**Functional Specification**

**Instructor-Course Assignment Application**

**Project Manager: Kiana McDaniel**

**Phillip Tran, David Johnson**

**Faculty Advisor: Dr. Rich Compeau**

**Texas State University**

**Ingram School of Engineering**

**Texas State University**

**601 University Dr**

**San Marcos, Texas, 78666**

**10/15/2018**



|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Date** | **Description** | **Author** |
| .1 | 09/28 | Started file, assigned sections | David + Kiana |
| .2 | 09/29 | Updated most -David sections | David |
| .3 | 10/04 | Started my sections | Phil |
| .4 | 10/07 | Finished first draft of use cases | David |
| .5 | 10/08 | Updated a lot of stuff | David + Kiana |
| .6 | 10/09 | Finished initial draft (excluding matrix tables and section 2.10) | David + Kiana + Phillip |
| .7 | 10/10 | Added some graphs and diagrams for the interface and dataflow | David + Kiana |
| .8 | 10/11 | Added Software Versions | Phil |
| .9 | 10/11 | Added matrix table, updated project name | Kiana |
| 1.0 | 10/14 | Added hardware diagram | Phil |
| 1.1 | 10/29 | Updated Functional Specification to reflect project changes made during IDR prep | David |
| 1.2 | 11/04 | Revised Functional Specification to reflect sponsor and instructor comments from IDR | David |

Table of Contents

[1 Introduction 3](#_Toc526926343)

[1.1 Summary – Kiana & David 3](#_Toc526926344)

[1.2 Sponsor Requirements – Kiana 3](#_Toc526926345)

[1.3 Existing System – Kiana 4](#_Toc526926346)

[1.4 Terminology – Phillip 4](#_Toc526926347)

[2 Functional Description 5](#_Toc526926348)

[2.1 User Attributes and Use Cases – David 5](#_Toc526926349)

[2.2 Administration Functions – Kiana 6](#_Toc526926350)

[2.3 Error Handling – Phillip 6](#_Toc526926351)

[2.4 Safety and Security – David 6](#_Toc526926352)

[2.5 Help and User Documentation – Phillip 7](#_Toc526926353)

[2.6 Interfaces 7](#_Toc526926354)

[2.6.1 User – Kiana 8](#_Toc526926355)

[2.6.2 Software – David 8](#_Toc526926356)

[2.6.3 Hardware – Phillip 9](#_Toc526926357)

[2.6.4 Mechanical N/A 9](#_Toc526926358)

[2.7 Boundary Conditions and Constraints – David 9](#_Toc526926359)

[2.8 Performance – Phillip 10](#_Toc526926360)

[2.9 Software Platforms – Kiana 11](#_Toc526926361)

[2.10 Service, Support, & Maintenance – Phillip 11](#_Toc526926362)

[2.11 Expandability or Customization – David 11](#_Toc526926363)

[3 Project Alignment Matrix – ALL 12](#_Toc526926364)

[4 References – David 13](#_Toc526926365)

[5 Approvals 13](#_Toc526926366)

# **Introduction**

This document contains the functional specification for the Instructor-Course Assignment Application. The Instructor-Course Assignment Application is a web-based application that allows University Program Coordinators to assign Instructors to the Courses that they are teaching for a certain school-year semester. It also allows the user to export a complete document of all Instructor-Course assignments as an XML file. Figure 1.1 is a top-level block diagram of our system which also lists the user inputs and system outputs.

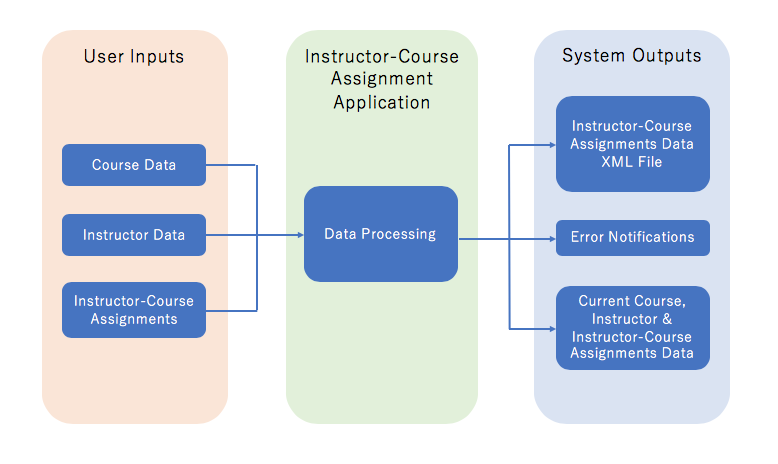


Figure 1.1 top-level block diagram

## Summary – Kiana & David

The Instructor-Course Assignment Application is a web-based application that encompasses the software components needed to allow users to remotely access a database and manage Instructor-Course assignments. Interaction with the system occurs through a web-page based front end that allows the user to view and edit Instructor and Course information. The primary goal of this project is to create an application that is convenient and ergonomic for the user, while remaining simple to implement and maintain by Administrators. The project is being designed with the intention of being implemented on a secure server on the Texas State University network system. Users of this application include Program Coordinators (who will eventually be Department Chairs), the Director (who will eventually be the Dean), Administrative Assistants of the Ingram School of Engineering, and System Administrators.

## Sponsor Requirements – Kiana

A key requirement outlined by the sponsor is to produce a system designed around usability for Program Coordinators. The proposed system will have a user-friendly, easy-to-navigate interface that presents an intuitive process of Instructor-Course assignment. A second requirement is to integrate the XML file export feature into the application. The exported XML file is to contain information regarding Instructor-Course assignment to be used by an outside class scheduling software, UniTime ref[1].

## Existing System – Kiana

Currently, Program Coordinators use an outdated, hard-to-navigate Excel Spreadsheet to assign Instructors to Courses. This method is inefficient, as it causes users to spend more time figuring out where features and data are located, rather than simply completing the task. Our product will benefit Program Coordinators by having a user-friendly, easy-to-navigate interface that provides a clear guidance on the process of Instructor-Course assignment. By developing a web application, users that have access will be able to view the most up to date assignments by logging in to the system. Another problem with the current method is that it exists solely on the hard drive of a user’s computer. Our product solves this problem of reliability by using a database server to host Instructor and Course data off client.

## Terminology – Phillip

|  |  |
| --- | --- |
| HTML – 4.0 | Hypertext Mark-up Language – Language used to generate browser visuals and functions |
| CSS | Cascading Style Sheets – Language used for bookmarking html styles |
| JS | JavaScript – Language used for client-side scripting and templating front end application |
| SQL | Structured Query Language – Language used for the database |
| Course | Unit subject as defined by the Texas State University Course Catalogue |
| Class | Singular unit/section of a course with time and location |
| HTTPS | Hypertext Transfer Protocol Secure – Application protocol for web systems + TLS |
| JSON | JavaScript Object Notation – Formatting for transferring data structures from front end to back end |
| Angular – 6.1.0 | JavaScript/TypeScript web framework for encapsulating and generating front end modules and handling observables/injectables |
| Flask – 1.0.2 | Python web framework |
| SQLAlchemy – 1.2.8 | Python library for handling SQL type initialization and database queries |
| TLS | Transport Layer Security – Standard security protocol for providing communications over HTTPS |

# **Functional Description**

## User Attributes and Use Cases – David

This system will have three types of users, the Program Coordinators and Director who can view all the data and make changes as necessary to any of the courses and Instructors, the Administrative Assistants who will have view only access to the system, and the System Administrators who will have access to the entirety of the program, but whose responsibilities extend only to upkeep.

The system will be designed to allow the user to stop any of these processes at any step and branch to a different use case.

* Review Instructor Information
  1. User logs in through the login portal using their credentials
  2. (optional) User clicks notifications/alert button to view conflicts
  3. User selects “Instructors” tab to view full list of Instructors
  4. User selects specific Instructor to open detailed view
* Review Course Information
  1. User logs in through the login portal using their credentials
  2. (optional) User clicks notifications/alert button to view conflicts
  3. User selects “Courses” tab to view full list of Courses
  4. User selects specific Course to open detailed view
* Add course

1. User logs in through the login portal using their credentials
2. User selects “Courses” tab
3. User selects “Add New Course”
4. User enters information regarding the Course and confirms
5. System adds Course to database and updates all pages

* Add Instructor
  1. User logs in through the login portal using their credentials
  2. User selects “Instructors” tab
  3. User selects “Add New Instructor”
  4. User enters information regarding the Instructor and confirms
  5. System adds Instructor to database and updates all pages
* Edit course
  1. User logs in through the login portal using their credentials
  2. User selects “Courses” tab to view full list of courses
  3. User selects specific course to open detailed view
  4. User selects “Edit Course”
  5. User changes information regarding course and confirms
  6. System updates Course information and all pages
* Edit Instructor
  1. User logs in through the login portal using their credentials
  2. User selects “Instructors” tab to view full list of Instructors
  3. User selects specific Instructor to open detailed view
  4. User selects “Edit Instructor”
  5. User changes information regarding Instructor and confirms
  6. System updates Instructor information and all pages
* Assign Instructor to course
  1. User logs in through the login portal using their credentials
  2. User selects “Courses” tab to view full list of Courses
  3. User selects specific Course to add Instructor
  4. User selects Instructor from list to add as Instructor for Course

\*\*\*OR\*\*\*

1. User selects “Instructors” tab to view full list of Instructors
2. User selects specific Instructor to add to a Course
3. User selects Course from list to assign the Instructor to
4. System updates Instructor/Course information and all pages

* Export data
  1. User logs in through the login portal using their credentials
  2. User selects “Export Data” tab
  3. User selects method of exportation
  4. System provides formatted export file to user

## Administration Functions – Kiana

The users of this application are Program Coordinators, the Director, Administrative Assistants, and System Administrators. Program Coordinators & Directors are authorized to view and edit Instructor & Course information and make Instructor-Course assignments. Administrative Assistants are authorized to view Instructor & Course information and Instructor-Course assignments. System Administrators have access to the administrative functions of the system, which include installing and maintaining the web-page, backend and database systems on a server. In addition to administrative functions, System Administrators have access to the entirety of the program, since it is not feasible to design a system that can be implemented by a third party that does not have this access.

## Error Handling – Phillip

Text Input Errors (User) - Errors in Text Input will be handled by JavaScript/HTML filters. If a user inputs an incorrect text input, a warning will pop up requesting that the user change their input, and the user will not be able to continue until the error is resolved.

Assignment Input Errors (User) - Assignment Input Errors occur when the user attempts to assign an Instructor to a Course that does not meet the requirements for the Instructor or the Course. A warning log will display all the accumulated assignment errors; however, the user will be able to continue creating assignments and override the errors.

Network Errors - Errors on the server side will be handled using HTTPS error codes. Such that if the server is down for maintenance or if there is some error in transferring data from the server to the webpage then a relevant error page will be displayed. (e.g. 404 Page not found if the user attempts to access a page that does not exist or is currently inaccessible)

## Safety and Security – David

There are no physical safety considerations that need to be made for this project, since it is purely software based and has no methods of physically interacting with the user. The data saved and used in the program will be frequently backed up (as needed) to a secure database to provide information safety in the event that data is accidentally deleted or incorrectly altered in the program. Access to the program itself will be restricted to certain authorized individuals via a username and password combination unique to each user. This is currently planned to be implemented via Texas State’s NetID log-in system. The data will be encrypted if and as needed during data transfer and storage.

## Help and User Documentation – Phillip

User documentation will be provided based on the type of user and the authorization that they have. These documents will serve as a “How To” Guide for specific users. A list of documents along with their contents is shown below.

* Program Coordinator & Director Guide
  1. View Course Information
  2. View Instructor Information
  3. View Instructor-Course Assignments
  4. Edit Course Information
  5. Edit Instructor Information
  6. Make Instructor-Course Assignments
  7. Export Instructor-Course Assignments
* Administrative Assistant Guide
  1. View Course Information
  2. View Instructor Information
  3. View Instructor-Course Assignments
* System Administrator Guide
  1. Install Dependencies
  2. Deploy the Software on Windows, OSX and Linux
  3. Navigate the Terminal to Administer the Database
  4. Request Data from the Flask API

## Interfaces

### User – Kiana

The users of this application interface with it through their personal computers or laptops, and web browser. A keyboard and mouse are used to enter data and navigate the system. As long as the user has a device with text input capabilities and access to the web-page via the internet or a direct connection then the application will function. The user interfaces with the system through a web-page based frontend. Figure 2.1 shows a mock-up of the “Courses” page of the frontend webpage.

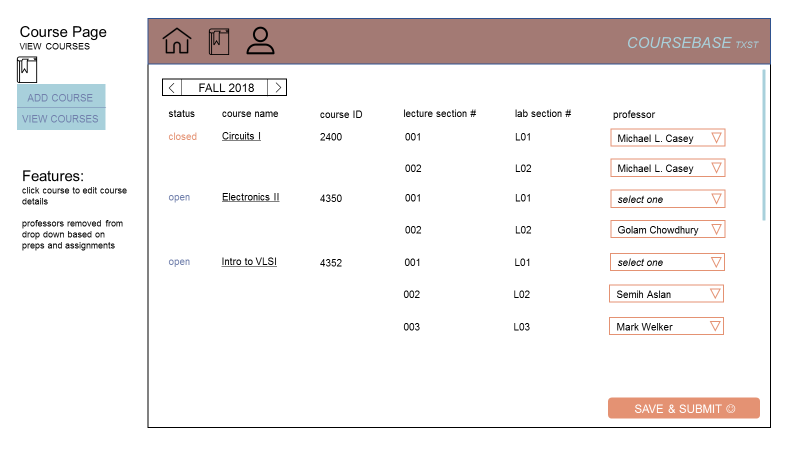


Figure 2.1 UI Mock-up

### Software – David & Phillip

The second stage of the project is the class scheduling (as opposed to course scheduling), this functionality will be implemented entirely by UniTime ref[1]. To work with UniTime, the course scheduling software will include the functionality of outputting a formatted XML file which contains all the course and Instructor information to be used by UniTime.

The project will be dependent on and will include the following software:

* NodeJS - 10.11.0
* AngularCLI - 6.1.0
* Python - 3.7.0
* Flask - 1.0.2
* SQLAlchemy - 1.2.8
* PostgreSQL - 9.5.14

Figure 2.2 shows the dataflow between the user, the frontend webpage interface, the backend software that handles error checking and manages the user interface, and the SQL database that all of the used data and information is saved to and accessed from.

In addition to these interfaces, we’re currently exploring the possibility of interfacing with some of Texas State’s existing systems. Specifically, their log-in system to handle regulating access to our program, as well as their course and instructor databases to handle populating our program with initial instructor and course information.

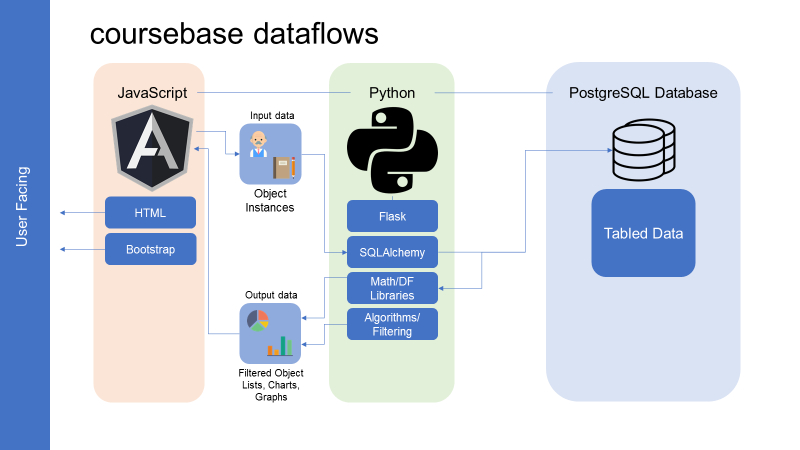


Figure 2.2 Software Dataflow

### Hardware – Phillip

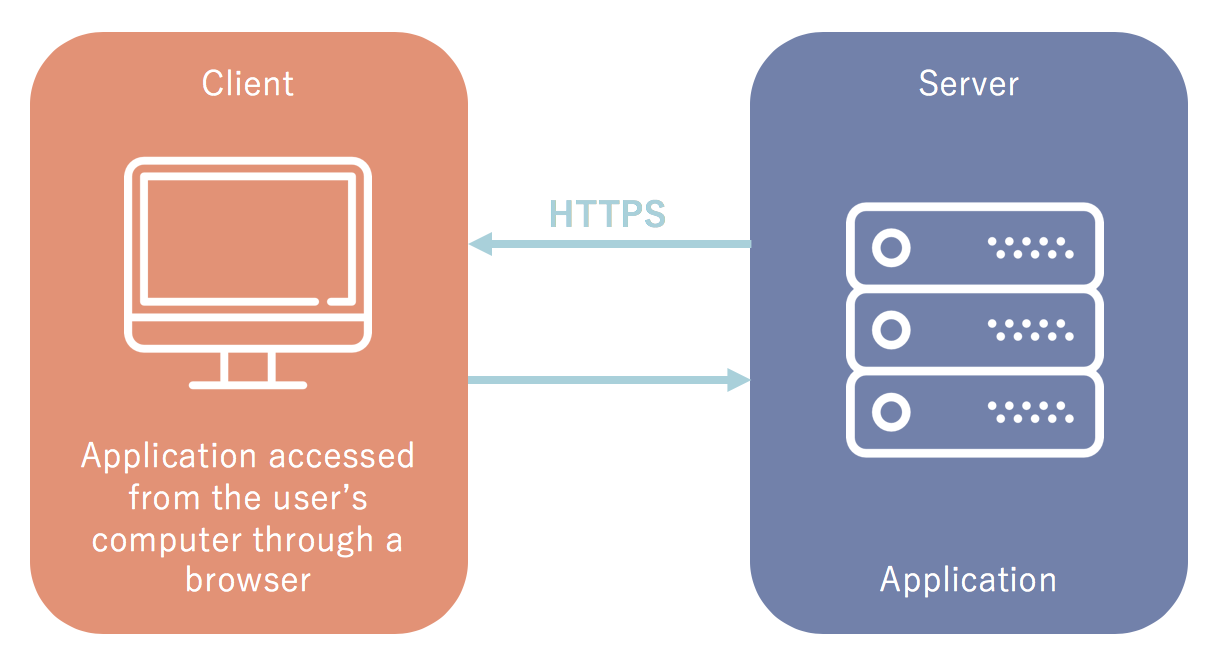


Figure 2.3 Hardware Interface

The project is purely software, however, regarding hardware, the application will require a server to run on. The specifications listed here are the hardware used during development and testing.

Recommended System Requirements (For Hosting Application)

* Processor: 2.4GHz Intel Core i5-6300U (Or Equivalent)
* Memory: 8GB LPDDR3
* Storage: 20GB
* Network: An ethernet or Wi-Fi connection

### Mechanical N/A

This project understandably does not have any mechanical interfaces.

## Boundary Conditions and Constraints – David

The user interface must be able to be embedded into a webpage or otherwise remotely accessed, this necessarily limits the language and software that can be used to implement the user interface.

The platform that the system is deployed on has to support the software the system is employing, namely AngularJS, Flask, and MySQL. In other words, it must satisfy the hardware conditions as listed in 2.6.3.

The system itself needs to be deployed on a platform that is stable enough in nature that throughout its normal usage will not corrupt or interrupt the system. This means that the platform needs to be able to reset to a functioning state if it is shut off or loses power and begin running the course scheduling system automatically.

Per Dr. Compeau’s request, a functional beta version of the program must be operational before Winter break.

Texas State University has strict policies on what programs can be hosted on their servers, it is unclear whether their policies would allow a program like ours to be run on their servers. If Texas State University will not allow our program to run on their servers, an alternative will have to be determined.

## Performance – Phillip

Since this program is purely software based, it is difficult to exactly define certain performance parameters. For the sake of simplicity, the performance section has been split into “non-functional requirements” which are close to what would be expected of the performance, these are requirements of the system’s performance that are not tied to any specific function. The other type of requirement are the “functional requirements” which are measurements of whether or not the program works properly as intended.

|  |  |  |
| --- | --- | --- |
| **Non Functional Requirements** | | |
| **Parameter** | **Requirement** | **Test Method** |
| Response time | 50ms response | Check response time during simulated normal use with browser dev tools |
| Memory Usage | Less than 500MB | Use operating system tools to check memory usage |
| Security | Binary | Check data stream with a middleman application |
| Stability | Binary | Use a “monkey test” to brute-force the UI |

|  |  |
| --- | --- |
| **Functional Requirements** | |
| **Function** | **How Tested** |
| View Courses | Unit Testing   * Use Scripts to automate functions and input * Retrieve data from the SQL database and compare to the function’s inputs * Manual inputs to the UI to test various edge cases * Observe the UI to ensure it reflects the inputs made |
| Add a Course |
| Edit a Course |
| View Instructors |
| Add an Instructor |
| Edit an Instructor |
| Make an Instructor-Course Assignment |

## Software Platforms – Kiana

Our system will support Linux Ubuntu -18.04, Windows 10 - 10.0.1734, and macOS - 10.13.6. It will also support any web browser as well as their associated mobile browsers. While this is not necessarily a requirement, all these environments have been and will continue to be used in the development process, so the system will be able to run on them

## Service, Support, & Maintenance – Phillip

Our project’s code is designed to be modular as shown in figure 2.4. The Database API is integrated into the frontend of our system via Angular Injectables. All frontend components are separated (Home Page, Instructor, Course, etc) into their own modules and classes, which makes maintenance more accessible.The backend will have maintenance functions with Python decorators, so System Administrators can manage the Database outside of the frontend application. API documentation will be provided for the Database.

## Expandability or Customization – David

The system is currently intended to interface with UniTime, however a potential expansion in the future would be to replace UniTime with an equivalent software integrated into the system. Since the system is intended to be modular (user interface, processing, and database/storage are separate), any individual component can be modified by changing its source code which will be provided to the system administrators and program coordinators.

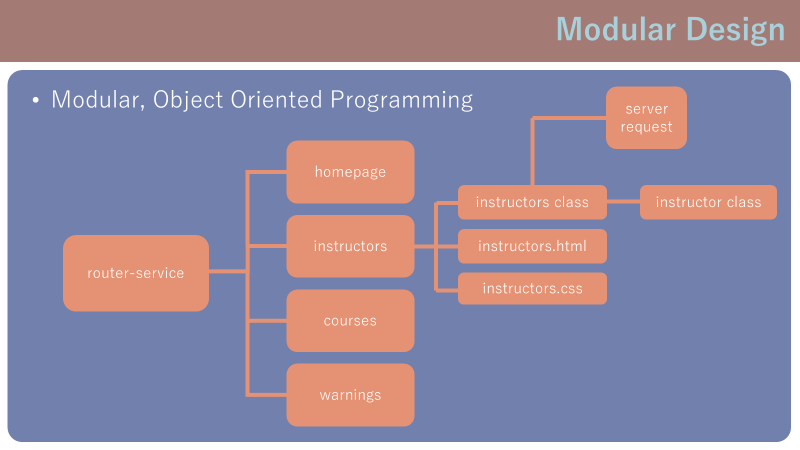


Figure 2.4

# **Project Alignment Matrix** – ALL

Outside Advisors (if any) and affiliations:

**TABLE 1: Knowledge Alignment Matrix**

|  |  |  |
| --- | --- | --- |
| **Course No.** | **Core knowledge** | **Specific knowledge incorporated by team** |
| EE 3350 (Electronics I) | Design and analysis of active devices and equivalent circuits | Understanding the distinctions between ideal and real performance and modelling the system so the real performance will still be sufficient. |
| EE 3370 (Signals and Systems) | Frequency domain representation of signals and frequency response, transfer functions | The data processing can be considered as a transfer function that is independent of its data, helping abstract the program |
| EE 3420 (Microprocessors) | Principles of operation and applications of microprocessors | Breaking down larger program goals into discrete, manageable pieces and designing them one at a time. |
| EE 4352 (Introduction to VLSI Design) | Analysis and design of CMOS integrated circuits | The method of comprising a larger system with smaller components is similar to our modular design process |
| EE 4370 (Communications Systems) | Transmission of signals through linear systems, analog and digital modulation, and noise | Analysing the channel and method of communication between two units and characterizing that channel and the format of data passed. |

**TABLE 2: Knowledge Alignment Matrix**

|  |  |  |
| --- | --- | --- |
| **Course No.** | **Core knowledge** | **Specific knowledge incorporated by team** |
| CS 3358 (Data Structures and Algorithms) | classic data structures and the analysis of algorithms | Selecting the proper algorithm and data type to most effectively achieve project goals |
| CS 3354 (Object-Oriented Programming) | Principles of object-oriented design and programming | UML, Encapsulation and Inheritance |
| CS 3398 (Software Engineering) | Principles of software design, implementation and validation techniques | UML, software design process |

**TABLE 3: Constraint Alignment Matrix (and applicable standards)**

ABET Criterion 3 (c): “an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.”

|  |  |
| --- | --- |
| **Constraint Type** | **Specific Project Constraint** |
| Economic | The project has to be able to be implemented in a low-cost form for the University, ideally no additional hardware upkeep cost and a single initial hardware purchase |
| Environmental | N/A |
| Health and safety | N/A |
| Social/Ethical | The data must be restricted to the aforementioned users, to avoid potential conflicts of interest or misunderstandings from other individuals involved in the University |
| Applicable Standards | N/A |

# **References** – David

[1] UniTime 4.3 Online Documentation: <http://help.unitime.org/>

# **Approvals**

The signatures of the people below indicate an understanding in the purpose and content of this document by those signing it. By signing this document, you indicate that you approve of the proposed project outlined in this Functional Specification and that the next steps may be taken to proceed with the project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Approver Name** | **Title** | **Signature** | **Date** |
|  | Project Manager |  |  |
|  | D2 Project Manager |  |  |
|  | Faculty Sponsor |  |  |
|  | Sponsor |  |  |
|  | Instructor |  |  |