Input
   1. User enters values disallowed by form validation
      1. Indicated by highlighting invalid field
   2. Failed table creation
      1. Notify user and allow retry

### Select Input type

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 3** | **Story Title: Select Input Type** | **Priority: High** | **Estimate: 5 hours** |
| As a User I want to select predetermined input types (radio buttons, checkboxes) for certain custom fields, and be able to specify custom answers. | | | |

**Confirmation:**

1. **Success** – Table created with defined input types and default question layout.
   1. User returned to list of tables including newly created one.
2. **Failure** – Invalid Input
   1. User enters values disallowed by form validation
      1. Indicated by highlighting invalid field
   2. Failed table creation
      1. Notify user and allow retry

### 

### Table Exports

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 4** | **Story Title: Table Exports** | **Priority: High** | **Estimate: 5 Hours** |
| As a User I want to export Tables so that I can easily import them to other applications. | | | |

**Confirmation:**

1. **Success** – File downloaded to client machine.
   1. File downloaded to client machine at user defined location
2. **Failure** – Cannot export table.
   1. Table may be being updated by another device (AJAX call)
   2. Table not found

### 

### Admin Access

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 5** | **Story Title: Admin access** | **Priority: High** | **Estimate:4 Hours** |
| As a sit administrator I want access users information to build profiles and send emails. | | | |

**Confirmation:**

1. **Success** – View list of usernames, full names, location and email addresses.
   1. View displayed to admin
2. **Failure** –
   1. Detail failures & messages

### 

### Account Creation

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 6** | **Story Title: Account Creation** | **Priority: High** | **Estimate: 5 Hours** |
| As a User I want to be able to create new account so that I can log in. | | | |

**Confirmation:**

1. **Success** – User account created
   1. user redirected personal homepage
2. **Failure** – Cannot create account, display message
   1. Account already exists
   2. Invalid inputs/password does not match

### 

### Public URLS

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 7** | **Story Title: Public URLs** | **Priority: Medium** | **Estimate:** |
| As a User I want to share public links to social networking sites so that clients can complete form remotely. | | | |

**Confirmation:**

1. **Success** – Public URL provided to user
   1. Option to share directly to social networking sites
2. **Failure** – Unable to provide public URL

### Mobile Forms

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 8** | **Story Title: Mobile forms** | **Priority: High** | **Estimate:** |
| As a User I want to populate tables from the mobile app so that I can present the form to customers in person | | | |

**Confirmation:**

1. **Success** – User verified and tables associated with user displayed
   1. User selects table to populate
   2. Form presented to screen
2. **Failure** – No Tables to display
   1. User provided with option of creating new table.

### Statistics

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 9** | **Story Title: Statistics** | **Priority: Medium** | **Estimate: 2 Days** |
| As a User I want to analyse table data so that I can better target and understand my clients and determine target markets | | | |

**Confirmation:**

1. **Success** – Table analytics displayed
   1. User can filter results into categories
2. **Failure** – No Analytics available
   1. Table empty
   2. No data fields in table are capable of having analytics associated with them

### Payment

|  |  |  |  |
| --- | --- | --- | --- |
| **ID: 10** | **Story Title: Payment** | **Priority: Low** | **Estimate:**  **2 Weeks** |
| As a User I want Clients to pay subscription fees in-line with the registration form. | | | |

**Confirmation:**

1. **Success** – User logs in with 3rd party payment details (e.g. PayPal, PayClick, WePay, etc.)
   1. Details verified, payment received.
2. **Failure** – Details Not verified
   1. User notified of payment failure and returned to form

# Implemented User Stories

## Story ID: 1 – Log In

User can log in to the application. Once acceptable log in criteria are entered, the user is redirected to their personal home screen.

This view allows the user to see any of their existing forms or create new forms. Forms may be edited, deleted or submitted to the SQL. Once submitted, data can be entered into the table through the user view. Any data which has been entered into the form can be viewed by clicking the form name from the user home page.

## Story ID: 2 – Generate Forms

The user can create new forms by providing the Form name and any required form fields. When submitting the form to the SQL database, a unique identifier for the form is created by concatenating the users’ ID and the form ID. Once submitted, the action is confirmed and the form is ready to be used.

## Story ID: 3

When generating forms and their fields, the user must specify of what type (Text, Number, Date, etc.) input the field should expect. This information is used when creating the SQL database tables for the forms and for formatting the user view displays.

## Story ID: 6 – Account Creation

Users must supply a username, email address and password (which must be entered twice) to register an account. Once a valid input is received, the account is created and the user is logged in and presented with their personal home page.

# Class Diagram



# Database Schema

### Entity Framework



### SMO Database



# C:\Users\shanemurphy\Dropbox\3rdYearImagineCup\Submitions\Upload 2 - 1st March\ReleasePlan.PNGRevised Release Plan

# Difficulties and Learning Outcomes

As we began coding our solution, we quickly discovered our experience of .NET and C# to be very basic. This proved to be the main obstacle we needed to overcome in this iteration, as we built on existing knowledge to understand the .NET framework and C# at a much deeper level.

We increased time spent on the system by a number of hours to compensate for time spent searching for coding solutions and logical solutions to our problems. Because of this, we managed to stay on course and complete all tasks and user stories required for Iteration 1.

At the end of this iteration, it was decided to implement error logging to better document, keep track of and understand our problems.

## Story ID: 1 – Log In

### Profile Linking

Linking the users’ profile, managed using SimpleMembership with their associated tables, which were generated and stored on a separate SQL Server database. To do this, we needed to change the standard SimpleMembership profile, connect this profile to our own Model Classes and get the User ID using WebSecurity.

## Story ID: 2 – Generate Forms

### Unique Table IDs

To store individual user tables, we needed to ensure that all table names were unique. To do this, we appended the User ID and Form ID to the table name. E.g. UserTable\_1\_2 for User ID = 1, Form ID = 2.

### Cascade Delete

Our application requires the use of Entity Framework and a Separate SQL Server. To keep our databases in sync, we needed to ensure deleting the Entity Framework table would also drop the SQL equivalent table.

### Views using Column Names

Column names, which were taking as input from the user, needed to be passed to a view to display this data to the user again. Because these columns were actually stored in an Entity Framework table as data in the table, we needed a way to pass this data to the view, rather than the actual column names (which display as Field 1, Field 2, Field 3, etc.)

### Reading data back from SQL Database

We found it challenging to read any data back from our SQL Server tables into our system. Past experience with Java and SQL through JDBC proved to be extremely useful to us in this task, however returning information to the application via a result set, as in JDBC, was not an option.

## Story ID: 6 – Account Creation

### Small issues

We encountered many issues working with 2 databases simultaneously. Using SMO with the SQL Database and managing User Profiles using Entity Framework brought on many small issues regarding combining data, authenticating data and returning data to user views.