|  |
| --- |
| UAH Fit Vault Software Design Specification |
| CPE 656/658 Software Studio |
| Timothy R. Wilkins  Whit J. Sisulak  Glen L. Riden  James J. Duggan IV |

11/14/2015

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Revision Date | Description of Change | Author |
| 0.1 | 10/12/15 | Initial Draft | J. Duggan  W. Sisulak |
| 0.2 | 10/25/15 | Updated scope and architecture. Changed title and file name. | J. Duggan  W. Sisulak |
| 0.3 | 11/07/15 | Added application framework section. Created detail design section. | W. Sisulak |
| 0.4 | 11/13/15 | Update scope and architecture overview. | J. Duggan |
| 0.5 | 11/14/15 | Added detailed design for SelectDataController and SelectActivityController. Added detailed design for Activity Model and ActivityType enumeration. | W. Sisulak |

# Table of Contents

[Revision History i](#_Toc435281011)

[Table of Contents ii](#_Toc435281012)

[1 Introduction 1](#_Toc435281013)

[1.1 Purpose 1](#_Toc435281014)

[1.2 Scope 1](#_Toc435281015)

[1.3 Definitions, Acronyms, and Abbreviations 2](#_Toc435281016)

[1.4 References 2](#_Toc435281017)

[1.5 Overview 2](#_Toc435281018)

[1.6 Application Framework 2](#_Toc435281019)

[2 System Architecture Description 2](#_Toc435281020)

[2.1 Overview of Components 2](#_Toc435281021)

[3 Detailed Description of Components 4](#_Toc435281022)

[3.1 Presentation Layer Overview 4](#_Toc435281023)

[3.2 Business Logic Layer Overview 6](#_Toc435281024)

[3.3 Data Access Layer Overview 7](#_Toc435281025)

Software Design Specification

# Introduction

## Purpose

The purpose of this document is to provide a detailed design of the UAH Fit Vault software projects. This document should be used as a reference for the software system architecture and detailed design descriptions of the system components. The intended audience for this document includes system developers, testers, customers, and any other stakeholders.

## Scope

The UAH Fit Vault software package will be a web application that will accept medical data from users and display the data in a meaningful way. There are two major components to this software. The first is the data collection tool that is used by the users to upload their medical data that is recorded by one of the supported wearable medical devices. There are three different medical devices supported for this project that record various types of data. The data provided by these devices consists of different file formats, and the data is different from device to device. The software will have to determine the contents of each file and how to process them. The software needs to able to take in files that a user has downloaded from their medical devices, process those files, and store the data in a database. The software should have the ability to process multiple files at a time as well as individual files and allow for an activity to be assigned to them by date and time.

The other major component of the web application is the data analysis tools used to analyze the data that is captured from the data collection tool mentioned above. The software needs to perform data analysis over different intervals of time such as one week, one month, etc. There will need to be some way to manage user access to the various medical data that has been inserted into the database that this software will access. Below are some proposed data analysis ideas that can be incorporated into the project.

* Simple Moving Average
* Data correlation discovery between the multiple devices.
* Simply display data that was uploaded to the customer in a graphical format.
* Calculate the user’s activity.

The data analysis possibilities will likely not fully be realized until the project team understands the different types of data that are available. Also, there will need to be collaboration with the customer for additions or changes to the data measurements provided by this software. The web application will have to have different levels of user access which will be defined later in this document.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| SDD | Software Design Document / Software Design Specification |
|  |  |
|  |  |
|  |  |

## References

IEEE Std 1016-1998, IEEE Standard for Software Design Specification

## Overview

The remainder of this design specification document addresses the software system architecture, detailed design information for the various system components, and the database schema design. Each major section will be broken into two pieces each detailing the design criteria for the two pieces of software the make the UAH Fit Vault.

## Application Framework

The application is designed to work with the Microsoft ASP.NET MVC web application framework. This framework uses the Microsoft .NET Framework and Common Language Runtime languages (specifically C#)

Using ASP.NET MVC helps guide the design process. The framework takes ideas from the traditional design patterns of Model-View-Controllers and bootstraps a web application allowing for simple design and allows us to leverage a lot of hard work done by people with far more expertise in web application framework development.

# System Architecture Description

## Overview of Components



**Figure 2.1**: Architectural Overview

The UAH Fit Vault application is a Web application that is comprised of and encompasses the presentation, business logic, and data access layers. The system interacts with the end user via a web interface and with an external SQL database hosted on a remote server. It is assumed that the user is authenticated to use the application.

* The presentation layer consists of a web interface for data processing and reporting. The various views and controllers used in this system will exist in the presentation layer. The views function as the user interface that gives the user a way to select the data files they wish to upload as well as a means for assigning activity information to the data. Other views will be used for account management and for displaying reports to the different users. The controllers process the data that is submitted to the system through the views from the user. The controllers are also responsible for determining the data that is used by the views to display the request information to the user.
* The business logic layer will consist of all the back end code comprised of classes to handle the parsing and processing of files as well as provide the logic to render metrics and reports. Business logic functions are those that are used to edit the data elements in the model classes in the system.
* The data access layer is structured using a singular repository object that provides a uniform means for persisting and retrieving data models from the database. This is intentionally done in order to abstract away other objects ability to communicate directly with the data source. The repository contains the necessary create, read, update, and delete functions necessary to change objects and processes requests from service objects. The data access layer will consist of all the back end code needed to interact with an external SQL database utilizing the data access library of choice. This system will use Entity Framework to manage the data access layer.

# Detailed Description of Components

## Presentation Layer Overview

### Controllers

#### SelectDataController

This class is responsible for selecting the data files for processing. This class provides the backend for the **SelectDataView**. See Figure 1 sequence diagram for this controller.

**<<Task>>**

**Attributes:**

* **Private Collection<int, String>** DataFiles
  + Dictionary containing the paths to the data files and a unique id.
* **Private String** DirectoryPath
  + Path to the directory containing the data files to be processed.

**Operations:**

* Sets and gets for all attributes.
* **Public ActionResult** btnFolderBrowseClick()
  + When the folder browser button is clicked a folder browser dialog object is created, a directory is selected by the user, and the DirectoryPath is set.
  + The results are returned to the **SelectDataView.**
* **Public ActionResult** DragAndDropFiles()
  + When the desired files are selected and dropped into the GUI window.
  + The results are returned to the **SelectDataView.**
* **Public ActionResult** btnClearClick()
  + When the clear button is clicked any stored data and file path(s) will be removed.
    - DirectoryPath variable reset
    - DataFiles collection emptied
    - The results are returned to the **SelectDataView.**
* **Private ActionResult** btnNextClick()
  + When the next button is clicked the file locations of the files in the DirectoryPath are added to the DataFiles collection.
  + The DataFiles collection is passed to the **SelectActivityController.**

#### SelectActivityController

This class is responsible for assigning an activity to a date and time portion of a data file. This class provides the backend for the **SelectActivityView**. See Figure 1 sequence diagram for this controller.

**<<Task>>**

**Attributes:**

* **Private Collection<Activity>** Activities
  + Activities is a collection used to hold all **Activity** objects.

**Operations:**

* Sets and gets for all attributes.
* **Public ActionResult** SelectFile()
  + A data file is selected (id).
  + The results are returned to the **SelectActivityView**
* **Public ActionResult** SelectActivity()
  + An activity is selected.
  + The results are returned to the **SelectActivityView**
* **Public ActionResult** AddDateRange()
  + A date and time range is selected.
  + The results are returned to the **SelectActivityView**
* **Public ActionResult** AddActivity()
  + An activity object is created and assigned the values from the selections
    - File id
    - Activity
    - Date time range
  + The results are returned to the **SelectActivityView**
* **Public ActionResult** ProcessData()
  + Iterates through the data files.
    - Calls the Validate method to validate each data file
      * If a file is invalid it is not process and any associated activates are dropped.
      * If a file is valid the UploadDataFile method is called and passed the data file object
        + Any activities associated with the data file is uploaded via the UploadDataActivity method.
  + When processing is complete the results will be returned to the **SelectActivityView**
* **Public ActionResult** Validate(**File file**)
  + Returns if the file is valid or invalid.
* **Public ActionResult** UploadDataFile(**File file**)
  + This method uploads the data file into the database.

* **Public ActionResult** UploadActivity(**Activity activity**)
  + This method uploads the activity into the database.

## Business Logic Layer Overview

### Entities

#### Activity

This class contains data associated with an activity tied to a date and time portion of a data file. The **Activity** class is used by the **ActivitySelectionController**.

**<<Entity>>**

**Attributes:**

* **Private int** fileId
  + The id of the file to be assigned an activity.
* **Private DateTime** TimeStamp
  + The date and time of the activity.
* **Private ActivityEnum** DataActivity
  + The activity of the data.

**Operations:**

* Sets and gets for all attributes.

### Enumerations

#### ActivityEnum

This is the enumeration that contains user roles within the system. It is an **Enumeration**.

**<<Enumeration>>**

**Attributes:**

* Running
* Walking
* Sleeping
* Sitting
* Standing

## Data Access Layer Overview