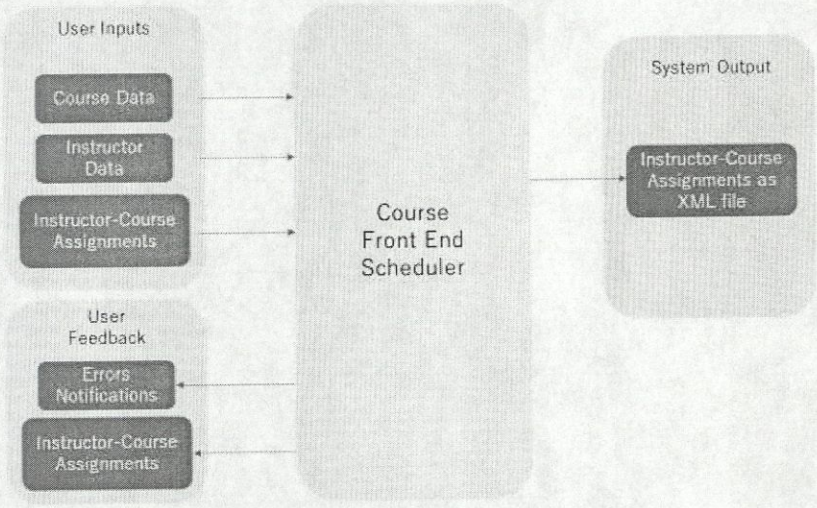
1 Introduction

This document contains the functional specification for the Course Front End Scheduler. The Course Front End Scheduler 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. Shown below is a block diagram of our system.



1.1 Summary – *Kiana & David*

The Course Front End Scheduler 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 and Administrative Assistants of the Ingram School of Engineering, and System Administrators.

1.2 Sponsor Requirements – *Kiana*

A key requirement outlined by the sponsor is to produce a system centred and 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].

1.3 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Currently, Program Coordinators use an outdated, hard-to-navigate Excel Spreadsheet to assign Instructors to courses.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. The existing method also lacks the functionality to be accessed as the most current version across all involved parties. 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.

1.4 Terminology – *Phillip*

HTML	Hypertext Markup Language – Language used to generate browser visuals and functions
CSS	Cascading Style Sheets – Language used for bookmarking html styles
JS	Javascript – Language used for clientside 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 Catalog
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	Javascript/Typescript web framework for encapsulating and generating front end modules and handling observables/injectables
Flask	Python web framework
SQLAlchemy	Python library for handling SQL type initialization and database queries

TLS	Transport Layer Security – Standard security protocol for providing communications over HTTPS

2 Functional Description

2.1 User Attributes and Use Cases – David

This system will have three types of users, the program coordinators 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
  2. User selects "Instructors" tab to view full list of Instructors
  3. User selects specific Instructor to add to a course
  4. User selects course from list to assign the Instructor to
  5. 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

2.2 Administration Functions – Kiana

The users of this application are Program Coordinators, Administrative Assistants, and System Administrators. Program Coordinators 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, back end 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.

2.3 Error Handling – Phillip

Errors in text inputs are handled automatically by HTML and the Javascript distribution being used in the program.

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. (Eg. 404 Page not found if the user attempts to access a page that does not exist or is currently inaccessible)

2.4 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.

Roll back & Undo?

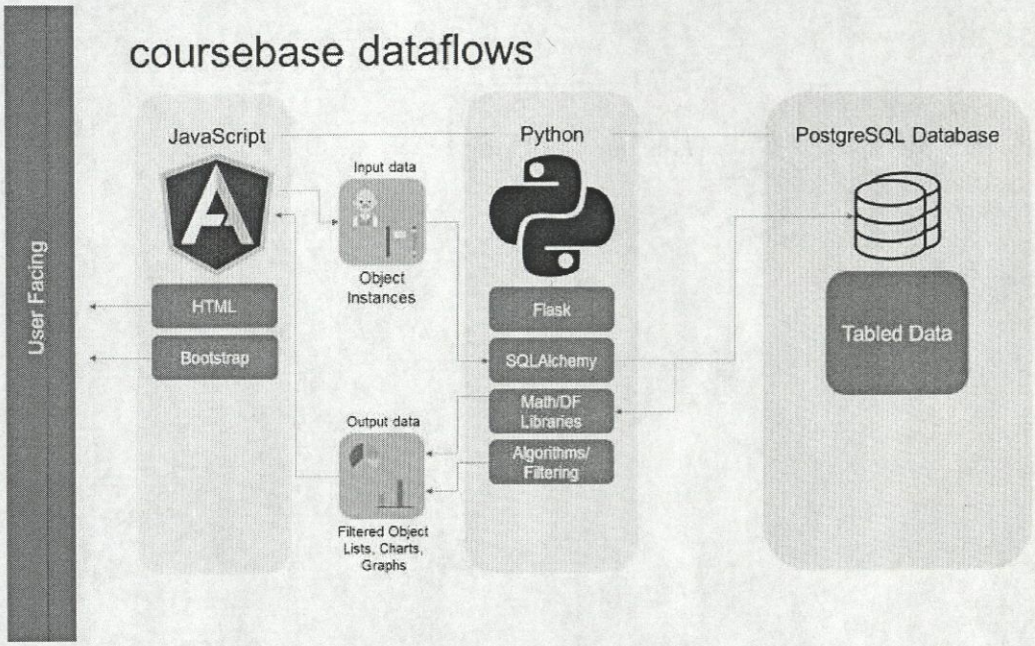
2.5 Help and User Documentation – Phillip

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 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

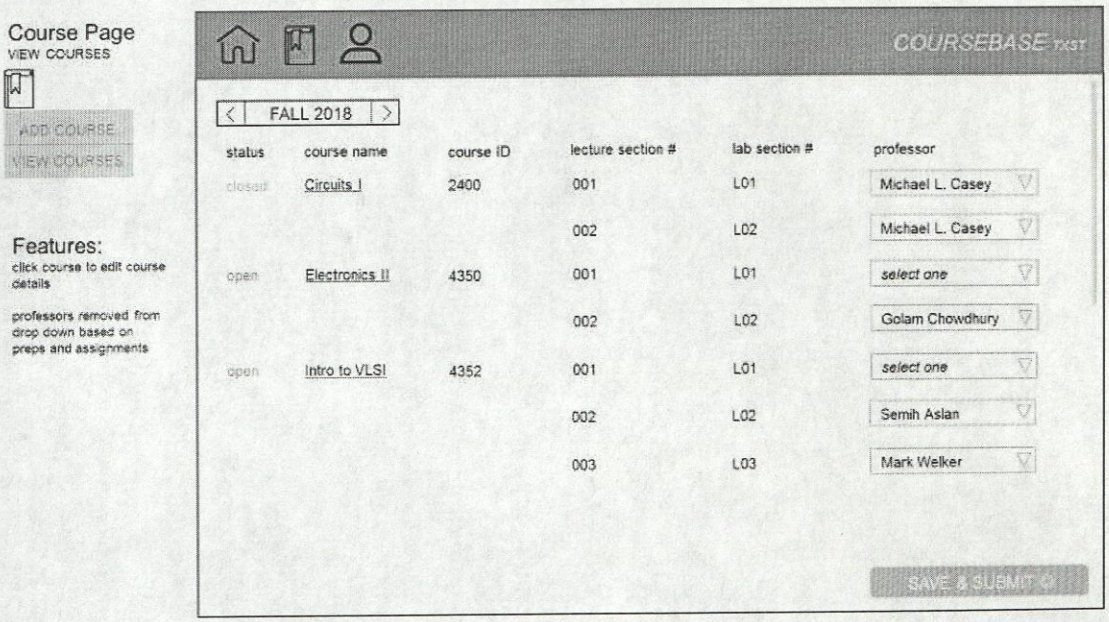
2.6 Interfaces

Shown below is a diagram showing 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.



2.6.1 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. Shown below is a mock-up of the "Courses" page of the frontend.



2.6.2 Software – David

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
- AngularCLI
- Python
- Flask
- SQLAlchemy
- PostgreSQL

Versions?

2.6.3 Hardware – Phillip

The project is purely software, however, regarding hardware, the application will require a server to run on.

Recommended System Requirements:

Processor: Quad Cortex A53 @ 1.2 Ghz (or equivalent)

Memory: 1GB SDRam

Storage: 1GB

Network: An ethernet or wi fi connection

2.6.4 Mechanical N/A

2.7 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 fairly strict policies on what programs are allowed to 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.

2.8 Performance – Phillip

This is a web application. However, to remove the variability of internet bandwidth and response time, response rates of the application will be tested locally. All of these tests will be done for every planned live release.

Performance Parameters		
Function	**Description**	**How Tested**
Releases	When a release is pushed from the development server to the live server, automatic updating should occur.	Updates will be pushed and checked to see if live server portrays updates correctly.
User interface	All pages load within a response time of 1ms	Response time will be measured in Chrome's Network Tool.
Memory usage	Memory Usage Limit will be 500MB, but we expect a much lower usage.	Memory Usage will be measured in Linux, using 'ps' and 'valgrind', Task Manager in Windows
User Input	User Input should populate in the correct formats and fields in the database	Test front end input forms, check database field and tables
Error Dialogs	Correct error dialogs should generate when a user assigns an Instructor to a course outside of Instructor parameters	Create error, check dialogs, majority will be QA tested in Alpha release
Security	There will be a login management system for the application. Data should also be encrypted using HTTPS protocol using TLS.	Check datastream with wireshark.
Platforms	The system will run on the following computer platforms: (a) Windows 10 (b) Windows 7 (c) OSX (d) Linux	Test on all these operating systems. This has already been done for initial development.