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

11/15/2015

# 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 |
| 0.6 | 11/15/15 | Added several new enumerations. Added detailed design for all the various system user objects. Added experiment object design. Created structure and relationship section with description of data ingestion feature. | J. Duggan |

# Table of Contents

[Revision History i](#_Toc435391019)

[Table of Contents ii](#_Toc435391020)

[1 Introduction 1](#_Toc435391021)

[1.1 Purpose 1](#_Toc435391022)

[1.1 Scope 1](#_Toc435391023)

[1.2 Definitions, Acronyms, and Abbreviations 2](#_Toc435391024)

[1.3 References 2](#_Toc435391025)

[1.4 Overview 2](#_Toc435391026)

[1.5 Application Framework 2](#_Toc435391027)

[2 System Architecture Description 2](#_Toc435391028)

[2.1 Overview of Components 2](#_Toc435391029)

[2.2 Structure and Relationships 4](#_Toc435391030)

[2.2.1 Data Ingestion 4](#_Toc435391031)

[3 Detailed Description of Components 5](#_Toc435391032)

[3.1 Presentation Layer Overview 5](#_Toc435391033)

[3.1.1 Controllers 5](#_Toc435391034)

[3.2 Business Logic Layer Overview 7](#_Toc435391035)

[3.2.1 Entities 7](#_Toc435391036)

[3.2.2 Enumerations 12](#_Toc435391037)

[3.3 Data Access Layer Overview 15](#_Toc435391038)

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

Definitions for New Race and Ethnicity Categories (https://nces.ed.gov/ipeds/reic/definitions.asp)

[UAH Fit Vault Sofware Requirements.docx](UAH%20Fit%20Vault%20Sofware%20Requirements.docx)

## 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.

## Structure and Relationships

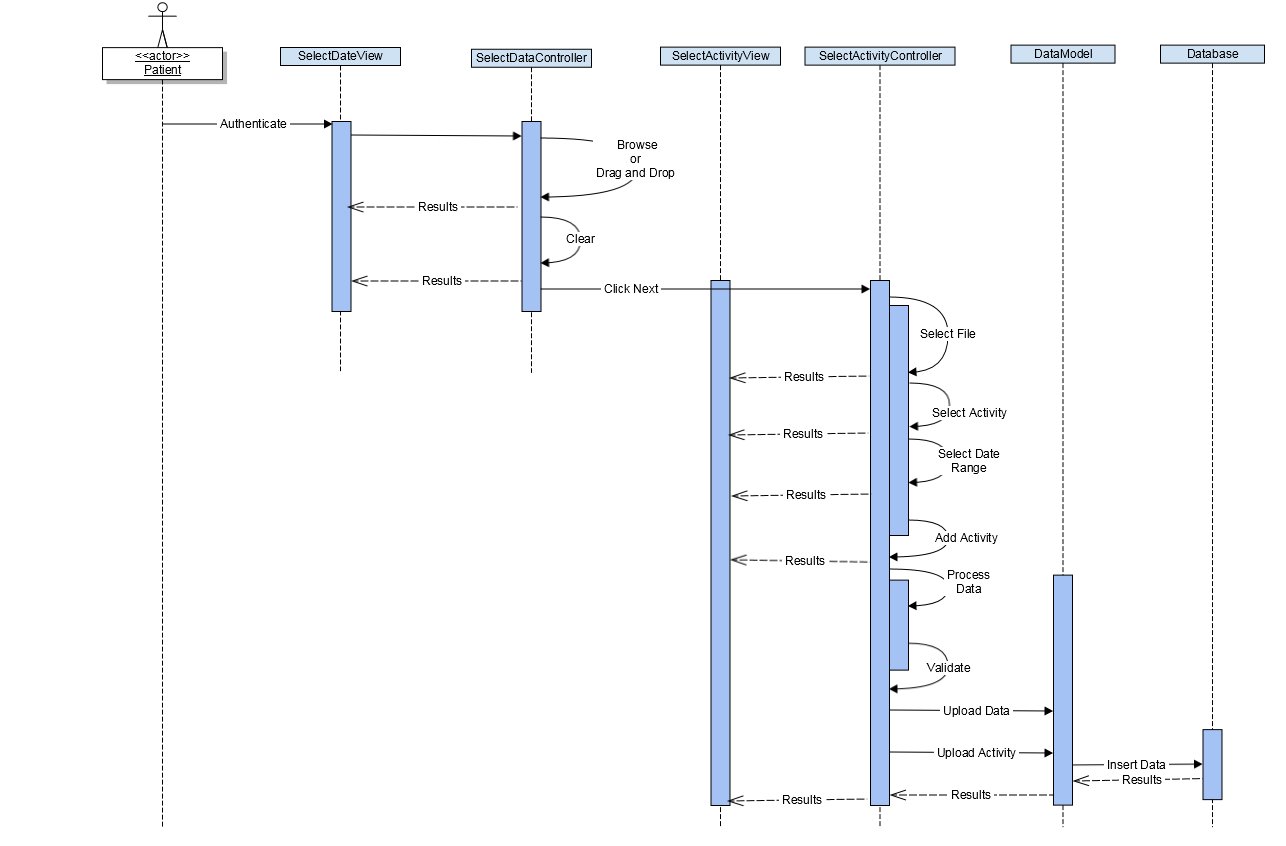
The following section is intended to describe the major features of the UAH Fit Vault system. Each feature shall contain a brief summary of the feature and information about how the feature will be implemented within the system. There will also be an accompanying sequence diagram to help visualize the relationship between the various system components needed to satisfy the feature’s functionality.

### Data Ingestion

Data ingestion is a feature that encapsulates the process of the patient selecting data files, assigning activities to the data, the system processing the data, and finally uploading the data and saving it to the system’s SQL database.

This process begins with a patient navigating to the **SelectDataView** and selecting the data that they would like to upload to the system. The data selection can occur in one of two ways. The system shall give the patient the ability to click a browse button on the view that will prompt the user with a modal that can traverse the file system structure of the machine the patient is using. The patient can use this tool to select a directory that contains the files that they wish to upload. The other option available for the patient is to drag and drop any files they wish to upload into the screen where the files would normally be listed. After either of these processes is performed the files will be listed in a list element on the view for review. The patient shall be given the opportunity to remove files from the list that they do not wish to upload. When the patient is satisfied the patient can click a “Next” button to proceed to assigning activities to the data.

After clicking the “Next” button the **SelectActivityView** will be loaded. This view allows the patient to enter any number of activities that occurred during the time of data capture. The patient will be provided with an interface to add, edit, and delete activities. Adding an activity will display date picker boxes to select the time period of the activity. There will also be a drop down select list containing preset activities that the user may choose from to label their data with an activity. Once the patient is satisfied with the activity list, the patient can click a “Upload” button to begin processing the data. After the system processes the data it will be uploaded and saved to the SQL database.



**Figure 2.2**: Sequence Diagram of Data Ingestion Feature

# 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:**

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

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** Activity()
  + Default constructor

#### AspNetUser

This class describes the attributes common to all users. This will act as a base class for **Patients**, **Physicians**, and **Experiment Administrators**.

**<<Entity>>**

**Attributes:**

* **Public Guid** Id
  + Unique id for each user.
* **Public string** UserName
  + The user name used by each user that will be used to log into the system.
* **Public string** PasswordHash
  + Hash string representing the user’s password used to log into the system.
* **Public AccountStatus** Status
  + Enumeration value that describes the current account status of the user.
* **Public Nullable<string>** PatientId
  + If the user is a patient this field will be used as a foreign key to the **Patient** table Id field.
* **Public Nullable<int>** Physician
  + If the user is a physician this field will be used as a foreign key to the **Physician** table Id field.
* **Public Nullable<int>** ExperimentManager
  + If the user is a experiment manager this field will be used as a foreign key to the **ExperimentManager** table Id field.
* **Public virtual AspNetRole** AspNetRoles
  + Link to the user role in the **AspNetRole** table

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** AspNetUser()
  + Default constructor

#### Patient

This class describes the attributes and construction of a **Patient** user object.

**<<Entity>>**

**Attributes:**

* **Public string** Id
  + Unique patient id created by the patient’s physician used to identify their patients.
* **Public int** Age
  + The age of the patient.
* **Public float** Weight
  + The weight of the patient.
* **Public int** Height
  + The height of the patient given in inches.
* **Public Region** Location
  + The state where the patient is located.
* **Public PatientEthnicity** Ethnicity
  + Enumeration value used to describe the patient’s ethnicity.
* **Public Collection<PatientRace>** Race
  + Collection of enumeration values used to describe the patient’s race.
* **Public PatientGender** Gender
  + Enumeration value used to describe the patient’s gender.
* **Public virtual Physician** Physician
  + Foreign key link to the patient’s physician.

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** Patient()
  + Default constructor

#### Physician

This class describes the attributes and construction of a **Physician** user object.

**<<Entity>>**

**Attributes:**

* **Public int** Id
  + Unique id generated by the system for each physician.
* **Public string** FirstName
  + Physician’s first name.
* **Public string** LastName
  + Physician’s last name.
* **Public string** Address
  + Physician’s office address information.
* **Public string** Email
  + Email address for the physician
* **Public string** PhoneNumber
  + Phone number for the physician
* **Public virtual Collection<Parent>** Patients
  + Collection of all of the physician’s patients.

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** Physician()
  + Default constructor

#### ExperimentAdministrator

This class describes the attributes and construction of an **Experiment Administrator** user object.

**<<Entity>>**

**Attributes:**

* **Public int** Id
  + Unique id generated by the system for each experiment administrator.
* **Public string** FirstName
  + Experiment administrator’s first name.
* **Public string** LastName
  + Experiment administrator’s last name.
* **Public string** Address
  + Experiment administrator’s address information.
* **Public string** Email
  + Email address for the experiment administrator
* **Public string** PhoneNumber
  + Phone number for the experiment administrator.
* **Public virtual Collection<Experiment>** Experiments
  + Collection of all of the experiment administrator experiments.

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** ExperimentAdministrator()
  + Default constructor

#### Experiment

This class describes the attributes and construction of an **Experiment** object.

**<<Entity>>**

**Attributes:**

* **Public int** Id
  + Unique id generated by the system for each experiment.
* **Public string** Name
  + Experiment name.
* **Public DateTime** LastModified
  + Date information for when experiment was last modified.
* **Public string** QueryString
  + Query information used to collect the experiment data.
* **Public virtual ExperimentAdministrator** ExperimentAdministrator
  + Foreign key relationship to the experiment administrator that created the experiment.

**Operations:**

* Sets and gets for all attributes.

**Constructors:**

* **Public** ExperimentAdministrator()
  + Default constructor

### Enumerations

#### ActivityType

This is the enumeration that contains the various activities a user could be performing while wearing the medical device and capturing data.

**<<Enumeration>>**

**Attributes:**

* Running
* Walking
* Sleeping
* Sitting
* Standing

#### AccountStatus

This is the enumeration that contains the different account statuses.

**<<Enumeration>>**

**Attributes:**

* Pending
* Active
* Inactive

#### PatientEthnicity

This is the enumeration that contains the ethnicity categories defined in the National Center for Education Statistics that a patient may belong to.

**<<Enumeration>>**

**Attributes:**

* Hispanic or Latino
* Not Hispanic or Latino

#### PatientRace

This is the enumeration that contains the racial categories defined in the National Center for Education Statistics that a patient may belong to.

**<<Enumeration>>**

**Attributes:**

* American Indian or Alaska Native
* Asian
* Black or African American
* Native Hawaiian or Other Pacific Islander
* White
* Other

#### PatientGender

This is the enumeration that contains the different gender types.

**<<Enumeration>>**

**Attributes:**

* Female
* Male

#### Region

This is the enumeration that contains the names of the fifty states that make up the United States plus other major regions in the world.

**<<Enumeration>>**

**Attributes:**

* Alabama
* Alaska
* American Samoa
* Arizona
* Arkansas
* California
* Colorado
* Connecticut
* District of Columbia
* Delaware
* Florida
* Georgia
* Guam
* Hawaii
* Idaho
* Illinois
* Indiana
* Iowa
* Kansas
* Kentucky
* Louisiana
* Maine
* Maryland
* Massachusetts
* Michigan
* Minnesota
* Mississippi
* Missouri
* Montana
* Nebraska
* Nevada
* New Hampshire
* New Jersey
* New Mexico
* New York
* North Carolina
* North Marianas Islands
* North Dakota
* Ohio
* Oklahoma
* Oregon
* Pennsylvania
* Puerto Rico
* Rhode Island
* South Carolina
* South Dakota
* Tennessee
* Texas
* Utah
* Vermont
* Virginia
* Virgin Islands
* Washington
* West Virginia
* Wisconsin
* Wyoming
* European Union
* Eastern Europe
* Africa
* South America
* Central America
* Asia
* Oceania
* North America
* The Caribbean

## Data Access Layer Overview