Software Architecture Document

<Project Name>

Company Name

Street Address

City, State Postcode

4 July 2019

FOR NGO – I USED THIS

<https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm#Purpose>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | **Date** | **Author(s)** | **Change Description** |
| 1.0 | dd/mm/yyyy | xxxxxxxxxxxx | First draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1 Introduction 3](#_Toc15682846)

[1.1 Purpose 3](#_Toc15682847)

[1.2 Scope 3](#_Toc15682848)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc15682849)

[1.4 References 3](#_Toc15682850)

[1.5 Overview 3](#_Toc15682851)

[2 Architectural Representation 3](#_Toc15682852)

[3 Architectural Goals and Constraints 3](#_Toc15682853)

[4 Use-Case View 3](#_Toc15682854)

[4.1 Use-Case Realizations 4](#_Toc15682855)

[5 Logical View 4](#_Toc15682856)

[5.1 Overview 4](#_Toc15682857)

[5.2 Architecturally Significant Design Packages 4](#_Toc15682858)

[6 Process View 4](#_Toc15682859)

[7 Deployment View 4](#_Toc15682860)

[8 Implementation View 4](#_Toc15682861)

[8.1 Overview 5](#_Toc15682862)

[8.2 Layers 5](#_Toc15682863)

[9 Data View (optional) 5](#_Toc15682864)

[10 Size and Performance 5](#_Toc15682865)

[11 Quality 5](#_Toc15682866)

# Introduction

## Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made during the project’s lifetime.

## Scope

This Software Architecture Document provides an architectural overview of the SRV System. The SRV System is being developed by Team\_NAG to allow students to keep track of his or her curricular results.

Additional implementations allow the lecturers to check students progress and provide admin staff with CRUD functionalities.

## Definitions, Acronyms, and Abbreviations

*Table1 - Definitions*

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SRS | Software Requirements Specification; a document that details the scope and criteria for the project’s solution. Will be used to refer to this document. |
| SRV | Student Results View, the name of the project this document will refer to. |
| TAFEBuddy | Refers to the overarching TAFEBuddy architecture. TAFEBuddy is a previously explored software suite by faculty at TAFESA that seeks to delegate minor administrative functions to the end user (i.e. the student or the lecturer). |
| TAFE | Acronym for Tertiary And Further Education. Term used to refer to government-owned tertiary educational institutes within Australia. TAFESA refers to the entity that exists solely within South Australia (Tertiary And Further Education South Australia) |
| User | A stakeholder who interacts with the software solution. |
| Role | A title given to a user that determines the purpose of the user in the system. Role is defined as an attribute of the user. |
| Student | A user has the role of student and is enrolled as a student at TAFESA. |
| Lecturer | A user who has the role of lecturer and is employed at TAFESA as an instructor for the institution. |
| Admin/Administrator | A user who has the role of administrator and is employed at TAFESA as faculty for administrative purposes. An actor whose purpose is to maintain the backend of the institution. |
| Stakeholder | Any person who interacts with the system that is not a developer. |
| Client | Refers to the project’s sponsor, liaison or otherwise. |
| ICT | Acronym for Information & Computation Technology. |
| HTTPS | Hypertext Transfer Protocol Secure |

## References

*[This subsection should provide a complete list of all documents referenced elsewhere in the* ***Software Architecture Document****.  Each document should be identified by title, report number (if applicable), date, and publishing organization.  Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

As Listed in each section, with a link if supporting documentation is needed.

## Overview

*[This subsection should describe what the rest of the* ***Software Architecture Document*** *contains and explain how the* ***Software Architecture Document*** *is organized.]*

*Todo*

# Architectural Representation

This document presents the architecture as a series of views: use case view, logical view, process view and deployment view.

These views are represented using Unified Modeling Language (UML).

# Architectural Goals and Constraints

*[This section describes the software requirements and objectives that have some significant impact on the architecture, for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]*

Key requirements put into place for the SRV software solution are:

* 1. The SRV won’t operate as a stand alone project, but will be part of a larger overarching app (TAFEBuddy System) and any solution provided will have to have room for expansion.
  2. The application will be accessible on the web from different devices and will be available both on desktop and mobile systems.
  3. All students, lecturers and admin staff will access a different view upon login according to his or her designated role.
  4. The SRV System must ensure complete protection of data from unauthorized access. All access is to be ruled by server-side validation.
  5. The SRV software solution will be implemented as a client-server system. The software solution should provide a service which should be accessible by intended stakeholders.

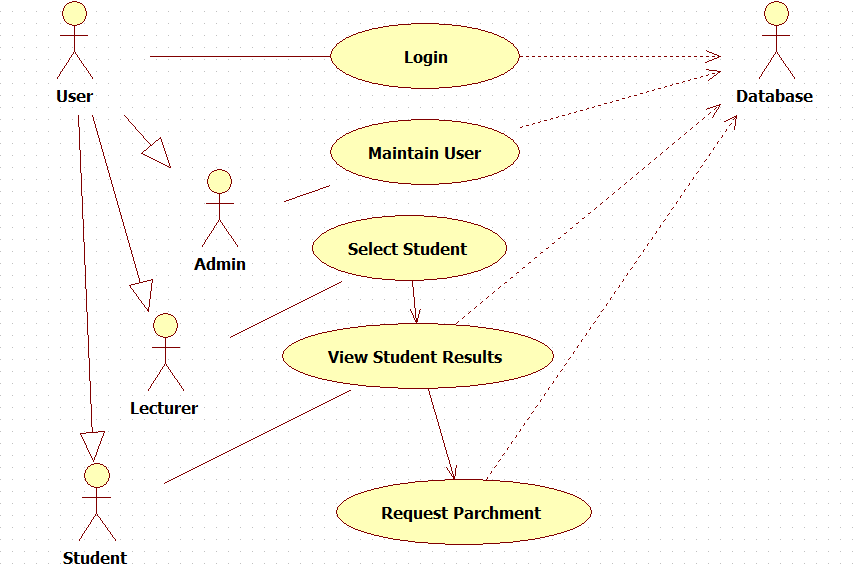
# Use-Case View

*[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage - they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.]*

The use case modelling of the software solution represents the possible interations a stakeholder may have with the implemented system.

The identified use cases for SRV are:

* + - Login
    - Maintain User
    - Select Student
    - View Student Results
    - Request Parchment



4.0.1 Login:

This use case describes how a user logs into the Course Registration System. The actors starting this use case are Student, Lecturer, and Admin.

4.0.2 Maintain User:

This use case allows the Admin to maintain a student information, a lecturer information or another admin infromation. This includes adding, modifying, and deleting users from the system. The actor of this use case is the Admin.

4.0.3 Select Student:

This use case allows the Lecturer to select one student from a list of enrolled students, in order to access the student’s results. The actor starting this use case is the Lecturer.

4.0.4 View Student Result:

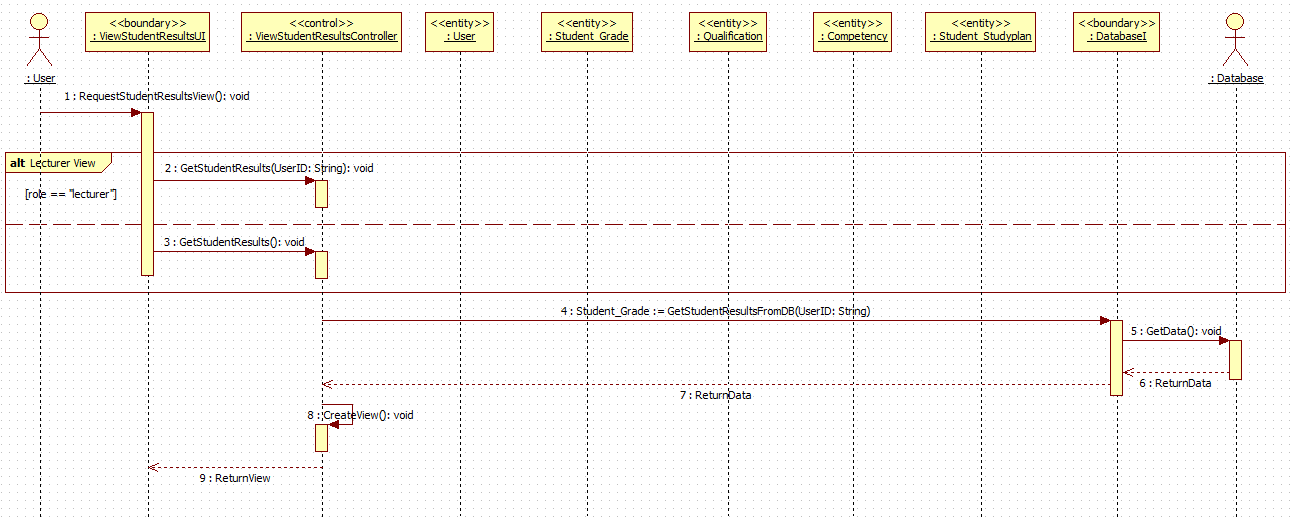
This use case allows the user to access the student result view for a particular student. The actors starting this use case are the Student and the Lecturer.

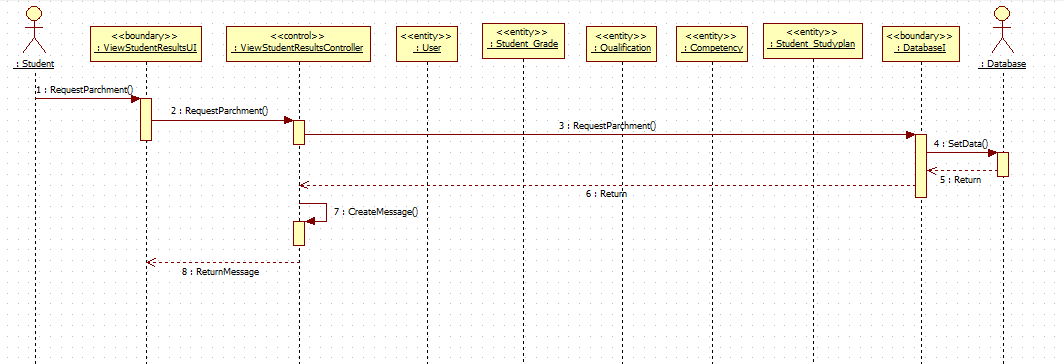
4.0.5 Request Parchment**:**

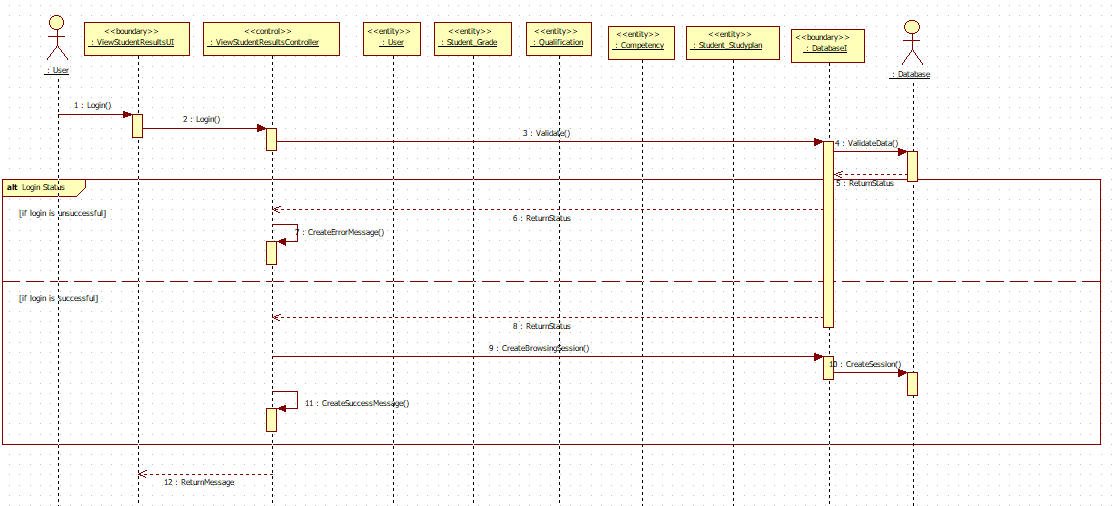
This use case describes how, upon reaching 100% of the completion of a qualification, a student is allowed to request a parchment. The actor starting this use case is the Student.

## Use-Case Realizations

*[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]*







# Logical View

*[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]]*

*Todo*

## 

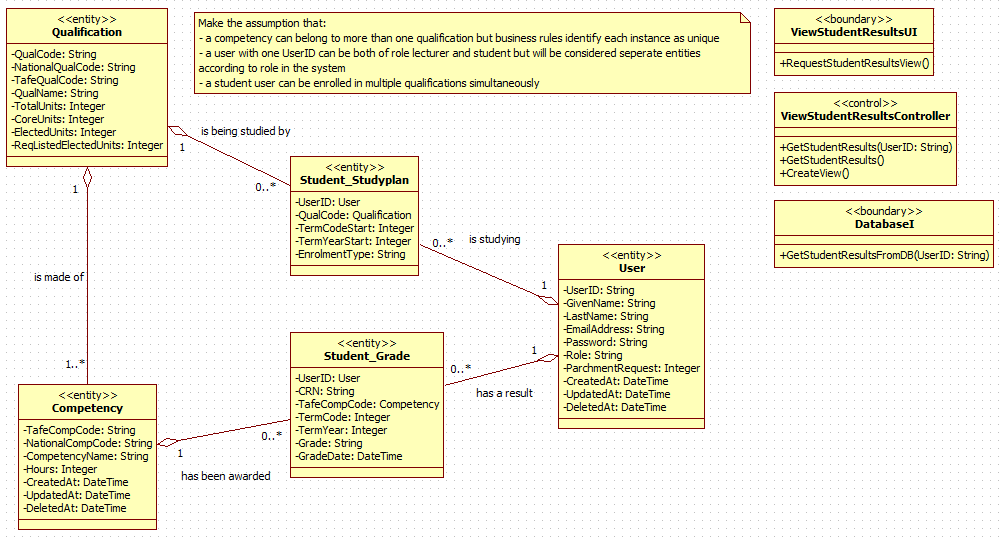
## Overview

*[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]*

The logical view of the software architecture represents the data structure aspect of the intended software solution. This can be represented by the use of class, sequence and deployment models to visually respresent the structure of the data being manipulated.

A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

## Architecturally Significant Design Packages



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class name | Description | Role | Attributes | Relations to |
| Qualification | Qualification entity of the database. | Provides details about the certificates being studied. | QualCode VARCHAR(20)  NationalQualCode VARCHAR(32)  TafeQualCode VARCHAR(32)  QualName VARCHAR (100)  TotalUnits INT(11)  CoreUnits INT(11)  ElectedUnits INT(11)  ReqListedElectedUnits INT(11) | studyplan\_qualification  student\_studyplan  subject\_qualification |
| Competency | Competency entity of the database. | Provides details about the competency components of the qualification being studied. | TafeCompCode VARCHAR(10)  NationalCompCode VARCHAR(20)  CompetencyName VARCHAR(128)  Hours INT(11) | crn\_detail  subject\_competency  competency\_qualification |
| Student\_Studyplan | Student\_Studyplan entity of the database with OOP. | Provides details about the relationship between a user and the qualification he/she is studying. | StudentID VARCHAR(9)  QualCode VARCHAR(20)  TermCodeStart INT(1)  TermYearStart INT (4)  ENROLMENTtYPE varchar(20) | user  qualification  term\_datetime |
| Student\_Grade | Student\_Grade entity of the database with OOP. | Provides details about the relationship between the user and the competencies he/she has been awarded. | StudentID VARCHAR(9)  CRN VARCHAR(32)  TermCOde INT(1)  Grade VARCHAR(2)  GradeDate DATE | user  term\_datetime  crn\_detail |
| User | User entity of the database. | Provides details about memebers of the TAFESA system. | UserID VARCHAR(9)  GivenName VARCHAR(64)  LastName VARCHAR(64)  EmailAddress VARCHAR(64)  Password VARCHAR(255)  Role VARCHAR(45)  ParchmentRequest TINYINT(1) | student\_studyplan  student\_grade |

# Process View

*[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]*

The system consists of a web based application that relies on HTTPS to transfer messages between the user and the client.

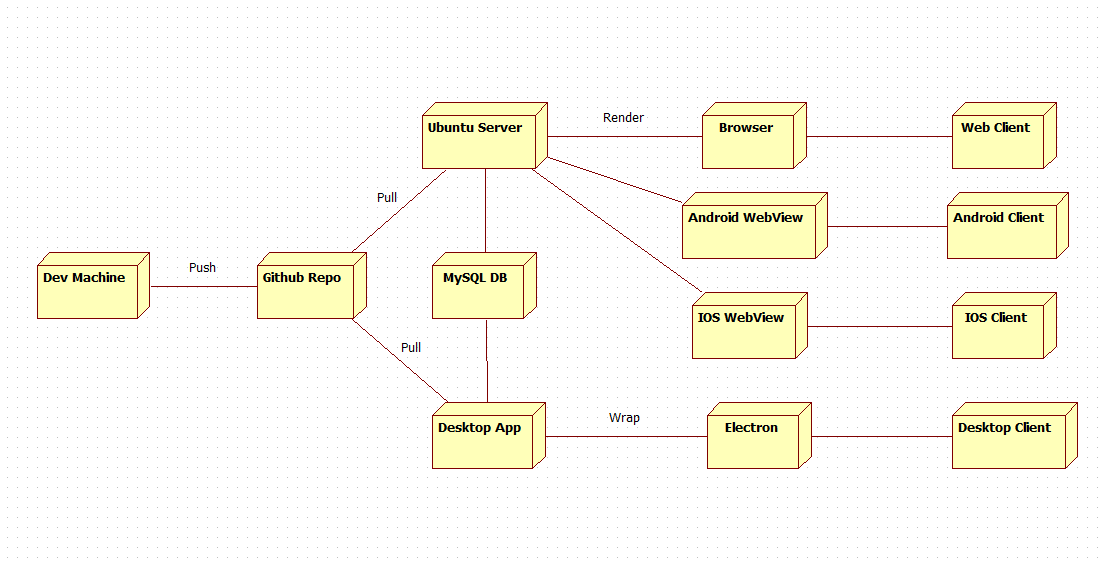
The communication is secured by encryption using a Secure Socket Layer.

The application is hosted on a remote server.

# Deployment View

*[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software, and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the* ***Process View*** *onto the physical nodes.]*

To run the application all that is required is a browser and a connection to the internet. At current stage every mobile device and every computer with installed a functioning browser will be able to access use the SRV and all of its functionalities.



# Implementation View

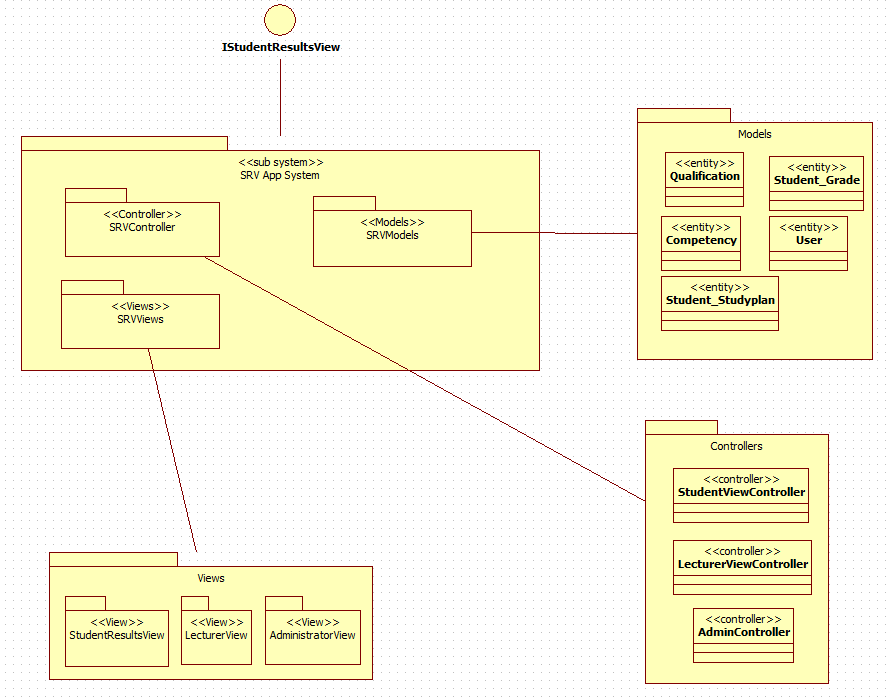
*[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]*

## Overview

*[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers.]*

## Layers

*[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]*

**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | Type | Name | Item Type | Item Name |
| Presentation | Framework | Helper |  | Node.JS, JS Handlebars |
|  | Package | Views | <<views>> | StudentResultsView, LecturerView, AdministratorView |
| Business | Subsystem | Persistence | <<server>> | db\_tafebuddy |
|  | Subsystem | Models | <<entity>> | Qualification,Competency, Student\_Grade, Student\_Studyplan, User |
| Service | Package | Controllers | <<controller>> | StudentViewController, LecturerViewController, AdminController |

# Data View (optional)

*[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]*

# Size and Performance

*[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]*

The Electron Framework being implemented for the project will allow SRV to be near-infinitely scalable. The use of web technologies to produce a responsive front-end layout will allow the project to be implemented on all types of platforms available on the market today. Although the project was initially targeted at desktop and mobile applications, the final iteration featuring Electron will mean the software solution should meet target outcomes and even exceed client expectations.

The use of web technologies also means that the software solution will be relatively less intensive to run than most native implementations. Heavy duty frameworks like UWP, ASP.NET or J2EE would result in a much bulkier implementation than if done by running the service as a Node.JS webservice.

Caching

Just in time

# Quality

*[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, for example safety, security or privacy implications, they should be clearly delineated.]*

The software solution proposed will use the JavaScript Handlebars framework to implement a Model View Controller (MVC) design pattern. This will allow the intended solution to have a means to separate the concerns of each module designed for the solution and provide an easier method of implementation across a wider area network.

The implementation of the end-user graphical interface will only concern the graphical representation of the input/output from the system so anything programmatically generated by the client’s device will be kept agnostic about the workings of the business logic supporting the application. This layer of obfuscation will provide the final implementation an extra layer of security by isolating the mission critical infrastructure from any tampering by an unwarranted third party.

UI standards

Bootstrap framework

Coding standards

Handlebars templating framework

Architecture (MVC)

MVC Fully supported

Testing implemented

JUNIT Test

Database tweaking

Express MySQL

User acceptance factors

<https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces>