**Architecture**

Workforce

Research

Guide

**Team: Group2**

Shin Yi Lin

Fenil Shah

Ramprasadh Srivathsa

Harshal Pawar

Mark Dwight

SE 6387.001

**ABSTRACT**

This document serves to outline the architecture used in the design and development of the Workforce Research application. We describe the architectural styles to be utilized and how they complement the functionality of the program. We also describe the architectural model, giving a view of the subsystems to be implemented and how they will interact. This document also includes information regarding the software and hardware necessary to implement the application. Finally we rationalize our choices regarding architectural style and model.

**TABLE OF CONTENTS**

1. Introduction

1.1 Introduction to the document.....……………….……………………………………………………………4

1.2 Purpose and scope of the document..……………………………………………………………………..4

2. Architectural Style Used.……………………………………………………..…..…………….……………………………..4

3. Architectural Model……………………………………………………………………………………………………………….5

4. Technology, Software and Hardware Used.…………………………………………………………………………………6

5. Rationale for your Architectural Style and Model…………………………..………………………………………7

6. Evidence that the document has been placed under configuration management….………..…….8

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

**LIST OF FIGURES**

Figure 1: Architecture Diagram………..……….……………………………….……………….……………………………..5

Figure 2: Application server database server interaction...……………………………….……………..………..6

**1. INTRODUCTION**

In order to help better define the design of the application, the software architecture of the system needs to be determined fairly early on in development, as it defines the nature of the interrelated components that make up the system and allow it to perform the functions necessary to fulfill its requirements. In this document, we outline the architecture to be used in the Workforce Research application, first by detailing the architectural styles to be used and then by detailing the architectural model, using both to give a broad view of the various subsystems and how they interact to realize the various features of the program. We also describe the necessary software and hardware required to implement the system, as well as provide the rationale for the architectural decisions made.

**2. ARCHITECTURE STYLE USED**

We have decided to use MVC architecture style for the application. The MVC architecture style will allows us to keep the HTML markup in applications 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.

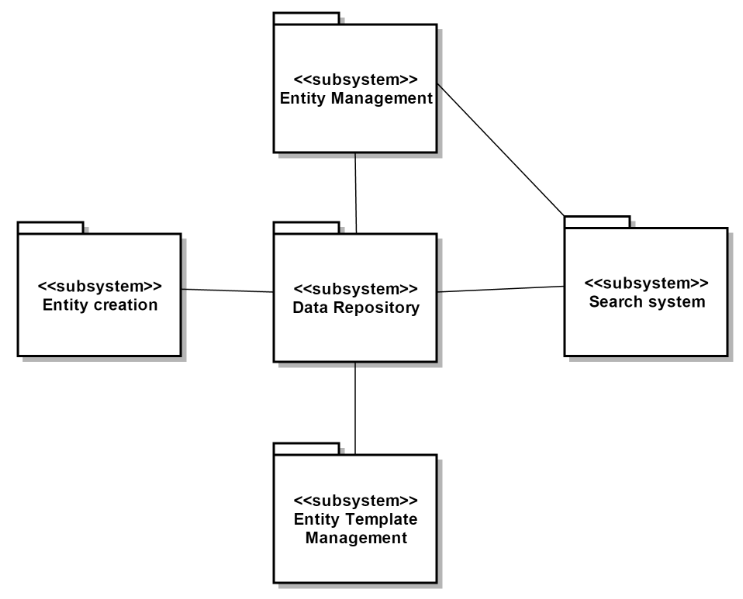
Another advantage of MVC architecture style we think that might benefit us is that the model can have a reference back to the view. This means when data is updated the model can push this data back to possibly multiple views. This would prove essential in our application. Imagine a situation where the 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 planned to implement the application in parallel by dividing the work amongst ourselves. We think that MVC architecture would be ideal for us as business logic developers can build the java classes which will contain the main functionality logic, while the UI developers can involve in designing UI screens. Both these activities can be carried out simultaneously reducing the interdependency issues and thus might help us built the application with sufficient time to test. 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.

Considering all these initial implementation plans we think that MVC architecture will be best suited for our application.

**3. ARCHITECTURAL MODEL**

The architecture diagram is depicted in Figure 1. 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 1 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.

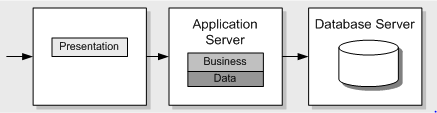
Entity creation system supports use cases 1, 2 and 3. Entity management system supports use cases 5, 6 and 7. Search system supports use case 4. Entity template management system supports use case 8 and 9. All these systems interact with the data repository to retrieve the existing information or store 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. TECHNOLOGY, SOFTWARE, AND HARDWARE USED**

The following technologies will be used for the project.

1. Development IDE: Eclipse, IntelliJ
2. Programming Language: JAVA
3. Documentation: Microsoft Office
4. Database: SQLITE
5. UML diagrams: StarUML
6. Document repository: Github
7. Hardware: PC or Laptop

**Connection between application server and database server:**



**Figure 2 Application server database server interaction**

Application server provides middleware services for easy data access. The database server performs tasks such as data analysis, storage, data manipulation, archiving, and other non-user specific tasks. The database server is the only thing that connects the database to the application. The user interface of the application interacts with the application server which requests or sends data to the database server and gets it response. The database server then proceeds to make the changes requested by the application server to the database.

**5. RATIONALE FOR YOUR ARCHITECTURAL STYLE AND MODEL**

The architectural style we use combines MVC (Model-View-Control) model. 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 announces 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 small, fast, and reliable which accords with user’s expectation. There is only one user, and the data files are all stored in only this user’s machine. The data size, CPU, and memory are not large, so SQLite would be sufficient.

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

**MANAGEMENT**

GitHub Repository:

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

**7. REFERENCES**

[1] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller