**Final Report**

**Workforce**

**Research**

**Guide**

**Team: Group2**

**Fenil Shah**

**Harshal Pawar**

**Mark Dwight**

**Ramprasadh Srivathsa**

**Shin Yi Lin**

**SE 6387.001**

**Executive Summary**

This document serves as the final report for the course project of SE 6387 class. The project is completed during the Spring 2016 semester at the University of Texas at Dallas. Here we outline the project management plan, requirements, architecture, detailed design, and the testing plan implemented during the course of the project.

The purpose of the project was to develop an application to assist our sponsor, Alliance Data, in organizing and linking facts and beliefs held by their organization and sub-organizations regarding business metrics.

**Table of Contents**

[1. INTRODUCTION](#h.yfqwsgeh3f7l)

[1.1 Introduction, Purpose and Scope](#h.yhxc7sd2865w)

[1.2 Product Overview](#h.95inb26vdc9v)

[1.3 Structure of the Document](#h.reftv6j7jgel)

[1.4. Terms, Acronyms, and Abbreviations](#h.209yy1yudfd2)

[2. Project Management Plan](#_2._Project_Management)

[2.1 Project Organization](#h.wczyxh2o14cv)

[2.2 Lifecycle Model Used](#h.lkkihkck2mex)

[2.3 Risk Analysis](#h.76ztxdjf640n)

[2.4 Hardware and Software Resource Requirements](#h.emirexb79re6)

[2.5 Deliverables and Schedule](#h.geftolgrfzxh)

[2.6 Monitoring, Reporting, and Controlling Mechanisms](#h.j8cu1jwcdmnf)

[2.7 Professional Standards](#h.scfyizzg6nlp)

[2.8 Evidence that all artifacts have been placed under configuration management](#h.x0yhxctq9ie9)

[2.9 Impact of the project on all individuals and organizations](#h.ocqfersp708v)

[3. Requirement Specifications](#h.gae871cpfw1x)

[3.1 Stakeholders for the System](#h.d7dd3wwp5ieq)

[3.2 Use Case Model](#h.8xdwb4yh3n92)

[3.2.1 Graphic Use Case Model](#h.i3kya8ej5w0n)

[3.2.2 Use Case Textual Descriptions](#h.t8shyqcja481)

[3.3 Rationale for Use Case Model](#h.xf414bi0nson)

[3.4 Non-functional Requirements](#h.7ap82kqqwwjb)

[4. Architecture](#h.6yurhjl16o5z)

[4.1 Architectural Style Used](#h.hyhz2cv6qyc7)

[4.2 Architectural Model](#h.o73aid6irbl1)

[4.3 Technology, Software and Hardware Used](#h.98g9p3s0b2kq)

[4.4 Rationale for Architectural Style and Model](#h.w61b4ju8bz3m)

[5. Design](#h.x8c1fo9d50mj)

[5.1 Graphical User Interface Design](#h.2tx0vuhqs5bq)

[5.2 Static Model – Class Diagrams](#h.7tc8bfzbezvy)

[5.3 Dynamic Model – Sequence Diagrams](#h.vsywe1d0558n)

[5.4 Rationale for Detailed Design Model](#h.vvhz6zqr03yz)

[5.5 Traceability from Requirements to Detailed Design](#h.gjdgxs)

[6. Test Plan](#h.hxlwvqp33xg5)

[6.1. Requirements/specifications-based system level test cases](#h.kri1ji2avsir)

[6.2. Traceability of test cases to use cases](#h.uta5q8wyn1zg)

[6.3. Techniques used for test generation](#h.17a4ikyaau6q)

[6.4. Assessment of the goodness of your test suite](#h.2hbuejg5dvat)

[Acknowledgements](#h.kdmq1rc11b77)

[References](#h.n99no8qwlpuk)

**List of Figures**

[Figure 1 Image used by Boehm to illustrate his ‘Spiral Model’ for Software Development 10](#_Toc449376941)

[Figure 2 Use case diagram 15](#_Toc449376942)

[Figure 3 Architecture Diagram 26](#_Toc449376943)

[Figure 4 Home Screen - GUI 28](#_Toc449376944)

[Figure 5 Add Entity - GUI 29](#_Toc449376945)

[Figure 6 Template Manager - GUI 30](#_Toc449376946)

[Figure 7 Add Template – GUI 30](#_Toc449376947)

[Figure 8 Edit entity - GUI 31](#_Toc449376948)

[Figure 9 Class Diagram of system 32](#_Toc449376949)

[Figure 10 Add Entity - SD 33](#_Toc449376950)

[Figure 11 Add Entity - batch - SD 34](#_Toc449376951)

[Figure 12 Add Entity - folder scan - SD 35](#_Toc449376952)

[Figure 13 Add Template - SD 36](#_Toc449376953)

[Figure 14 Delete Entity – SD 36](#_Toc449376954)

[Figure 15 Delete Template - SD 37](#_Toc449376955)

[Figure 16 Edit Entity - SD 37](#_Toc449376956)

[Figure 17 Search Entity - SD 38](#_Toc449376957)

[Figure 18 View Entity - SD 38](#_Toc449376958)

[Figure 19 Differences between Black Box and White Box Testing 55](#_Toc449376959)

**List of Tables**

[Table 1 Project Responsibilities 9](#_Toc449376988)

[Table 2 Project deliverables and schedule 11](#_Toc449376989)

[Table 3 Add Fact/Belief use case description 16](#_Toc449376990)

[Table 4 Batch adding of facts use case description 17](#_Toc449376991)

[Table 5 Adding facts via folder scan use case description 18](#_Toc449376992)

[Table 6 Search Fact/Belief use case description 18](#_Toc449376993)

[Table 7 View Fact/Belief use case description 19](#_Toc449376994)

[Table 8 Delete Fact/Belief use case description 20](#_Toc449376995)

[Table 9 Edit Fact/Belief use case description 20](#_Toc449376996)

[Table 10 Add Fact/Belief meta-data item 21](#_Toc449376997)

[Table 11 Delete Fact/Belief meta-data item 22](#_Toc449376998)

[Table 12 Traceability Matrix - Requirements to Detailed design 39](#_Toc449376999)

[Tables of Test cases 40](#_6.1._Requirements/specifications-ba)

[Table 13 Traceability matrix - Tests to Use case 53](#_Toc449377000)

## 1. INTRODUCTION

### 1.1 Introduction, Purpose and Scope

This is the final report for the Workforce Research Guide project. This project is part of the SE 6387 course project at the University of Texas at Dallas and completed during Spring 2016 semester.

This document serves as the final deliverable for the completed project. It is organized into several parts which explain our initial project plan, requirements, architecture, detailed design as well as testing plan.

### 1.2 Product Overview

The team has developed Workforce Research Guide for Alliance Data. The purpose of this product is to help the management easily organize and correlate their facts and beliefs. The HR department of every organization occasionally needs to study their internal data and derive conclusions about the organization environment. The reason behind this can be anything like employee growth rate, employee turnover rate, financial matters, etc. So they need to correlate reports and other information generated by the various department systems and derive an overall conclusion. They can use this research information to improve the working of the organization and take informed decisions.

The HR representatives and other management people have some beliefs/ assumptions which might be based on some information. However, it doesn’t blatantly imply that these beliefs are facts. To consider those as facts, those have to be proven and backed by research data. So they generate and go through various reports and at the end of the day come up with concrete facts. These facts prove/disprove the previously held beliefs. Producing these facts is a time-consuming task and currently, there is no good solution to keep track of all such research information. So Workforce Research Guide will provide means to organize the beliefs and facts and also supporting information. The management will be able to add new beliefs as they encounter and after thorough research they can store the analyzed facts into the system with respect to the beliefs and relate the supporting documents.

So in future, if a new employee has some belief then first they can go through the existing information base and see if it was recently researched by someone. This can save huge amount of resources in terms of employee time by reducing the rework and encourage collaboration.

### 1.3 Structure of the Document

This document is structured as follows. First, we describe the project management plan decided upon at the start of the project. We then describe the specified requirements captured using the use case model. Next, we describe the overall architectural style employed. We then describe the specific design of the project, including graphical user interface design as well as static and dynamic models via class and sequence diagrams respectfully. Finally, we describe the test plan employed to test the end product, including our assessment of the goodness of the test suite.

### 1.4. Terms, Acronyms, and Abbreviations

* UC diagram – Use case diagram
* GUI – Graphical user interface
* SD – Sequence diagram
* TC – Test case

## 2. Project Management Plan

### 2.1 Project Organization

The team roles are divided into Project Manager, System Analyst, Software Engineer, UI Developer and Test Engineer:

1. **Project Manager**: Communicates with the customer, monitors the schedule and working process, coordinates between each team member, and solves the problems between them.
2. **System Analyst**: Analyzes the hardware and software parts of the system needed. Hardware part may involve HR system, financial system, and external third party system. Excel format of report should be imported from these systems into the developed software. Software part involves using Java programming language in Eclipse, UML design tool for clarifying developed goal, etc.
3. **Software Engineer**: finds the customer’s requirement, designs the software architecture concerning about functional and nonfunctional requirement.
4. **User Interface Developer**: Develops the easy-to-use and pretty interfaces, connects interfaces and the logic functionality, and performs unit testing.
5. **Test Engineer**: Does the unit testing as well as integration testing.

Below we list each team member’s primary responsibility with regards to the general organization of the project. We note, however, that each member is free to assist with the other project portions as needed, so effectively each member will be actively contributing to all portions of the project.

Table 1 Project Responsibilities

|  |  |  |
| --- | --- | --- |
| Name | Primary Responsibilities | Contact Information |
| Harshal Pawar | Software Engineer | hvp130230@utdallas.edu |
| Shin-Yi Lin | Software Engineer | sxl148530@utdallas.edu |
| Mark Dwight | UI Developer | mhd100020@utdallas.edu |
| Fenil Shah | Project Manager, Software Engineer | fxs140830@utdallas.edu |
| Ramprasadh Srivathsa | Software Engineer, Test Engineer | rxs140130@utdallas.edu |

### 2.2 Lifecycle Model Used

For this project, we plan to use the spiral model. We chose this lifecycle model as it enables us to more readily gain feedback from our sponsor to facilitate the creation of the product that best fits their needs.

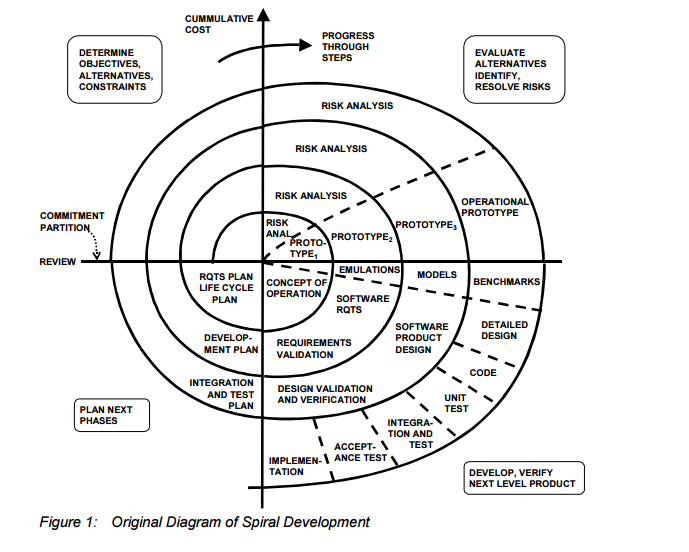


Figure 1 Image used by Boehm to illustrate his ‘Spiral Model’ for Software Development

### 2.3 Risk Analysis

Risks for this project included failure of team member(s) to contribute and complete the assigned tasks as well as the inability to deliver a functioning product by the end of the semester.  The former was dealt in-accordance to the professional standards guideline outlined in Section 2.7 while the latter was handled over the course of development by adjusting expectations for the end product based on the customer’s changing requirements, remaining time and feasibility. We tried our best to accommodate as much as possible to make the final product as much professional as possible. Also, we were always planning in advance for all the meetings so it gave us enough time to work on each deliverable and handle the unexpected situations.

### 2.4 Hardware and Software Resource Requirements

Each member has used their own Windows computer, either desktop or laptop, during the course of development.  The product has been developed in an IDE such as NetBeans or Eclipse and written in an object oriented programming language, namely Java. For the purposes of version control and configuration management, we used Github. We used Google Drive briefly for informal collaboration purpose.

In terms of new software and hardware, our team became familiar with utilizing JavaFX for developing the user interface and SQLite for developing the database portion of the program. Team members also became familiar with using Github’s shell interface when pushing and pulling updates to the project. Moreover, we learnt how to utilize the source control efficiently so that each of us can use different IDEs as per own convenience to work productively.

### 2.5 Deliverables and Schedule

The schedule for delivering each of the documents is given below. Also, as we followed the spiral model as software development process, we always submitted our updated documents along with each of the due deliverable.

Table 2 Project deliverables and schedule

|  |  |  |
| --- | --- | --- |
| Deliverable | Due Date | Description |
| Project Management Plan | 01/29/2016 | This document describes a clear understanding of the project boundaries, with main contributions structured as deliverables. |
| Requirements Documentation | 02/12/2016 | This document describes what the product should do. It contains all the functional and nonfunctional requirements of the product. |
| Architecture Documentation | 02/26/2016 | This document describes the software architecture model that is to be used in the project. |
| Detailed Design Documentation | 03/18/2016 | This document gives a detailed description of design of the product. |
| Testing Plan | 04/08/2016 | This document outlines the test cases to be used to test the product, the traceability of said test cases to their associated use cases, and the techniques used to generate each test case. |
| Final Project Report and demonstration | 04/29/2016 | The report shall be a compilation of all the previous documents with an assessment of the goodness of the test suite. The demonstration will be a live exhibition of the end product. |

### 2.6 Monitoring, Reporting, and Controlling Mechanisms

The team leader was primarily responsible for scheduling and communicating about the meetings to everyone.

We had regular planning meeting for each of the deliverables described in the previous section. During the planning meeting we identify the tasks for that deliverable and each of the team-member took-up the tasks. We kept track of this in a shared word document internally.

To keep track of the documents as well as codebase, we used Github as the shared repository. Everyone on the team has add, remove and modify permissions all the files of the repository. Having all the permissions made it easy to take help from teammates. This policy worked out very well for us.

We had weekly meetings on Friday with our sponsor to discuss the project progress and next steps. Our meetings took place on-campus at UT Dallas Library Room, mostly in Room 2.540 (Lone Star Room), during 1:00 PM to 4:00 PM and in-person meeting. We also had our weekly group meeting at the same place before/after the meeting with the sponsor.

### 2.7 Professional Standards

The team members are expected to perform with 100% commitment. Meetings will be held every Friday tentatively. Depending on the availability of sponsor and team members, it is subject to change. All team members are expected to actively attend the meeting. In the event that they can’t do so, they must inform the Project Manager/team members. We believe in quality of work and not quantity. All team members are expected to perform the assigned tasks within the timeframe and also meeting the norms set by the Professor and the team. Any disputes within the team would be first discussed within the team & with team leader. If not resolved, then the Professor would be contacted and evidences will be presented in front of him. Professor’s decision will be final.

### 2.8 Evidence that all artifacts have been placed under configuration management

Please navigate to Deliverables directory for all the reports.

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

### 2.9 Impact of the project on all individuals and organizations

This project provides a flexible and extensible solution for the problem of correlating, organizing and reconciling facts and beliefs held by an organization. By using this software, the HR department and other people at Alliance Data will be able to organize their facts and beliefs in a proper way. They will be able add relationships between those and existing documents as well. This way it becomes very easy for them to control the redundancy and save employee efforts on the same topics.

Though this project was developed with the needs of our sponsor in mind, the resulting code can potentially be used by other organizations that seek a way to organize their internal information in a similar manner.

## 3. Requirement Specifications

### 3.1 Stakeholders for the System

For this project, stakeholders include the following:

* **Alliance Data**: Our sponsor/customer, who desires a custom solution to solve the issue of tracking organizational facts and beliefs
* **Jeffrey Buchmiller**: He and his colleagues are the primary users of the application
* **Team members**: We seek to create a quality product that satisfies the customer requirements. Following are the team members.
  + Fenil Shah
  + Harshal Pawar
  + Mark Dwight
  + Ramprasadh Srivathsa
  + Shin-Yi Lin
* **Professor W. Eric Wong**: Seeks to encourage the students to learn new technologies and grow. He evaluates the project and gives constructive feedback.
* **The University of Texas at Dallas**: Seeks to produce outstanding Computer Science and Software Engineering students as well as foster good relationships with local industry

### 3.2 Use Case Model

#### 3.2.1 Graphic Use Case Model

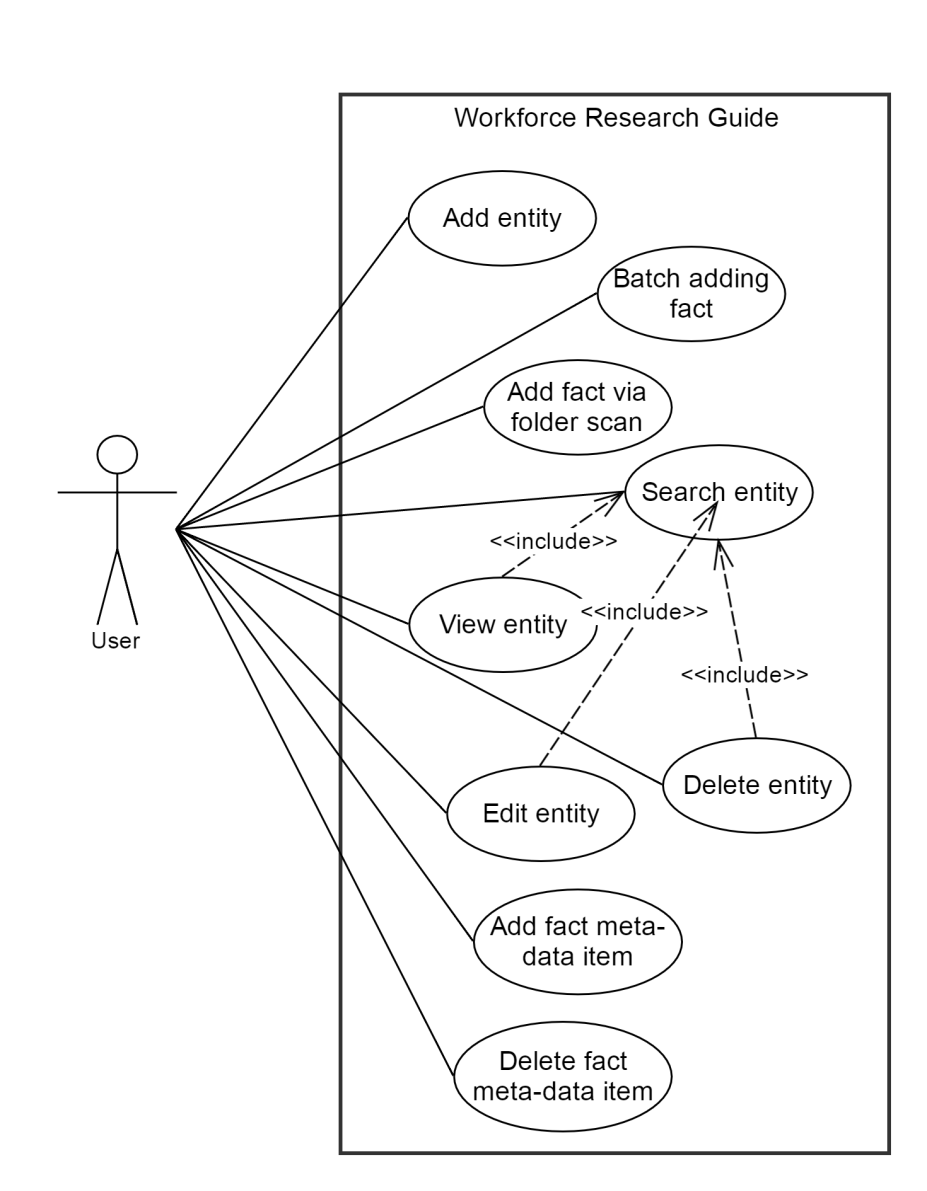


Figure 2 Use case diagram

#### 3.2.2 Use Case Textual Descriptions

Table 3 Add Fact/Belief use case description

|  |
| --- |
| **Use Case: Add Fact/Belief** |
| **ID**: 1 |
| **Brief Description**:  The user enters information about a fact or belief so that it can be stored by the software for future reference. |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:** |
| **Main Flow:**   1. The user selects the option to add a fact/belief. 2. The user enters information regarding that fact/belief (e.g. a name, description, country, time period, etc.) in all required fields 3. The user associates zero to many supporting documents for the fact/belief 4. The user submits the new fact/belief to the software |
| **Post-conditions:**  A new fact or belief is successfully added to the software’s collection of facts and beliefs |
| **Exceptions**:  The user selects to abort the addition before submitting. The software does not add a fact/belief and the use case ends. |

**CSV File format:**

For batch adding of facts/beliefs, we support csv files. This file should be properly formatted. It is mandatory that the file have following columns in given order: Statement, Region, Metric, Timeperiod, isBelief, Person, Strength, and Note.

Table 4 Batch adding of facts use case description

|  |
| --- |
| **Use Case: Batch Adding of Facts** |
| **ID**: 2 |
| **Brief Description**:  The user adds multiple facts at once by entering them via a batch file |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The batch file used is composed of facts only and is in .csv format |
| **Main Flow:**   1. The user selects the option to add a batch of facts 2. The user selects the batch file containing the facts to be added 3. The software adds all facts in the batch file to the software’s collection of facts and beliefs |
| **Post-conditions:**  All facts in the batch file have successfully been added to the software’s collection of facts and beliefs |
| **Exceptions**:  2.a. The user selects to abort the addition before submitting. The software does not add any facts and the use case ends.  2.b. The batch file is not formatted properly and is rejected by the software. The software does not add any facts and the use case ends. |

Table 5 Adding facts via folder scan use case description

|  |
| --- |
| **Use Case: Adding Facts via Folder Scan** |
| **ID**: 3 |
| **Brief Description**:  The user adds a fact and associates supporting documents with it all at once by selecting a folder |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The folder to be scanned exists on the user’s hard drive |
| **Main Flow:**   1. The user selects the option to add a fact via folder scan 2. The user selects the folder to be scanned 3. The software adds a fact with the same name as the scanned folder and associates all contained documents with that fact |
| **Post-conditions:**  A new fact is successfully added to the software’s collection of facts and beliefs. This new fact is associated with all documents contained in the user’s selected folder. |
| **Exceptions**:  2.a. The folder contains no documents and is rejected by the software. The user either selects another folder or exits the use case having added no new facts. |

Table 6 Search Fact/Belief use case description

|  |
| --- |
| **Use Case**: **Search Fact/Belief** |
| **ID**: 4 |
| **Brief Description:**  Enables user to search for a fact or belief |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The software must be open |
| **Main Flow:**   1. User enters search items 2. User hits “Search” 3. Software looks for the items    1. If the search items are present, the software provides a list of possible facts or beliefs    2. If the search items are not present, the software informs the user that the items are not present |
| **Post-conditions:**  None. |

Table 7 View Fact/Belief use case description

|  |
| --- |
| **Use Case: View Fact/Belief** |
| **ID:** 5 |
| **Brief Description:**  Enables user to view a fact or belief |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The software must be open |
| **Main Flow:**   1. Include (Search) 2. User selects the fact/belief to view. 3. User clicks on ‘View’. |
| **Post-conditions:**  The selected items are displayed to the user. |

Table 8 Delete Fact/Belief use case description

|  |
| --- |
| **Use Case: Delete Fact/Belief** |
| **ID:** 6 |
| **Brief Description:**  Enables user to delete a fact or belief |
| **Primary Actor:**  Admin |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The software must be open |
| **Main Flow:**   1. Include (Search) 2. Admin selects the fact/belief to delete 3. Admin clicks on ‘Delete’ |
| **Post-conditions:**  The selected item and its associated docs are deleted from the database/file. |

Table 9 Edit Fact/Belief use case description

|  |
| --- |
| **Use Case: Edit Fact/Belief** |
| **ID:** 7 |
| **Brief Description:**  Enables user to edit a fact or belief |
| **Primary Actor:**  User |
| **Secondary Actor:**  None |
| **Pre-conditions:**  The software must be open |
| **Main Flow:**   1. Include (Search) 2. User selects the fact/belief to edit. 3. User makes the changes and clicks on save |
| **Post-conditions:**  The changes made to the selected items are stored to the database/file. |

Table 10 Add Fact/Belief meta-data item

|  |
| --- |
| **Use case: Add fact/belief meta-data item** |
| **ID**: 8 |
| **Brief Description**:  Admin adds new dropdown entry for fact/belief meta-data. |
| **Primary Actor**:  User |
| **Secondary Actor**:  None |
| **Pre-conditions**: |
| **Main Flow**:   1. User clicks on ‘Manage Dropdown data’ from the menu. 2. User clicks on ‘Add item’ on the screen. 3. User selects particular dropdown and enters entry into textbox. 4. User clicks on ‘Add entry’. |
| **Post-conditions**:  Newly added entry is available in that dropdown list. |

Table 11 Delete Fact/Belief meta-data item

|  |
| --- |
| **Use case: Delete fact/belief meta-data item** |
| **ID**: 9 |
| **Brief Description**:  User deletes existing dropdown entry for fact/belief meta-data. |
| **Primary Actor**:  User |
| **Secondary Actor**:  None |
| **Pre-conditions**:  User is logged in. |
| **Main Flow**:   1. User clicks on ‘Manage Dropdown data’ from the menu. 2. User clicks on ‘Delete item’ on the screen. 3. User selects particular dropdown. 4. The system will show all the entries for that dropdown. 5. User selects particular entry from the list and clicks on ‘Delete entry’. |
| **Post-conditions**:  Deleted entry is not available in that dropdown list. |

### 3.3 Rationale for Use Case Model

The greatest challenge a software developer faces is sharing the vision of the final product with the customer. All stakeholders in the project - developers, end users, software managers, customer managers - must achieve a common understanding of what the product will be and do, or someone will be surprised when it is delivered. Surprises in software are almost never good news. Therefore, we need ways to accurately capture, interpret, and represent the voice of the customer when specifying the requirements for a software product.

Often the customer will present as "needs" some combination of: the problems he/she has in her work that she expects the system to solve; the solutions he/she has in mind for an expressed or implied problem; the desired attributes of whatever solution ultimately is provided; and the true fundamental needs, that is, the functions the system must let him/her perform. The problem becomes more complex if the systems analyst is dealing with a surrogate customer, such as a marketing representative, who purports to speak for the actual end users of the application. The challenge to the analyst is to distinguish among these four types of input and identify the real functional requirements that will satisfy the real user's real business needs.

Many techniques are used for eliciting user requirements, all of which attempt to include the voice of the customer in the product design process. A typical project might employ a combination of meetings with user representatives and developers, facilitated workshops (for example, joint application design sessions) with analysts and users, individual customer interviews, and user surveys. The use case approach is an especially effective technique for deriving software requirements, analysis models, and test cases.

Each use case describes a scenario in which a user interacts with the system being defined to achieve a specific goal or accomplish a particular task. Use cases are described in terms of the user's work terminology, not computers. By focusing on essential use cases, stripped of implementation constraints or alternatives, the analyst can derive software requirements that will enable the user to achieve his/her objectives in each specific usage scenario.

For each use case, we stated the goal that the user needed to accomplish-one reason someone would use this application. Estimating the anticipated frequency of execution for each use case gave us a preliminary indication of concurrent usage loads, the importance of ease of learning versus ease of use, and capacities of data storage or transaction throughput. We could also identify the relative priority of implementing each use case at this stage. The sequence in which users identify candidate use cases suggests an approximate implementation priority.

By walking through individual use cases in the meetings, we drilled down to the fundamental customer needs the system really needed to satisfy. We also explored many "what if" scenarios to reveal exception and decision situations the system must handle.

### 3.4 Non-functional Requirements

* **Performance:** The application shall provide high performance and we expect the query time to be around 5 seconds.
* **Security and safety:** The application shall provide complete security for the user’s files and data. Also there would be a separate login for admin.
* **Reliability:** The application shall be reliable in the terms that it will function as expected under circumstances like crashes, load balancing for a specified period of time.
* **Usability:** The application shall provide usability by using simple, understandable graphical user interface. In this way, when a new user uses this app, she/he finds the application easy to use. Also, there will be separation of concerns between the user and the admin.
* **Extensibility:** The application shall be implemented in such a way by modularizing the components so that it becomes easy for adding/removing functionalities.

## 4. Architecture

### 4.1 Architectural Style Used

We decided to use MVC architecture style for the application. The MVC architecture style allows us to keep the presentation layer (the view) separate from the methods that receive and handle client requests (the controllers) and the data representations that are returned within the view (the models). We are using MVC predominantly as it provides a separation of concerns, that is, it will keep the code that serves one functional purpose (e.g. handling client requests) separate from the code that serves an entirely different functional purpose (e.g. representing data). In our application this architecture style will highly benefit us as we have multiple views; for instance - facts /beliefs view, batch add view, folder upload view, edit fact/belief view, delete fact/ belief view etc. All these views will communicate via the model with the controller and will thus help us manage the various functionalities by providing a separation of concern.

Another advantage of MVC architecture style we think that benefited us is that the model can have a reference back to the view. This means when data is updated the model can push this data back to possibly multiple views. This proved essential in our application. Imagine a situation where the user makes an edit to a fact (which is a change in one view), this would be first pushed via the controller to the model, which then pushes it back to the display fact/belief view(the other view).

We have implemented the application in parallel by dividing the work amongst ourselves. In our opinion, MVC architecture proved ideal for us as some developers built the java classes which contain the business logic, while the UI developers worked on designing UI screens. Both these activities were carried out simultaneously reducing the interdependency issues and thus helped us build the application with ease. Also any UI update when needed can be made without slowing down the business logic process as business logic rules need very less revisions as compared to the UI.

### 4.2 Architectural Model

The architecture diagram is depicted in **Figure 1XX**. There are mainly 5 subsystems of the Workforce Research Guide desktop application. As shown in the figure 1, the subsystems are (1) Entity Creation System, (2) Entity Management System, (3) Search System, (4) Entity Template Management System and, (5) Data Repository System.

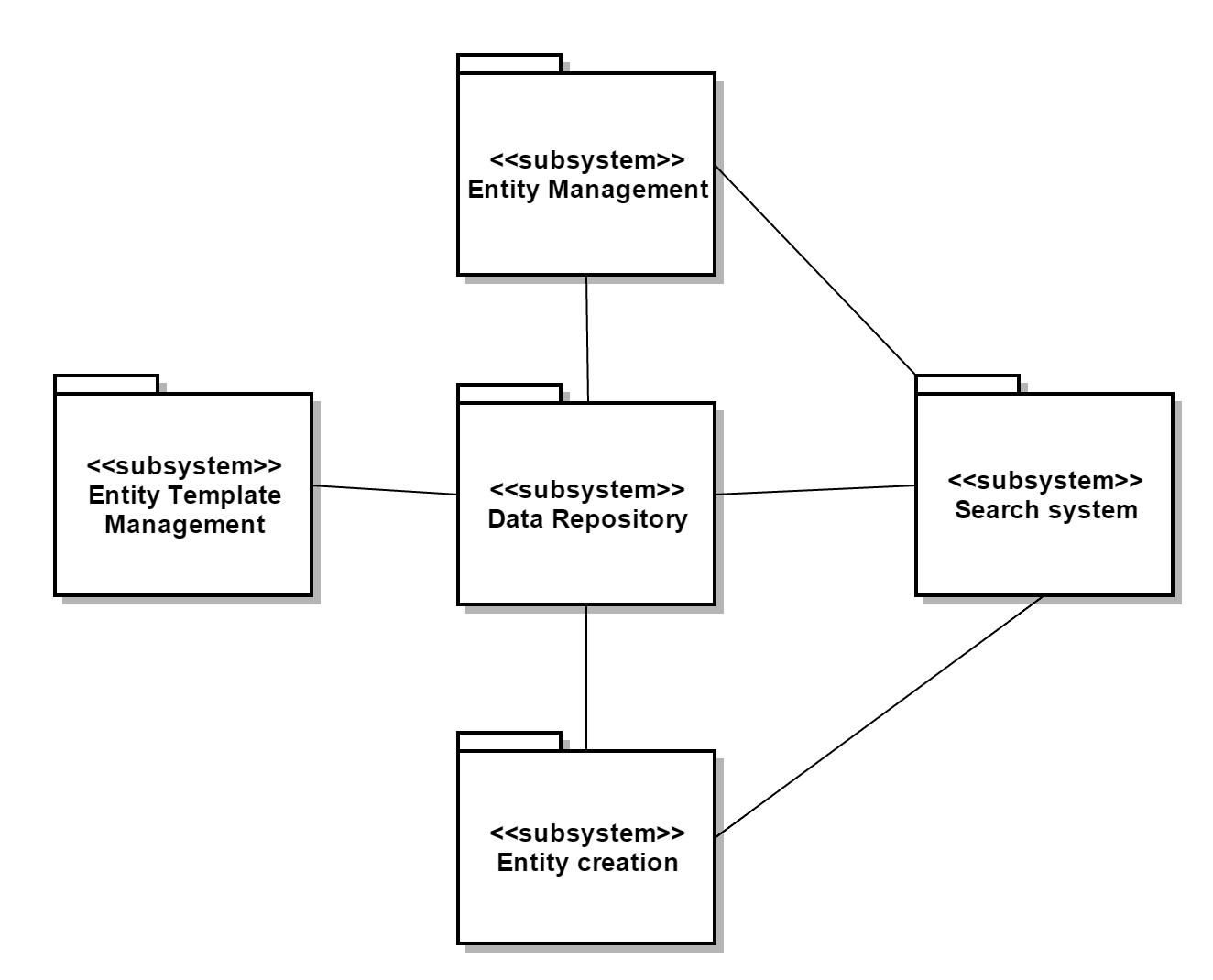


Figure 3 Architecture Diagram

There are total 9 use-cases of the system. They are (1) Add entity, (2) Batch adding of facts, (3) Add fact via folder scan, (4) Search entity, (5) View entity, (6) Delete entity, (7) Edit entity, (8) Add fact metadata and, (9) Delete fact metadata. Entity can be either fact or belief.

Entity creation system supports use cases 1, 2 and 3. Entity management system supports use cases 5, 6 and 7. Search system supports use case 4. Entity template management system supports use case 8 and 9. All these systems interact with the data repository to retrieve the existing information or store new/modified information. Entity management system interacts with search system to retrieve specific set of entities and then user can view, edit or delete one of them.

### 4.3 Technology, Software and Hardware Used

The following technologies are used for the project.

1. Development IDE: Eclipse, NetBeans, IntelliJ
2. Programming Language: Java
3. Documentation: Microsoft Office
4. Database: SQLite
5. UML diagrams: StarUML, Gliffy
6. Document repository: Github
7. Hardware: PC or Laptop
8. Wrapper program Launch4j to create the final .exe file

### 4.4 Rationale for Architectural Style and Model

We have used MVC architecture style. For MVC model, view part sends the input events from UI. Control part decides which view event triggers which functions in the model part. Model part can automatically announce the changes in the model to the view part. Then, the view part updates its view.

The reason to use MVC model is because separation between view and model parts can avoid code changing in one part impacts the code in the other part. UI developer focuses functions in the view part and doesn’t need to worry about the codes in the model part, and similarly for function developer in the model part. This makes development more efficient and application quality better.

Database SQLite will be used to connect model part of MVC, because it is lightweight, fast, and reliable which accords with user’s expectation. The database is not centralized. It is stored on the user’s machine. The data size, CPU, and memory are not large, so SQLite would be sufficient.

## 5. Design

### 5.1 Graphical User Interface Design

#### GUI I. 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.

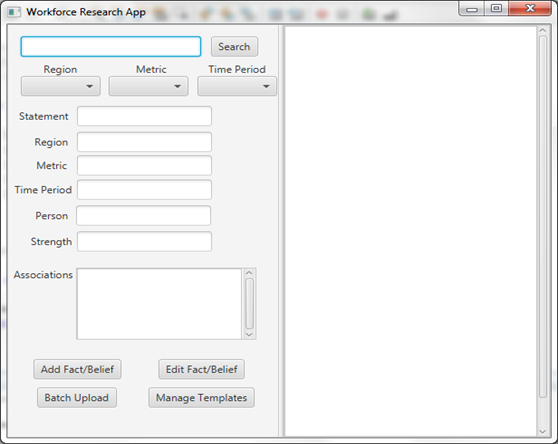


Figure 4 Home Screen - GUI

#### GUI II. 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.

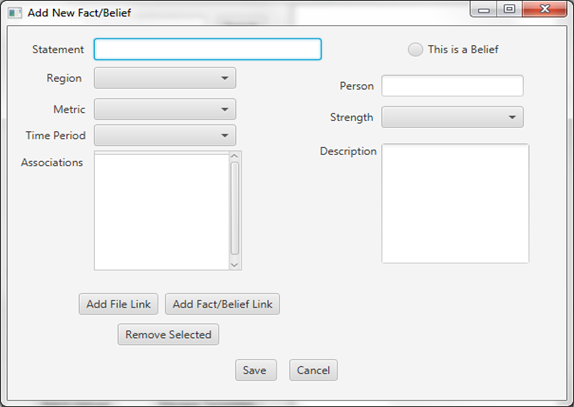


Figure 5 Add Entity - GUI

#### GUI III. 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.

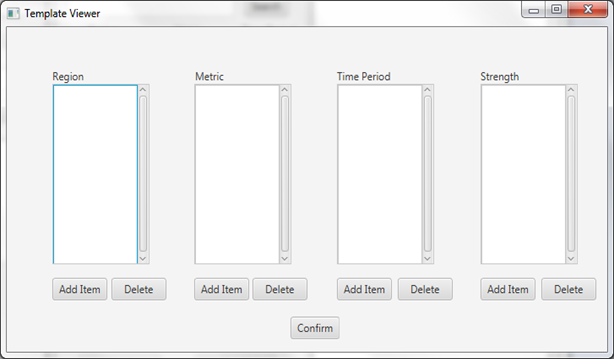


Figure 6 Template Manager - GUI

#### GUI IV. 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.

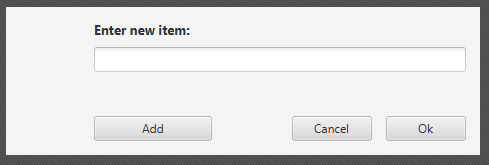


Figure 7 Add Template – GUI

#### GUI V. Edit entity

From the home screen, when user clicks on ‘Edit Fact/Belief’ this screen is displayed. All the fields are editable here. After making changes, user clicks on save button and the updates will be saved into data repository.

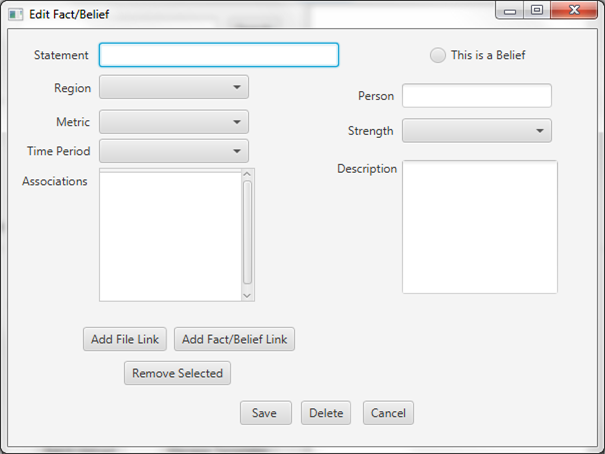


Figure 8 Edit entity - GUI

### 5.2 Static Model – Class Diagrams

AppHandler is the application level controller for the system. All the user requests go 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 (Region, Metric, Timeperiod, and Strength) 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.

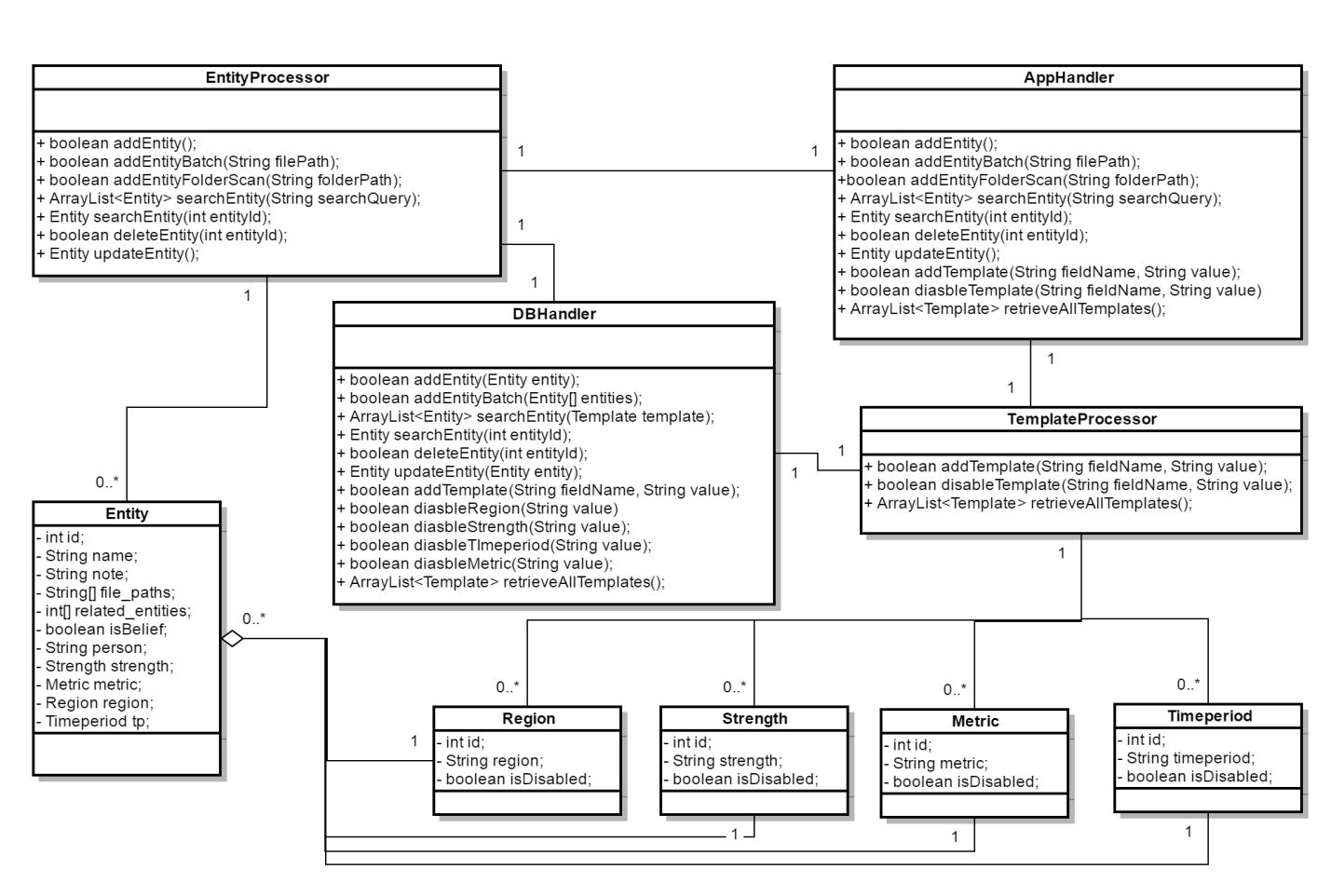


Figure 9 Class Diagram of system

### 5.3 Dynamic Model – Sequence Diagrams

#### SD 1. Add Entity

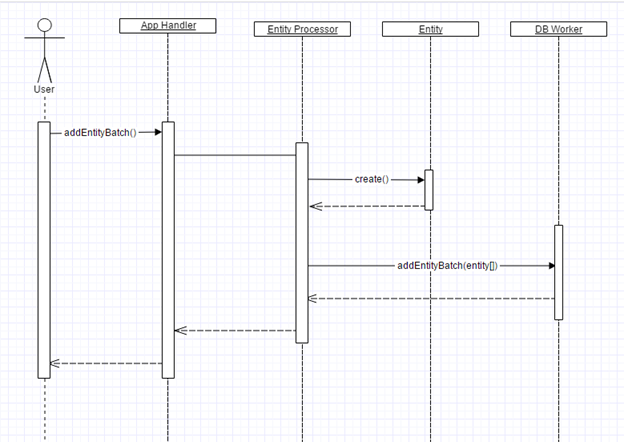


Figure 10 Add Entity - SD

#### SD II. Add Entity - Batch

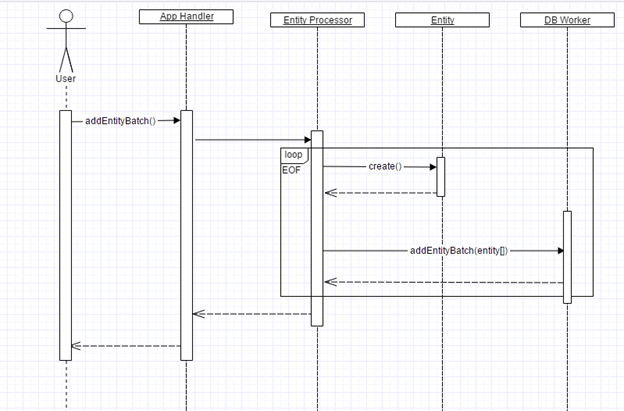


Figure 11 Add Entity - batch - SD

#### SD III. Add Entity – folder scan

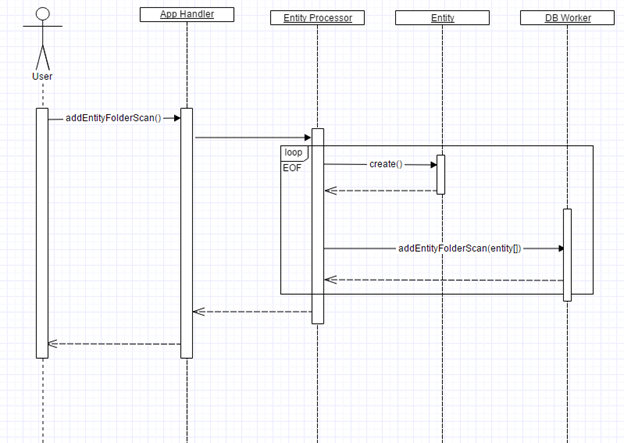


Figure 12 Add Entity - folder scan - SD

#### SD IV. Add Template

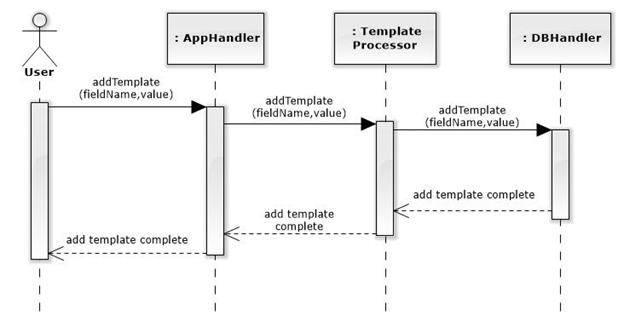


Figure 13 Add Template - SD

#### SD V. Delete Entity

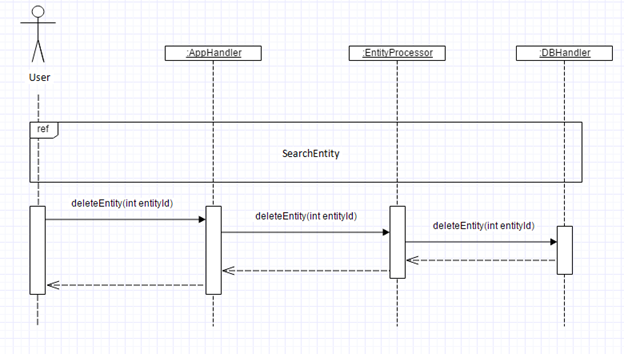


Figure 14 Delete Entity – SD

#### SD VI. Delete Template

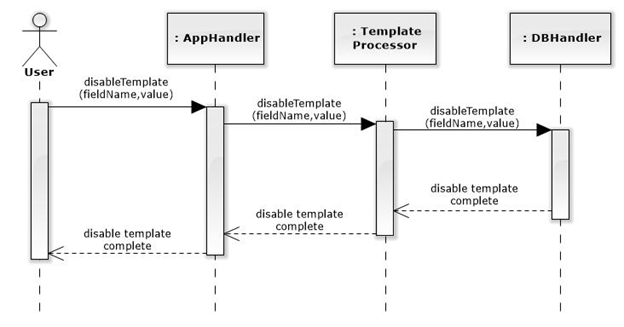


Figure 15 Delete Template - SD

#### SD VII. Edit Entity

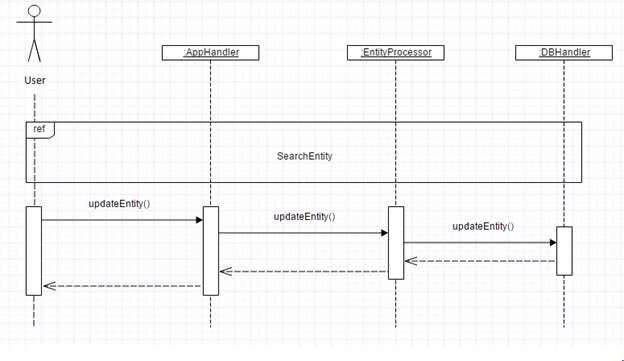


Figure 16 Edit Entity - SD

#### SD VIII. Search Entity

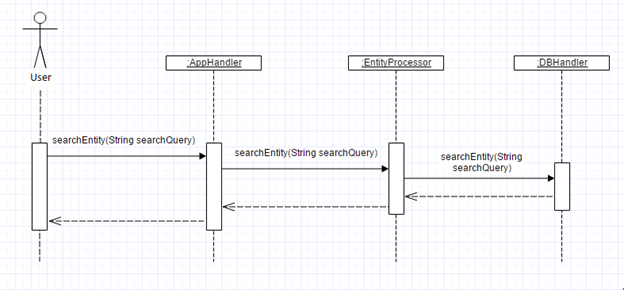


Figure 17 Search Entity - SD

#### SD IX. View Entity

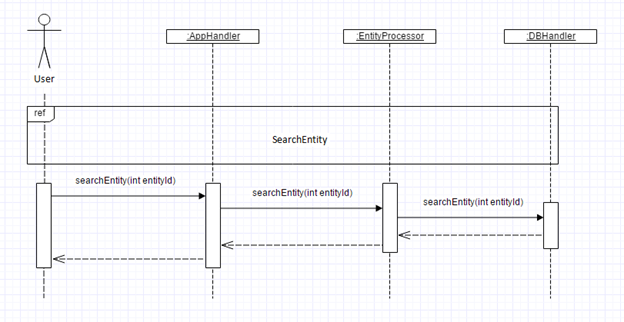


Figure 18 View Entity - SD

### 5.4 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.

### 5.5 Traceability from Requirements to Detailed Design

Table 12 Traceability Matrix - Requirements to Detailed design

|  |  |
| --- | --- |
| **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 |

## 6. Test Plan

### 6.1. Requirements/specifications-based system level test cases

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 1: Add Entity** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a statement, geography, metric, and time period | (Employment Decreased 2%, USA, employment rate, 2014) | All required fields are now filled |
| 3 | User presses the save button |  | Entity has been added to the database and can now be searched for |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 2: Add Entity (statement field not filled)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a geography, metric, and time period but no statement | (<empty string>, USA, employment rate, 2014) | All but one required field are now filled |
| 3 | User presses the save button |  | Program alerts the user that one or more required fields are empty, fact/belief is not added |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 3: Add Entity (geography field not filled)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a statement, metric, and time period but no geography | (Employment Decreased 2%, <no selection>, employment rate, 2014) | All but one required field are now filled |
| 3 | User presses the save button |  | Program alerts the user that one or more required fields are empty, fact/belief is not added |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 4: Add Entity (metric field not filled)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a statement, geography, and time period but no metric | (Employment Decreased 2%, USA, <no selection>, 2014) | All but one required field are now filled |
| 3 | User presses the save button |  | Program alerts the user that one or more required fields are empty, fact/belief is not added |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 5: Add Entity (time period field not filled)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a statement, geography, and metric but no time period | (Employment Decreased 2%, USA, employment rate, <no selection>) | All but one required field are now filled |
| 3 | User presses the save button |  | Program alerts the user that one or more required fields are empty, fact/belief is not added |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 6: Canceling out of Adding an Entity** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Add Fact/Belief" Button |  | Add Entity Window appears |
| 2 | User enters a statement, geography, metric, and time period | (Employment Decreased 2%, USA, employment rate, 2014) | All required fields are now filled |
| 3 | User exits the add entity window |  | Program does not add a fact/belief to the database |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 7: Batch Uploading Facts** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Batch Upload" button |  | File selection window appears |
| 2 | User selects a .csv file | valid.csv | Program has uploaded a fact for each line in the .csv and has displayed the success message for each fact |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 8: Batch Uploading Facts (.csv file not formatted properly)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Batch Upload" button |  | File selection window appears |
| 2 | User selects a .csv file | invalid.csv | Program displays an error message for each incorrectly formatted row in the .csv |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 9: Folder Scan** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Folder Scan" button |  | File selection window appears |
| 2 | User selects a folder | ../Test\_Folder | Program has uploaded a fact for each file found in the specified folder |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 10: Folder Scan (unreadable folder)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Folder Scan" button |  | File selection window appears |
| 2 | User selects a folder | ../Unreadable\_Folder | Program alerts the user that the folder is unreadable, no facts are added to the database |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 11: Folder Scan (empty folder)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | User presses "Folder Scan" button |  | File selection window appears |
| 2 | User selects a folder | ../Empty\_Folder | Program alerts the user that the folder is unreadable, no facts are added to the database |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 12: Search Entity (Entities exist/match)** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Type in search keywords in the text box | statement: Employment Increased | The matched entities are listed in the right panel. |
| 2 | Select geography, metric, and time period from three drop down lists | geography: USA  metric: 1.12%  time period: 2015 | The matched entities are listed in the right panel. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 13: Search Entity (Entities don't exist/match)** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Type in search keywords in the text box | statement: relocate | Empty list in the right panel. |
| 2 | Select geography, metric, and time period from three drop down lists | geography: UK  metric: 1%  time period: 2014 | Empty list in the right panel. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 14: Edit Entity** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The values of the selected entity are automatically filled in the blanks in the window. |
| 3 | Edit Entity by filling text boxes, selecting from drop down lists, and selecting the associated entities. | statement: environmental issue  geography: USA  metric: 15%  time period: 2013  Person: Tom  is Belief: yes | The values of the edited entity are changed in the fields in the window. |
| 4 | Press Update Entity button |  | (1) The Entity is updated into the database.  (2) Update successful message is displayed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 15: Cancel Out of Edit Entity** | | | |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The values of the selected entity are automatically filled in the blanks in the window. |
| 3 | Edit Entity by filling text boxes, selecting from drop down lists, and selecting the associated entities. | statement: environmental issue  geography: USA  metric: 15%  time period: 2013  Person: Tom  is Belief: yes | The edited entity changes values in the attribute fields in the window. |
| 4 | Press Cancel Button |  | The Entity fields in the window become empty. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 16: Edit Entity (without required fields)** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The selected entity automatically fills the attribute fields in the window. |
| 3 | Edit Entity by filling text boxes, selecting from drop down lists, and selecting the associated entities. | statement:  geography:  metric:  time period: | Error message: "Missing statement, geography, metric, and time period. They are required." |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 17: Delete Entity** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The selected entity automatically fills the attribute fields in the window. |
| 3 | Press Delete Button |  | Pop up a window message: "Are you sure to delete this XXX Fact/Belief?" |
| 4 | Press Yes |  | (1) The Entity fields in the window become empty.  (2) The Entity disappears in the right panel list. |
| **TC 18: Cancel Out of Delete Entity** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The selected entity automatically fills the attribute fields in the window. |
| 3 | Press Delete Button |  | Pop up a window message: "Are you sure to delete this XXX Fact/Belief?" |
| 4 | Press No |  | Cancel delete |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 19: View Fact/Belief** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Search Entity | statement: environment  geography: Canada  metric: 12%  time period: 2017 | The matched entities are listed in the right panel. |
| 2 | Select Entity |  | The selected entity automatically fills the attribute fields in the window. |
| **TC 20: Add Region** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the region to be added | Region: Dallas | The entered region is added to the database and displayed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 21 : Cancel out of adding Region** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the region to be added | Region: Dallas |  |
| 2 | Press Cancel button |  | The entered data will not be added. The data will be cleared. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 22: Add Metric** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the metric to be added | Metric: Employee quit rate | The entered metric is added to the database and displayed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 23: Cancel out of adding Metric** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the metric to be added | Metric: Employee quit rate |  |
| 2 | Press Cancel button |  | The entered data will not be added. The data will be cleared. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 24: Add Timeperiod** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the Timeperiod to be added | Timeperiod: 2015 | The entered timeperiod is added to the database and displayed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 25: Cancel out of adding Timeperiod** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the Timeperiod to be added | Timeperiod: 2015 |  |
| 2 | Press Cancel button |  | The entered data will not be added. The data will be cleared. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 26: Add Strength** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the strength to be added | Strength: high | The entered strength is added to the database and displayed. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 27: Cancel out of adding Strength** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Enter the strength to be added | Strength: high |  |
| 2 | Press Cancel button |  | The entered data will not be added. The data will be cleared. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 28: Disable Region** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the region to be disabled | Region: Dallas | The selected data is disabled and will not be displayed in the future selections. |
| **TC 29: Cancel out of disabling Region** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the region to be disabled | Region: Dallas |  |
| 2 | Press Cancel button |  | The data will not be disabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 30: Disable Metric** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the metric to be disabled | Metric: Employee quit rate | The selected data is disabled and will not be displayed in the future selections. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 31: Cancel out of disabling Metric** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the metric to be disabled | Metric: Employee quit rate |  |
| 2 | Press Cancel button |  | The data will not be disabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 32: Disable Timeperiod** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the timeperiod to be disabled | Timeperiod: 2016 | The selected data is disabled and will not be displayed in the future selections. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 33: Cancel out of disabling timeperiod** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the timeperiod to be disabled | Timeperiod: 2016 |  |
| 2 | Press Cancel button |  | The data will not be disabled. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 34: Disable Strength** | |  |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the strength to be disabled | Strength: high | The selected data is disabled and will not be displayed in the future selections. |

|  |  |  |  |
| --- | --- | --- | --- |
| **TC 35: Cancel out of disabling Strength** | | |  |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** |
| 1 | Select the strength to be disabled | Strength: high |  |
| 2 | Press Cancel button |  | The data will not be disabled. |

### 

### 6.2. Traceability of test cases to use cases

Table 13 Traceability matrix - Tests to Use case

|  |  |
| --- | --- |
| **Testing Plan – Test Case** | **Requirements – Use Case** |
| TC 1: Add Entity | UC1: Add Entity |
| TC 2: Add Entity (statement field not filled) | UC1: Add Entity |
| TC 3: Add Entity (geography field not filled) | UC1: Add Entity |
| TC 4: Add Entity (metric field not filled) | UC1: Add Entity |
| TC 5: Add Entity (time period field not filled) | UC1: Add Entity |
| TC 6: Canceling out of Adding an Entity | UC1: Add Entity |
| TC 7: Batch Uploading Facts | UC2: Batch Adding of Facts |
| TC 8: Batch Uploading Facts (.csv file not formatted properly) | UC2: Batch Adding of Facts |
| TC 9: Folder Scan | UC3: Adding Facts via Folder Scan |
| TC 10: Folder Scan (unreadable folder) | UC3: Adding Facts via Folder Scan |
| TC 11: Folder Scan (empty folder) | UC3: Adding Facts via Folder Scan |
| TC 12: Search Entity (Entities exist/match) | UC4: Search Entity |
| TC 13: Search Entity (Entities don't exist/match) | UC4: Search Entity |
| TC 14: Edit Entity | UC7: Edit Entity |
| TC 15: Cancel Out of Edit Entity | UC7: Edit Entity |
| TC 16: Edit Entity (without required fields) | UC7: Edit Entity |
| TC 17: Delete Entity | UC6: Delete Entity |
| TC 18: Cancel Out of Delete Entity | UC6: Delete Entity |
| TC 19: View Fact/Belief | UC5: View Entity |
| TC 20: Add Region | UC8: Add Metadata Item |
| TC 21 : Cancel out of adding Region | UC8: Add Metadata Item |
| TC 22: Add Metric | UC8: Add Metadata Item |
| TC 23: Cancel out of adding Metric | UC8: Add Metadata Item |
| TC 24: Add Timeperiod | UC8: Add Metadata Item |
| TC 25: Cancel out of adding Timeperiod | UC8: Add Metadata Item |
| TC 26: Add Strength | UC8: Add Metadata Item |
| TC 27: Cancel out of adding Strength | UC8: Add Metadata Item |
| TC 28: Disable Region | UC9: Delete Metadata Item |
| TC 29: Cancel out of disabling Region | UC9: Delete Metadata Item |
| TC 30: Disable Metric | UC9: Delete Metadata Item |
| TC 31: Cancel out of disabling Metric | UC9: Delete Metadata Item |
| TC 32: Disable Timeperiod | UC9: Delete Metadata Item |
| TC 33: Cancel out of disabling timeperiod | UC9: Delete Metadata Item |
| TC 34: Disable Strength | UC9: Delete Metadata Item |
| TC 35: Cancel out of disabling Strength | UC9: Delete Metadata Item |

### 6.3. Techniques used for test generation

We followed the two standard techniques used for Test Generation viz. Static Testing and Dynamic testing as shown in the below diagram.

**Static Testing** – By using this technique we tested the software without executing the code. We followed two ways for the same as listed below:

1. Review - Typically to find and eliminate errors or ambiguities in documents such as requirements, design, test cases, etc.
2. Static analysis - The code written was analyzed (by tools) for structural defects that may lead to defects. Following kinds of defects were found by the tools during static analysis - Syntax violations, Variables that are declared but never used, Unreachable code (or) Dead Code, good code practice violations.

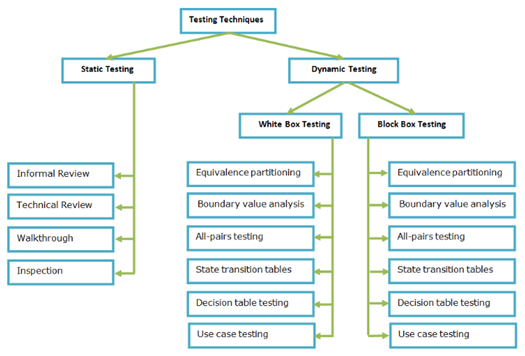


Figure 19 Differences between Black Box and White Box Testing

Dynamic Testing is a kind of software testing technique using which the dynamic behavior of the code is analyzed. These can be classified broadly as Unit Testing, Integration Testing, System Testing, and Acceptance Testing. These tests use the techniques like Boundary Value analysis, Equivalence partitioning, use case testing etc.

### 6.4. Assessment of the goodness of your test suite

The testing techniques our team used are white-box testing, black testing, and acceptance test. In white-box testing, the unit testing is used to test each unit of the codes. Before further integrating different units, it is important to test each unit to make sure there is no error or failure in it. Then, after further integrating, if there appears error or failure, we have more confidence that it might be the interfaces problem, not inside each unit. We use JUnit in Eclipse for unit testing. In black-box testing, we built up use cases with good scenarios and bad scenarios and followed the steps in these use cases to make sure if the outcomes match our expectation. In acceptance test, the sponsor, Jeff in Alliance Data, is satisfied with and accept the application our team developed.

The goodness of the test suite is evaluated by the test coverage rate. That is the percentage of the code covered in the whole codes after executing and operating the application. The higher the coverage rate is; the better the test suite is. We use Clover in Eclipse for getting the test coverage rate. There are 5 ways to measure the test coverage rate:

**(1) Test Coverage by Features**: Based on the specification, each feature, start-up, and shut-down are tested for test coverage rate.

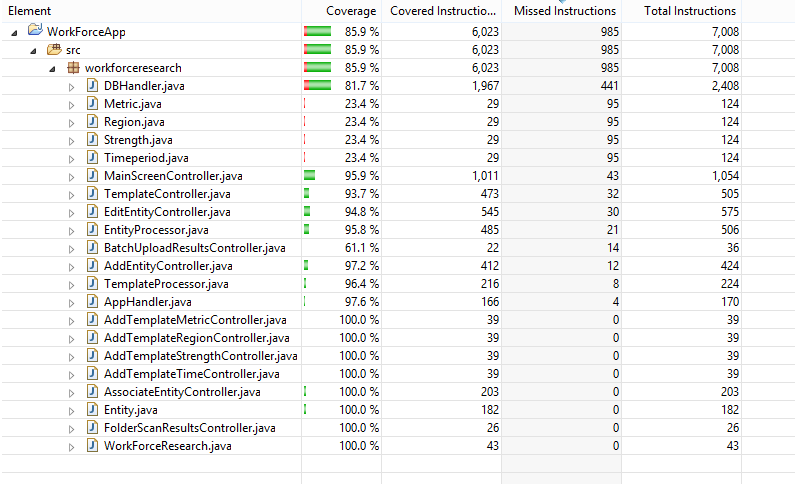
**(2) Test Coverage by GUI icon**: The user interfaces like buttons, pull-downs, text-box, list, etc. are tested for test coverage rate.

**(3) Test Coverage by Instrumentation**: Use a code instrumentation tool to instrument a build, and then test that build using the system tests already prepared. The tool output should indicate how much coverage in the codes the system had.

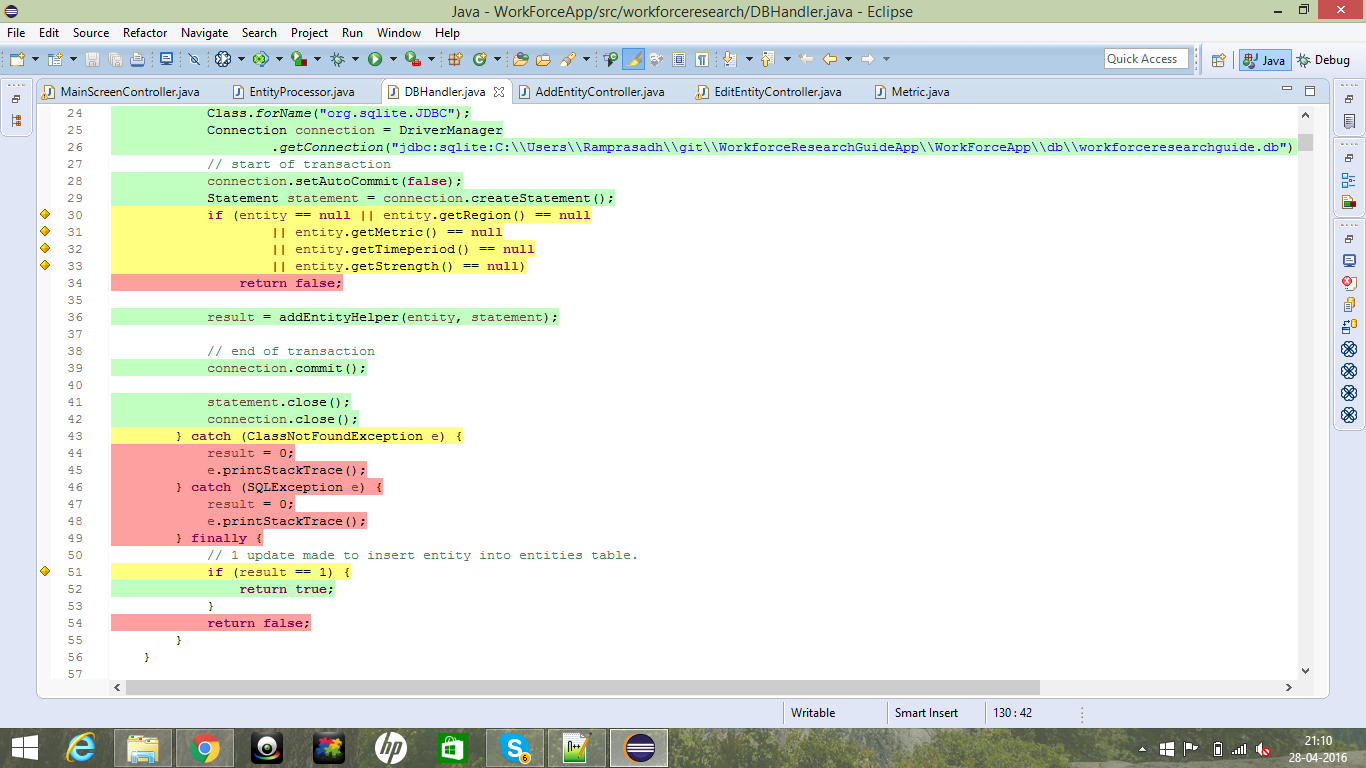
**(4) Test Coverage by Structure**: Testing should include different structures, Statement Coverage, Decision (Branch) Coverage, Condition Coverage, All-DU paths Coverage, and Linear Code Sequence and Jump (LCSAJ).

**(5) Test Coverage by scenario**: Users may have a number of goals which they want to achieve and a number of cases which they want to avoid. In doing so, we set up subtle feature interactions. Uses cases form the baseline of such an approach.

**Coverage Report:** The coverage report is displayed in following image for our test suite.



**Example for one of the file:**

****

## Acknowledgements

We would like to thank Prof. Wong, University of Texas at Dallas, and Alliance Data for giving this opportunity to work such project. We would also like to thank Jeff Buchmiller, our mentor from Alliance Data, for his guidance over the course of the project as well as the timely and constructive feedback. We received his guidance at all stages of the project.

## References

[1] B. Boehm and WJ Hansen, “Spiral Development: Experience, Principles, and Refinements: Spiral Development Workshop February 9, 2000”, July 2000, <<http://www.sei.cmu.edu/reports/00sr008.pdf>>

[2] S. Ambler, "UML 2 Use Case Diagrams: An Agile Introduction", *Agilemodeling.com*, 2016. [Online]. Available: <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>. [Accessed: 23- Apr- 2016].

[3] "Model–view–controller", *Wikipedia*, 2016. [Online]. Available: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>. [Accessed: 23- Apr- 2016].

[4] Bowden, *Creating Class Diagram*. <http://homepages.uel.ac.uk/D.Bowden/Creating_Class_Diagram/Creating_Class_Diagram.viewlet/Creating_Class_Diagram.swf> [Accessed: 23-Apr-2016].

[5] Bowden, *Creating Sequence Diagram*. <http://homepages.uel.ac.uk/D.Bowden/Creating_Sequence_Diagrams/Creating_Sequence_Diagrams.viewlet/Creating_Sequence_Diagrams.swf>

[6] *Black-Box vs. White-Box Testing*, 1st ed. Redstone Software, 2008. <http://www.cs.unh.edu/~it666/reading_list/Defense/blackbox_vs_whitebox_testing.pdf> [Acessed 23-Apr-2016].

[7] launch4j, <http://launch4j.sourceforge.net/>