**Detailed Design**

Workforce

Research

Guide

**Team: Group2**

Shin Yi Lin

Fenil Shah

Ramprasadh Srivathsa

Harshal Pawar

Mark Dwight

SE 6387.001

**ABSTRACT**

This document aims to document the detailed design of the Workforce Research Guide application. This document includes design of user interface as well as the system’s backend part. The user interface is depicted using the images of the proposed design. The static model of the system is depicted via class diagram. Sequence diagrams are used to depict the dynamic behavior of the system for each of the use case. Lastly, we include traceability matrix to correlate the requirements to the design.

**TABLE OF CONTENTS**

1. Introduction……………………………………………………………………………………………………………………………4

2. Graphical User Interface (GUI) design..………..……………………..…..…………….………………………………4

3. Static Model – Class Diagram..……………………………………………………………………………………………….5

4. Dynamic Model – Sequence Diagram……………………………….……………………………………………………6

5. Rationale for Detailed Design Model……………………………………………..………………………………………7

6. Traceability Matrix………………………….……………………………………………..………………………………………7

7. Evidence that the document have been placed under configuration management……….……….8

8. References……………………………………………………………………………………………………………………………..7

**LIST OF FIGURES**

Figure 1 Home Screen - GUI 5

Figure 2 Add Entity - GUI 6

Figure 3 Template Manager - GUI 7

Figure 4 Add Template – GUI 7

Figure 5 Verify Delete Template - GUI 8

Figure 6 Verify Delete Entity 8

Figure 7 Class Diagram of system 9

Figure 8 Add Entity - SD 10

Figure 9 Add Entity - batch - SD 11

Figure 10 Add Entity - folder scan - SD 12

Figure 11 Add Template - SD 13

Figure 12 Delete Entity – SD 13

Figure 13 Delete Template - SD 14

Figure 14 Edit Entity - SD 14

Figure 15 Search Entity - SD 15

Figure 16 View Entity - SD 15

**LIST OF TABLES**

[Table 1 Traceability Matrix 16](#_Toc446093599)

**1. INTRODUCTION**

This document serves to outline the details regarding the design of the Workforce Research application. To do this, we illustrate the flow and feel of the application via mockups of each screen. We then show the static model for the program via the class diagram and the dynamic model for the program case via the sequence diagrams. We finish by giving our rationale for our design decisions and tracing our previously defined requirements to the associated design element.

**2. GRAPHICAL USER INTERFACE (GUI) DESIGN**

This section gives an overall idea of the user interface of the application. Following are the screen designs for the application.

1. **Home Screen**

The right panel will list all the entities. The user can search using the search box in top left panel. The results will be listed into the right panel. When the user selects one of the entities from list, the details will be filled up in bottom left panel.

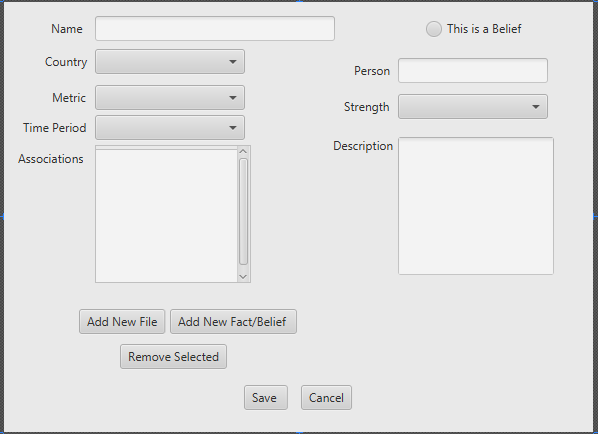


Figure 1 Home Screen - GUI

1. **Add entity**

User can click on Add Fact/Belief on the home screen and this screen will be prompted. User can enter all the details here and add related associations. Then click Save to create a new entity into repository.

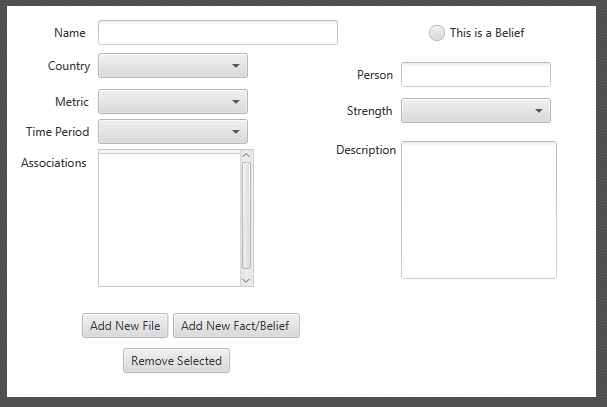


Figure 2 Add Entity - GUI

1. **Template Manager**

Template has contains three fields – country, metric and time period. The template manager screen shows list of all possible values for each of these three fields. From here, user can select of the entry for a particular field and delete it. User can also add new value for a particular field which will be available while adding new entities afterwards.

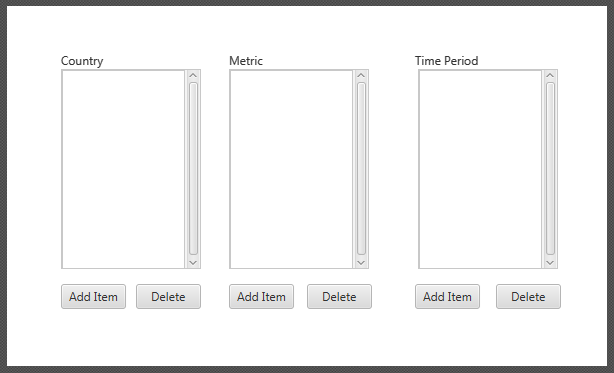


Figure 3 Template Manager - GUI

1. **Add template**

When user clicks on Add item button on the template manager screen, this screen is prompted. User will enter the textual value and click add to add that text as an option in that dropdown field.

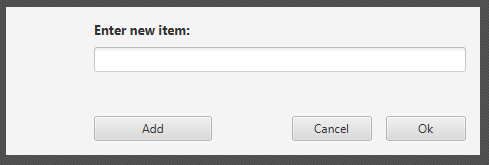


Figure 4 Add Template – GUI

1. **Delete template verify**

The following prompt is shown to confirm the deletion of the template.

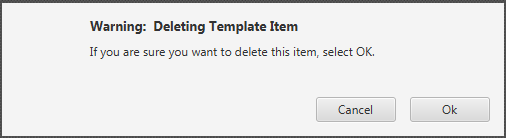


Figure 5 Verify Delete Template - GUI

1. **Delete entity verify**

The following prompt is shown to confirm the deletion of the entity.

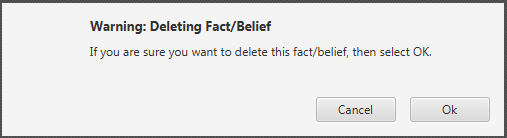


Figure 6 Verify Delete Entity

**3. STATIC MODEL – CLASS DIAGRAM**

AppHandler is the application level controller for the system. All the user requests goes through AppHandler. If the user request is related to the entity processing, then the request is delegated to EntityProcessor. If the request is related to template processing, then it is delegated to TemplateProcessor. The DBHandler class provides mechanism to interact with the data repository. Thus EntityProcessor is responsible for Entity management, TeplateProcessor is responsible for Template management and DBHandler for interacting with database. EntityProcessor and TemplateProcessor delegates database interaction requests to DBHandler. The AppController, EntityPRocessor and TemplateProcessor are singleton classes.

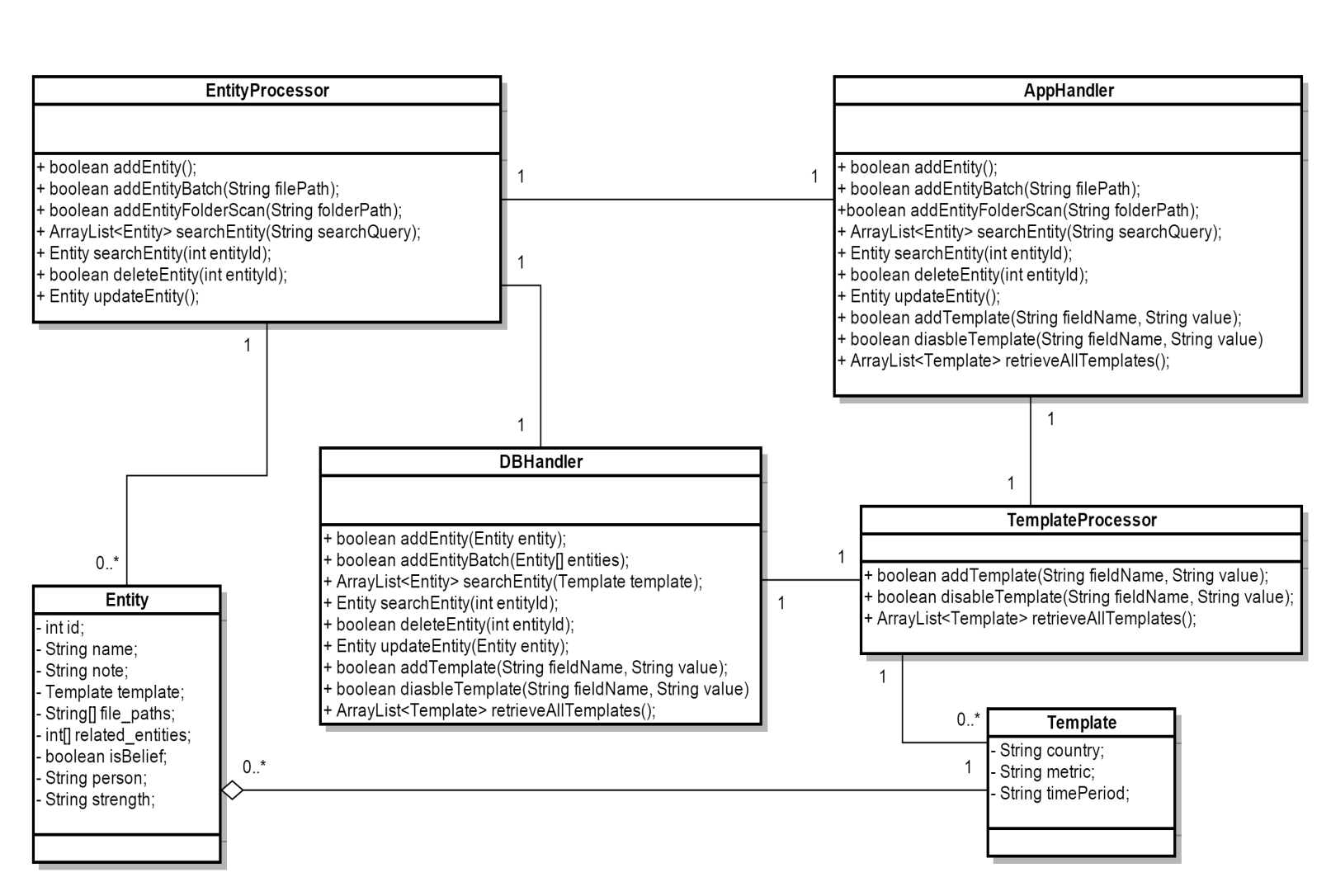


Figure 7 Class Diagram of system

**4. DYNAMIC MODEL – SEQUENCE DIAGRAM**

1. **Add Entity**

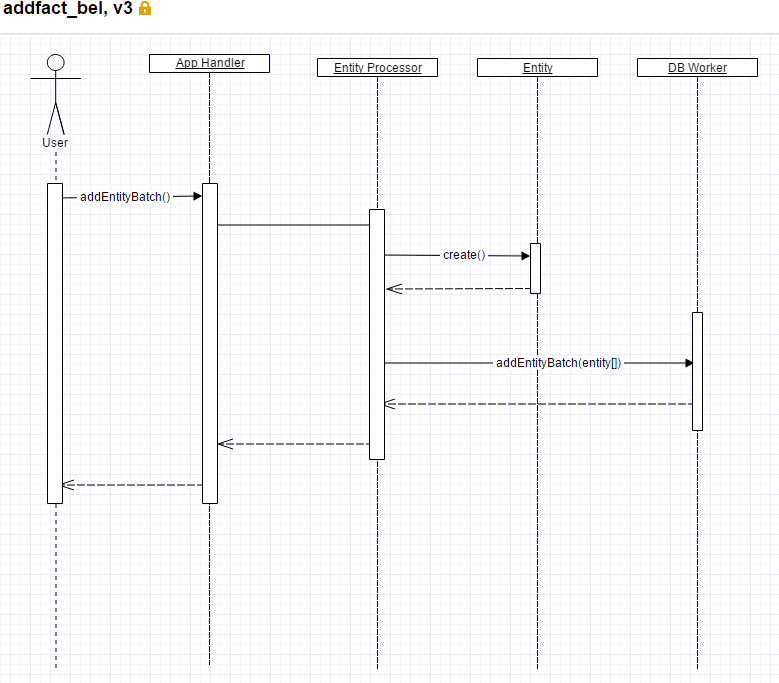


Figure 8 Add Entity - SD

1. **Add Entity – batch**

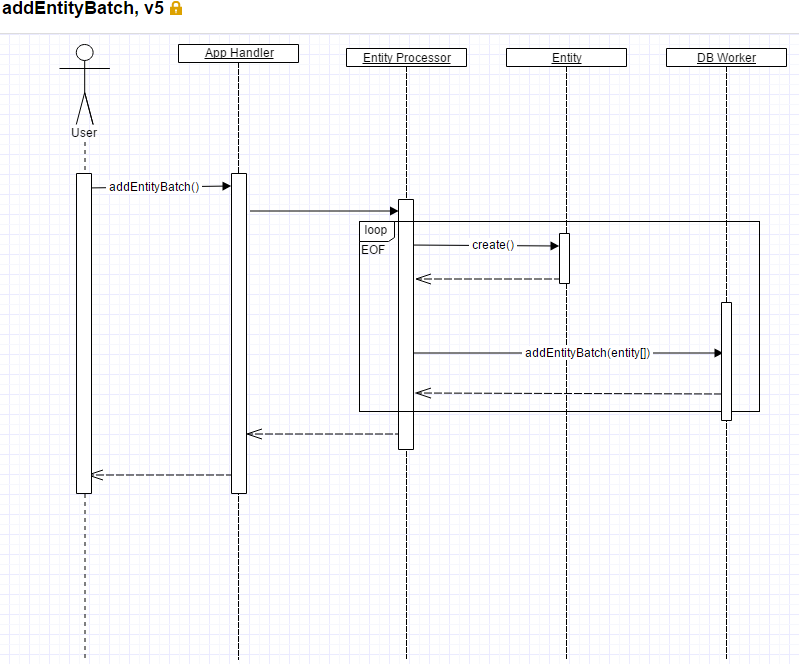


Figure 9 Add Entity - batch - SD

1. **Add Entity – folder scan**

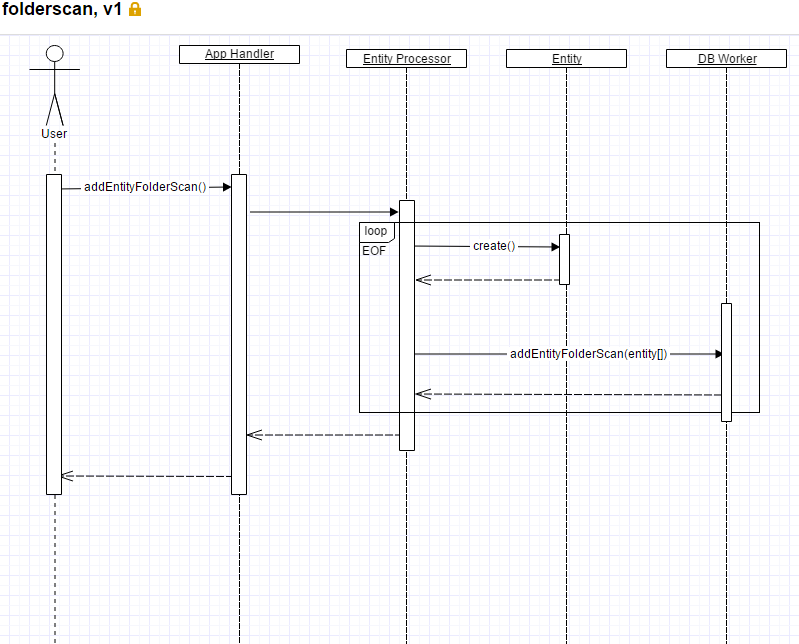


Figure 10 Add Entity - folder scan - SD

1. **Add Template**

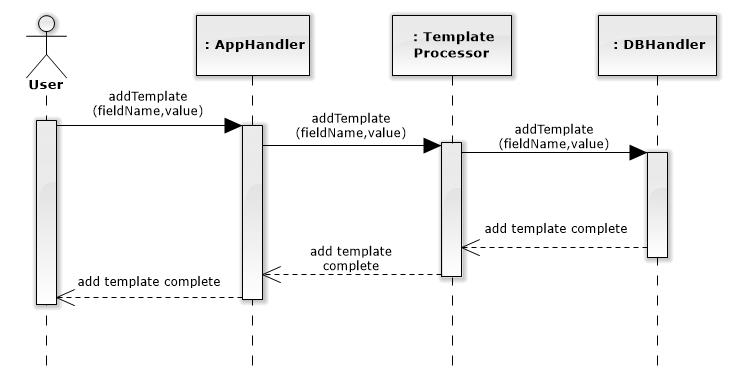


Figure 11 Add Template - SD

1. **Delete Entity**

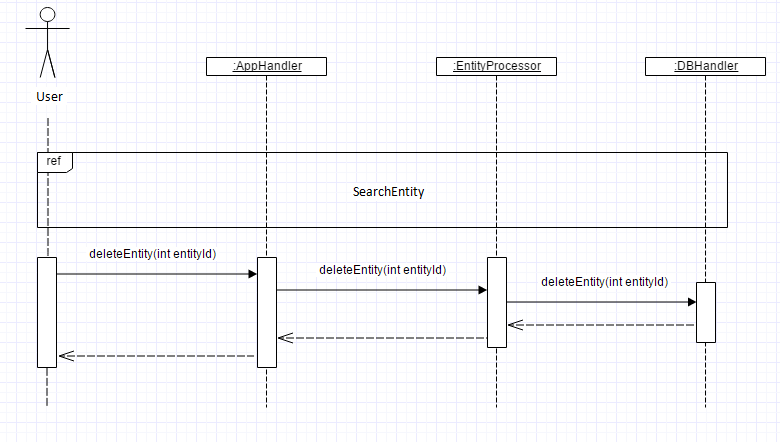


Figure 12 Delete Entity – SD

1. **Delete Template**

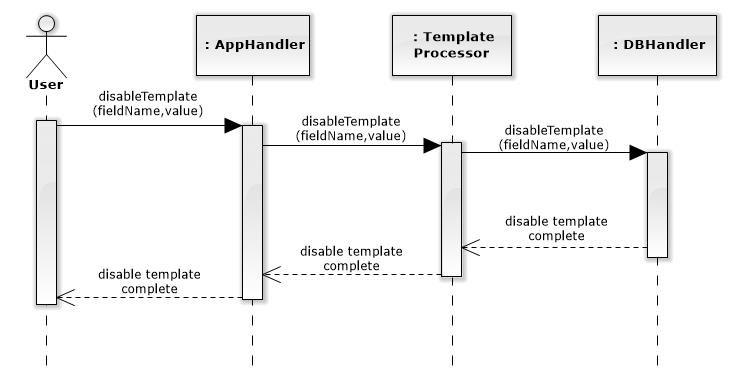


Figure 13 Delete Template - SD

1. **Edit Entity**

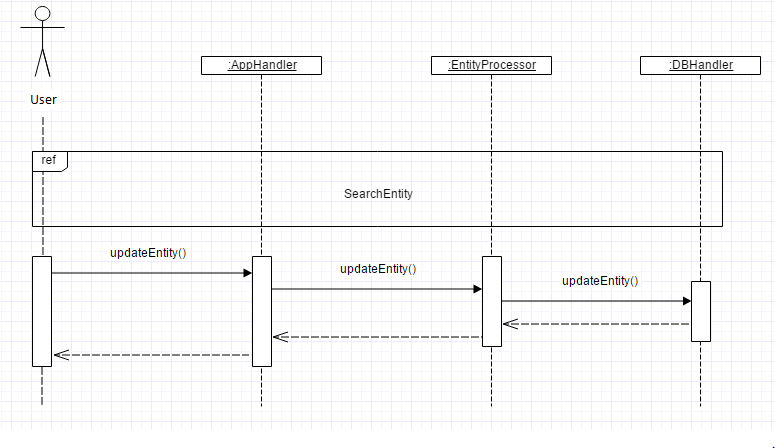


Figure 14 Edit Entity - SD

1. **Search Entity**

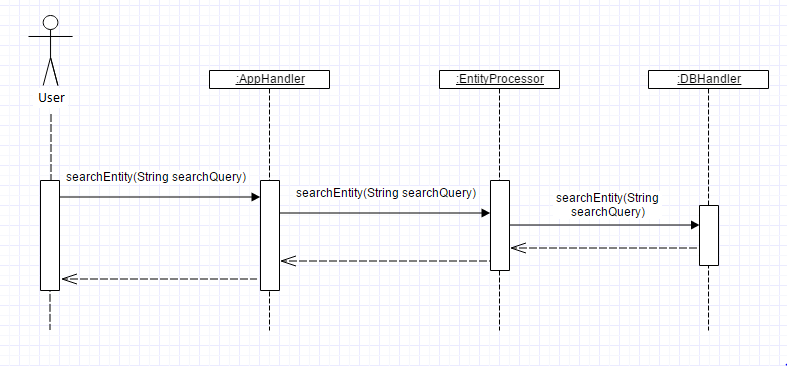


Figure 15 Search Entity - SD

1. **View Entity**

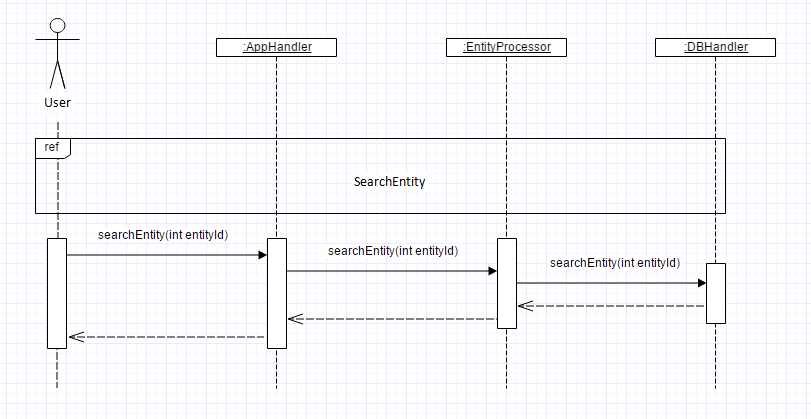


Figure 16 View Entity - SD

**5. RATIONALE FOR DETAILED DESIGN MODEL**

The above design elements were created with the intent of fulfilling all of the outlined requirements for the system while also providing the framework for the object-oriented solution to be developed.  The class diagram serves to outline the various logical components that make up the system and each sequence diagram serves to outline how these components interact to perform the specific functions necessary to fulfill each requirement.  These combined with the graphical user interface mockup images provide a model for the structure and flow of the program, allowing both the customer as well as the developers a more complete understanding of how the end product will look and eventually operate.

The specific details of the design were also chosen to reflect the architectural style and model that were previously outlined.  The Model-View-Controller (MVC) model necessitates the existence of certain components to divide responsibility of functionality, namely regarding the separation of the user interface from background functions.  The existence of a data repository necessitates the existence of a component to handle the interaction between the repository and the rest of the program.  The class diagram was created with these architectural decisions in mind, as well as with general separation of responsibility inherent in good object oriented design practices.

**6. TRACEABILITY MAATRIX**

|  |  |
| --- | --- |
| **Requirements – Use Case** | **Detailed Design Element – Sequence Diagram** |
| UC1: Add Entity | SD I: Add Entity |
| UC2: Batch Adding of Facts | SD II: Add Entity – batch |
| UC3: Adding Facts via Folder Scan | SD III: Add Entity – folder scan |
| UC4: Search Entity | SD VIII: Search Entity |
| UC5: View Entity | SD IX: View Entity |
| UC6: Delete Entity | SD V: Delete Entity |
| UC7: Edit Entity | SD VII: Edit Entity |
| UC8: Add Metadata Item | SD IV: Add Template |
| UC9: Delete Metadata Item | SD VI: Delete Template­­­­­­­­­­­ |

Table 1 Traceability Matrix

**7. EVIDENCE THE DOCUMENT HAS BEEN PLACED UNDER CONFIGURATION**

**MANAGEMENT**

GitHub Repository:

<https://github.com/WorkforceResearchGuide/WorkforceResearchGuideApp.git>

**8. REFERENCES**

1. <http://homepages.uel.ac.uk/D.Bowden/Creating_Class_Diagram/Creating_Class_Diagram.viewlet/Creating_Class_Diagram.swf>
2. <http://homepages.uel.ac.uk/D.Bowden/Creating_Sequence_Diagrams/Creating_Sequence_Diagrams.viewlet/Creating_Sequence_Diagrams.swf>