Design Document for Kairos Constraint-Based Scheduling Software System

Tyler Chapman, Nate Crandall, Vinh Dang, Vince Oveson, Tony Tuttle

November 7, 2014

Contents

List of Figures

1 Executive Summary

1.1 Overview

The final goal of the project is to provide a highly customizable, open-source, web-based scheduling tool available to solve a wide range of scheduling problems. Our software system will be accessible through a public website where users may build, modify, and maintain their solutions. The system will be capable of solving various types of scheduling problems for various types of user needs.

Scheduling problems are ubiquitous. Individuals, teams, organizations, and larger entities such as companies all must solve scheduling problems of various levels of complexity. Our tool aims to address the needs of such a wide base of potential users.

1.2 Features and Components

At its core, Kairos will be a web-based schedule solver; it will accept parameters from the user, specifying the details of their particular scheduling problem. The tool will analyze the input and algorithmically determine a schedule that will meet all of the supplied parameters. If meeting all of the constraints is not possible, it will prioritize based on weights and determine what compromises to make in the schedule.

Since the tool will be web-based, it will be open to any and all potential users. This will further encourage a wide breadth of users.

We intend to make the tool extremely customizable by creating a very general schedule solver core. We want users to have access to the powerful core components while maintaining sufficient flexibility to fit the solution to their specific needs.

Part of this customizability will come from working hard to make the API as clear and thorough as possible. A great piece of software may lose potential users if it is not clear to users how best to leverage the software. We intend to encourage a large user base by putting a lot of emphasis on creating a strong API.

Likewise, a great piece of software that lacks intuitiveness or a pleasing user experience will alienate users. We will put a great deal of thought and planning into determining how best to use visualization tools to represent our data. Since scheduling is a complex problem that produces data that will need to be viewed from several angles, this is a difficult problem in itself. By making visualization a priority we hope to attract users as opposed to driving them away.

1.3 Justification

Making this tool available to the public will potentially save individuals a great deal of time, provide organizations better scheduling solutions than they currently have, and save a great deal of money for businesses and other organizations. At the very least, it is our hope that we will make life a little easier for as many users as possible.

2 Background

2.1 Overview

The task of creating optimal schedules is a problem that event planners, groups, teams, and other organizations of all sizes encounter on a regular basis. The schedules used by these groups are often created manually. For all but the simplest schedules, this task is extremely complicated and time consuming. As the complexity increases, the difficulty of creating acceptable schedules, much less optimal ones, increases exponentially.

Moving as much of the work of scheduling from a manual process to one that is solved by a computer is the inspiration for Kairos. Kairos has the potential to save a great deal of time for users. Kairos will provide a better solution for those who need to schedule any but the simplest of events.

Since scheduling is a ubiquitous problem, Kairos will be usable to large number of users. By implementing a well-documented API we intend to attract those who seek a solution that they can tweak to fit their specific needs.

A specific use case that we have identified is the School of Computing at the University of Utah. We are working with staff in the department to provide a solution to their problem of scheduling classes each semester.

2.1.1 Similar Ideas

We have identified two software systems that address the scheduling problem in some manner similar to that which we are aiming at. These systems are Aurora Intelligent Planning and Scheduling System (Aurora) and Microsoft Project (MS Project). To be sure, there are other software systems out there, but these are adequate to represent the current state of this space.

2.1.2 How Kairos is Different

Based on their literature, Aurora addresses the needs that we are attempting to address more than adequately. However, they are focused on supplying solutions to very large organizations with correspondingly very large scheduling problems. We would like to address the needs of the smaller users for which the Aurora software would be overkill.

While MS Project does include scheduling tools in the software package, the larger goal of MS Project is to provide project management software. Thus, their software is attempting to address a problem with a wider scope than we are focused on. Based on reading the literature on their software, it is not clear that the scheduling tools provided in MS Project are constraint-based scheduling tools.

Both Aurora and MS Project cost money. Our software will be free. Neither Aurora or MS Project are web-based, but require installation on any machine that will use the software. There is certainly room for our tool in this space.

2.2 Required Technology

Our project can be broken down into three main components: a core web service, specific modules that connect to it, and an API that handles communication between the two. We will discuss the technologies we plan to utilize in developing each of these three components.

2.2.1 Core Web Service

This is where the logic for the schedule solver will live. This web service will be centered around the transfer of data through requests and responses. Our web service will use API keys to restrict access. In order to authenticate users, we will store these keys in a MySQL database. This will make our web service more secure.

2.2.2 SoC Module

We will create a module for the School of Computing that connects to our web service. This module will be a web application that allows the user to specify the events (classes) to be scheduled, the resources to be used (professors, rooms, etc.), and the constraints for both classes and resources. When the web service sends back a suggested schedule, the user will be provided with different options for viewing the data. We also plan to provide some web scraping tools in order to facilitate the collection of data relevant to scheduling classes.

This module with have a PHP backend, along with a MySQL database. The front end will be built using HTML, JavaScript, and CSS. We will utilize jQuery (specifcally ajax) to make asynchronous requests to the web service without interrupting the user experience.

2.2.3 API

Our API will use JSON to represent the data being shuttled back and forth between the web service and the modules which connect to it. We plan to provide detailed documentation that will help other developers leverage our service in their own applications. To make this documentation readily available and accessible, we will create a website where we will publish the API documentation, and also advertise our service. This site will also be where developers can request an API key.

2.3 Assets and Engines

The websites we create – both the API documentation site and the School of Computing module – will be built using the Laravel Artisan PHP framework. This will provide structure to our web sites, as well as allow us to keep our code organized and concise. Laravel also includes a command line interface which will provide us with shortcuts for common tasks, thus allowing us to focus our efforts on more important development tasks.

2.4 Software/Hardware Requirements for Users

Since Kairos will be a web application with all of the intensive computation taking place on our network, the only system requirements for users will be a machine running a modern browser with a reasonably fast internet connection.

3 Requirements Analysis

3.1 System Architecture

3.1.1 Constraint Solver

At the core of our system will be the constraint solver. The constraint solver is the software that solves general constraint-based scheduling problems. The solver will receive the parameters of a problem instance from the app server and will return a solution back to the app server if one is discovered. The solver will be written in Java and will make use of one or more constraint programming libraries.

A problem instance will be supplied to the solver as a set of activities, a set of resources, a set of constraints, and some objective function. The solver will attempt to fulfill the objective function given the resources and constraints and, if successful, will return a solution to the app server. If it is not possible to fulfill the objective function, a message indicating such will be returned.

3.1.2 App Server

The app server will be the means by which the constraint solver communicates with its users. These users will be one of two types. The first type is the API user. API users will use the solver to solve general scheduling problems. They are converting their concrete scheduling problems into instances of general constraint problems and providing those problem details to the solver. When they receive a solution they will convert it back in order to apply it to their specific scheduling problem.

The second potential user of the app server will be our own Apache server. Similarly to the API user, the problem will be supplied from the Apache server as a general constraint problem. The difference is that these problems will be generated from our implementation of the class scheduler for University of Utah classes.

3.1.3 Apache Server

The Apache server will serve our API documentation website as well as the University of Utah class scheduler site, and the School of Computing class scheduler tool. The API documentation site will be the reference for our API users. It will include all of documentation necessary in order for API users to leverage our solver. It will provide detailed explanations of the methods available as well as code examples.

The University of Utah class scheduler site will presently be a site that will be set up such that departments at the University will be easily able to begin using our scheduling tool. Additionally, it will be extended to fit the specific needs of our primary test user, the School of Computing.

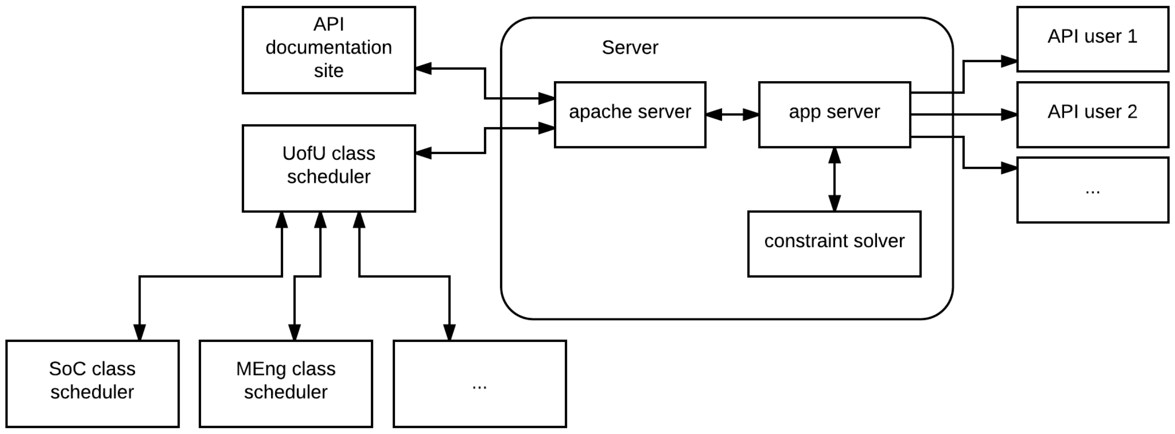


Figure 1: Overview of the System

3.2 Personnel

We have five team members that will work together to implement this project in its entirety. The members are Tyler Chapman, Nate Crandall, Vinh Dang, Vince Oveson, and Tony Tuttle.

Tyler's primary responsibilities will be tied to ensuring that the network functions correctly and securely. As an extension of network responsibilities, Tyler will be largely responsible for implementing the server back-end.

Nate's primary responsibility will be to develop the visualization component of the project. He will be working closely with the front-end developer and will use his knowledge of visualization to make sure that our the user interface will maximize usability and aethetic appeal. He will also be responsible for designing and implementing the necessary databases.

Vinh's main responsiblity will be implementing tools that are specific to the School of Computing portion of this project. They will need some additional tools such as web scrapers and other peripheral software in order to meet their needs. Vinh will also assist in writing the constraint solver.

Vince's primary responsibility will be designing the UIs and front-end development. He will use his extensive background in web applications to make sure that all of the front-end components function correctly and provide a pleasant experience for the user. He will also be responsible for design and testing of the overall system architecture, ensuring that all of the parts are working together correctly.

Finally, Tony's main responsibility will be designing and writing the constraint solver. He will use his knowledge and interest in algorithms and AI to research the best solution and implement it. As a natural extension to this responsibility, he will also be writing the API documentation.

3.3 System Features

Appendix A Use Cases

A.1 Entering Data Manually to Create a New Schedule

The user will manually enter the data required for the system to generate a proposed schedule. The steps for this use case are:

1. User navigates to Manual Data Entry page
2. System prompts user to enter classes
3. User enters classes and selects 'Next'
4. System prompts user to enter rooms
5. User enters rooms and selects 'Next'
6. System prompts user to enter professors
7. User enters professors and selects 'Next'
8. System prompts user to enter constraints on room availabilities
9. User enters constraints on room availabilities and selects 'Next'
10. System prompts user to enter constraints on professor availabilities
11. User enters constraints on professor availabilities and selects 'Create Schedule'
12. System calculates and returns a proposed schedule to user

See the related Figure 4 on page 11

A.2 Uploading a CSV File to Create a New Schedule

The user will generate a proposed schedule by uploading the required data via a .csv file. The steps for this use case are:

1. User prepares spreadsheet for upload
2. User navigates to CSV Import page
3. System prompts user to enter path to file for upload
4. User enters path and selects 'Upload'
5. System attempts to parse scheduling data from file
    (a) If the parse fails
        i. Failure is minimal
            A. System prompts user to adjust data
            B. User adjusts data
            C. Return to [5] above
        ii. Failure is large
            A. System informs user of error
            B. System returns user to Dashboard page
6. System calculates and returns a proposed schedule to user

See the related Figure 8 on page 14

A.3 Exporting Schedule as a CSV File

The user will download a local copy of a proposed or modified schedule as a .csv file. The steps for this use case are:

1. User navigates to one of the View pages on an existing schedule
2. User clicks Export to CSV option
3. Browser prompts user to accept download
4. Browser begins download of CSV file

A.4 Modifying a Proposed Schedule

The user will make modifications to an existing schedule. They will change the parameters of a given class in order to change some feature of the class. The steps for this use case are:

1. User navigates to one of the View pages on an existing schedule
2. User right clicks on an instance of a class and selects 'Modify This Class'
3. System displays the resources and constraints for that class and prompts user to modify as needed
4. User modifies the resources/constraints and selects 'Modify Schedule'
5. System calculates and returns a proposed schedule with the modifications to user

See the related Figure 4 on page 11 and Figure 5 on page 12

A.5 Comparing Two Schedules

The user will select two existing schedules for comparison. The system will display the discrepancies between the two schedules. The steps for this use case are:

1. User creates and saves at least two schedules
2. User navigates to the dashboard page
3. User selects 'Compare Two Schedules'
4. System prompts user to select two schedules for comparison
5. User selects two schedules and selects 'Compare'
6. System calculates the differences between the schedules and displays them simultaneously, highlighting the differences

See the related Figure 9 on page 15

A.6 Administrative Tasks

The schedule administrator will be able to load, delete, rename, update users who can view the schedule, and accept/reject changes made by other users of the schedule.

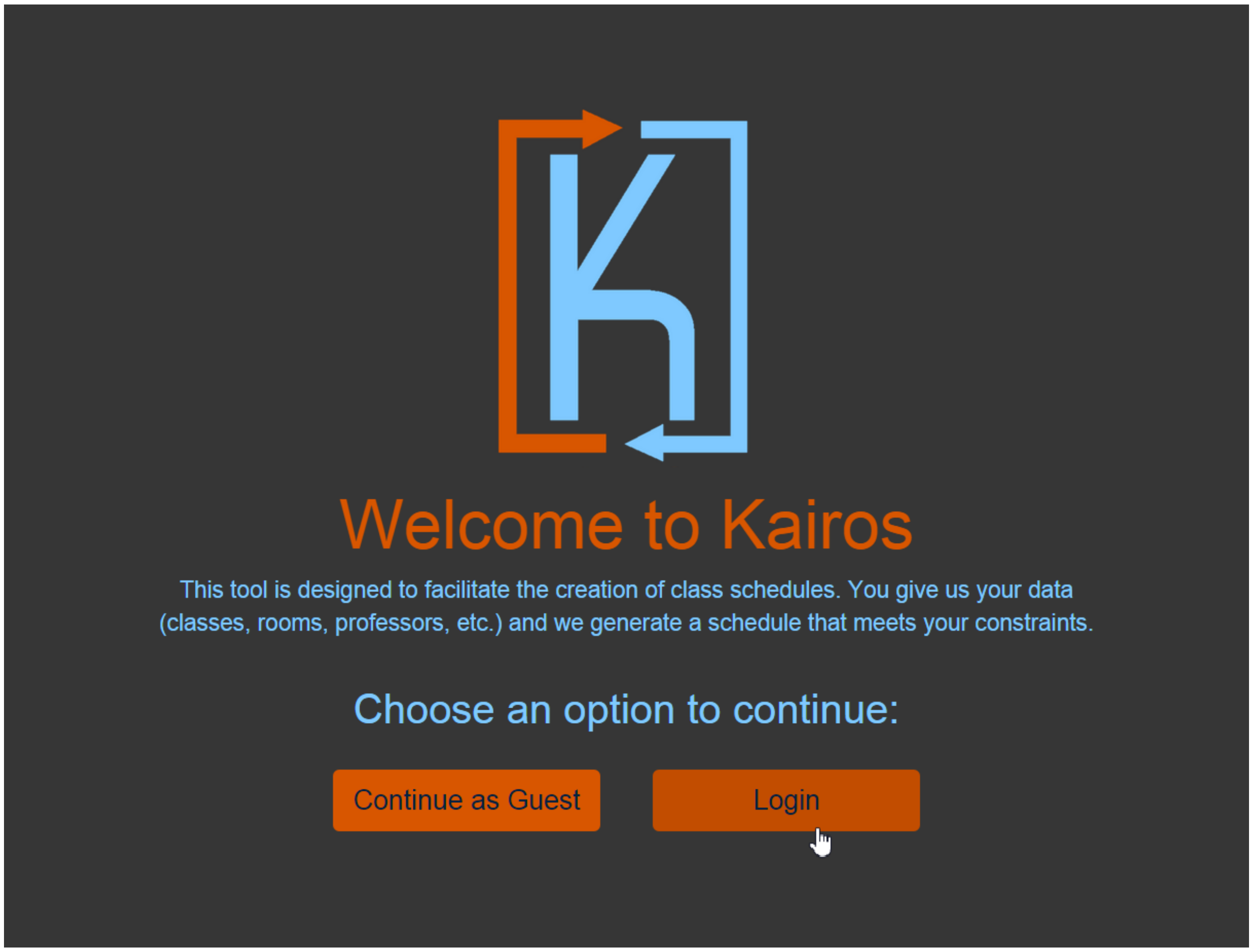
1. User creates and saves a schedule
2. User logs in to the system
3. User goes to the dashboard
4. User selects the schedule(s) that need to be updated.
5. The user can then select their administrative task(i.e. load, delete) on the right sidebar

A.7 API use

The third party developer will request a key and create their own implementation of a constraint based scheduling system using our api.

1. The developer will request an api key
2. The developer will read the api website to learn how to use our api
3. The developer will send the request he needs
4. The system will return json data that the developer can parse and use

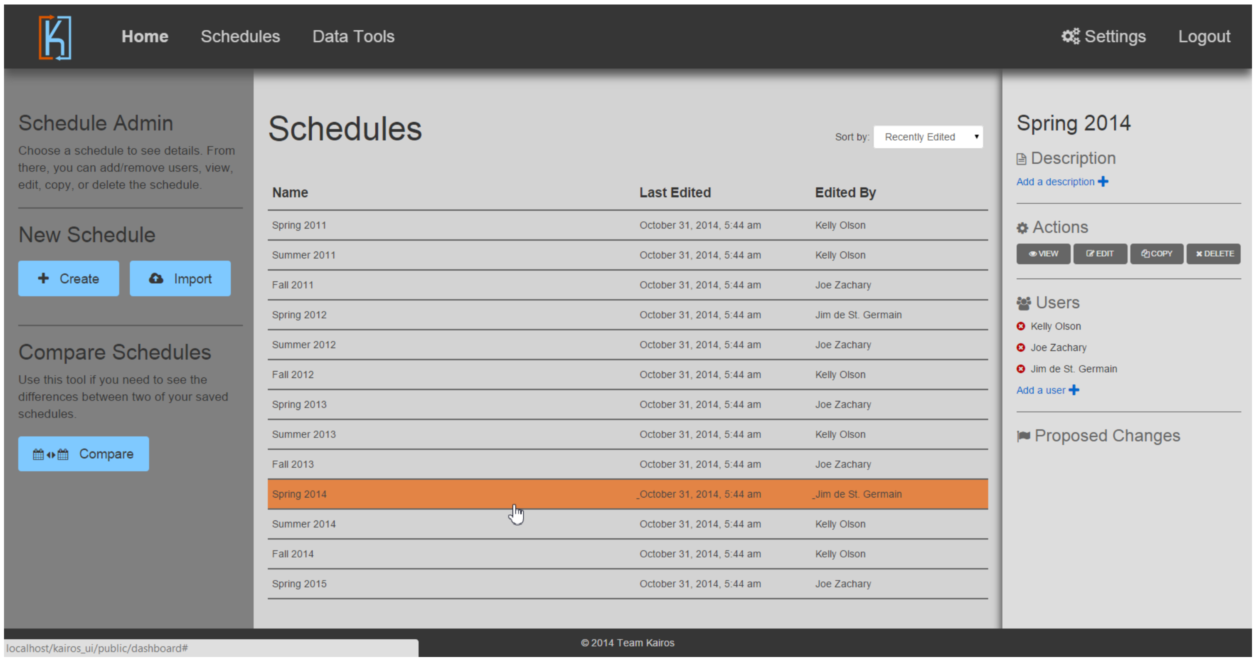
Appendix B UI Sketches


Welcome to Kairos

This tool is designed to facilitate the creation of class schedules. You give us your data (classes, rooms, professors, etc.) and we generate a schedule that meets your constraints.

Choose an option to continue:

Figure 2: Kairos landing screen



Schedule Admin

Choose a schedule to see details. From there, you can add/remove users, view, edit, copy, or delete the schedule.

New Schedule

Compare Schedules

Use this tool if you need to see the differences between two of your saved schedules.

Schedules

Sort by: Recently Edited

Name	Last Edited	Edited By
Spring 2011	October 31, 2014, 5:44 am	Kelly Olson
Summer 2011	October 31, 2014, 5:44 am	Kelly Olson
Fall 2011	October 31, 2014, 5:44 am	Joe Zachary
Spring 2012	October 31, 2014, 5:44 am	Jim de St. Germain
Summer 2012	October 31, 2014, 5:44 am	Joe Zachary
Fall 2012	October 31, 2014, 5:44 am	Kelly Olson
Spring 2013	October 31, 2014, 5:44 am	Joe Zachary
Summer 2013	October 31, 2014, 5:44 am	Kelly Olson
Fall 2013	October 31, 2014, 5:44 am	Joe Zachary
Spring 2014	October 31, 2014, 5:44 am	Jim de St. Germain
Summer 2014	October 31, 2014, 5:44 am	Kelly Olson
Fall 2014	October 31, 2014, 5:44 am	Kelly Olson
Spring 2015	October 31, 2014, 5:44 am	Joe Zachary

Spring 2014

Description

Add a description

Actions

Users

Kelly Olson

Joe Zachary

Jim de St. Germain

Add a user

Proposed Changes

localhost/kairos_ui/public/dashboard#

© 2014 Team Kairos

Figure 3: User schedule selection dashboard

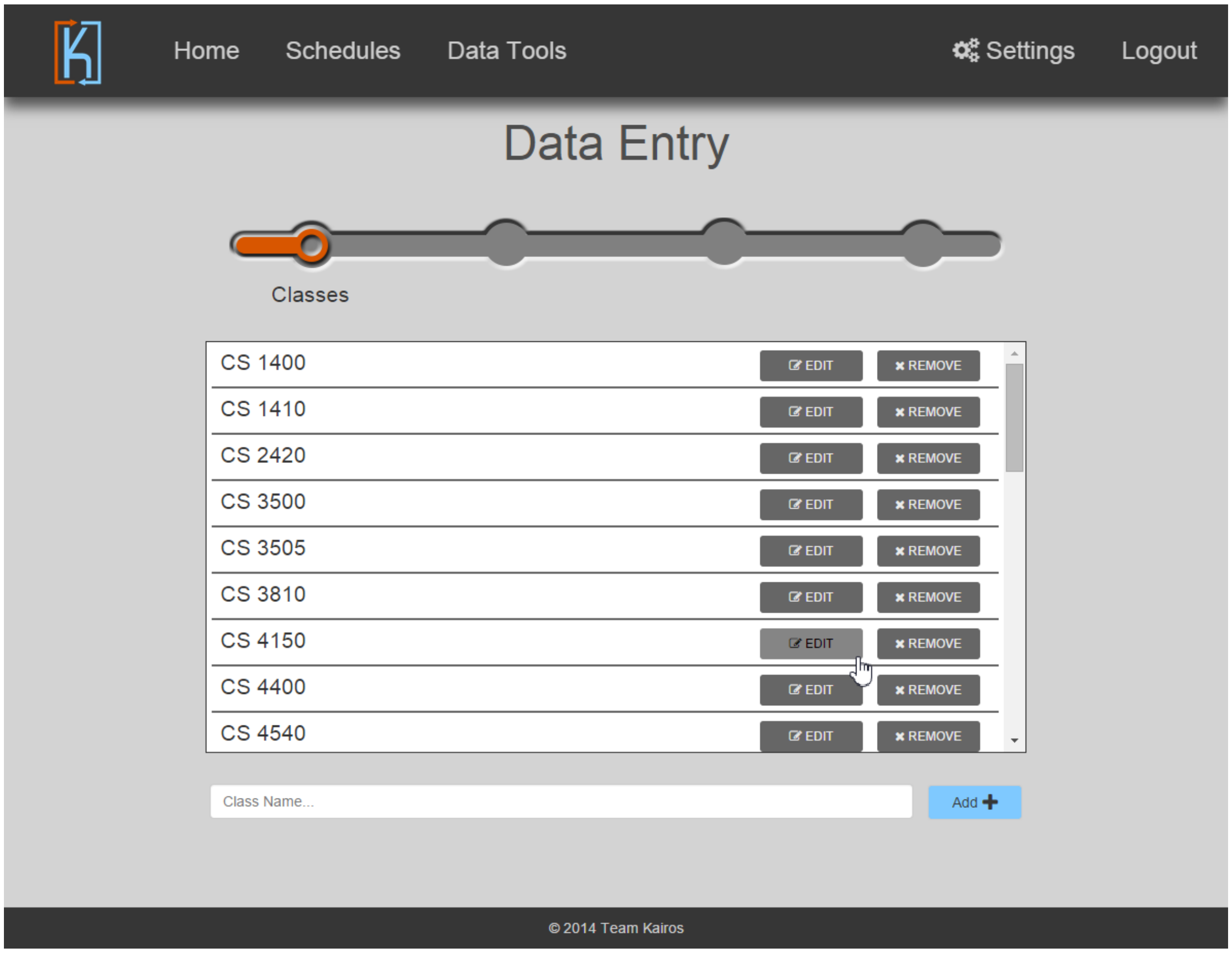
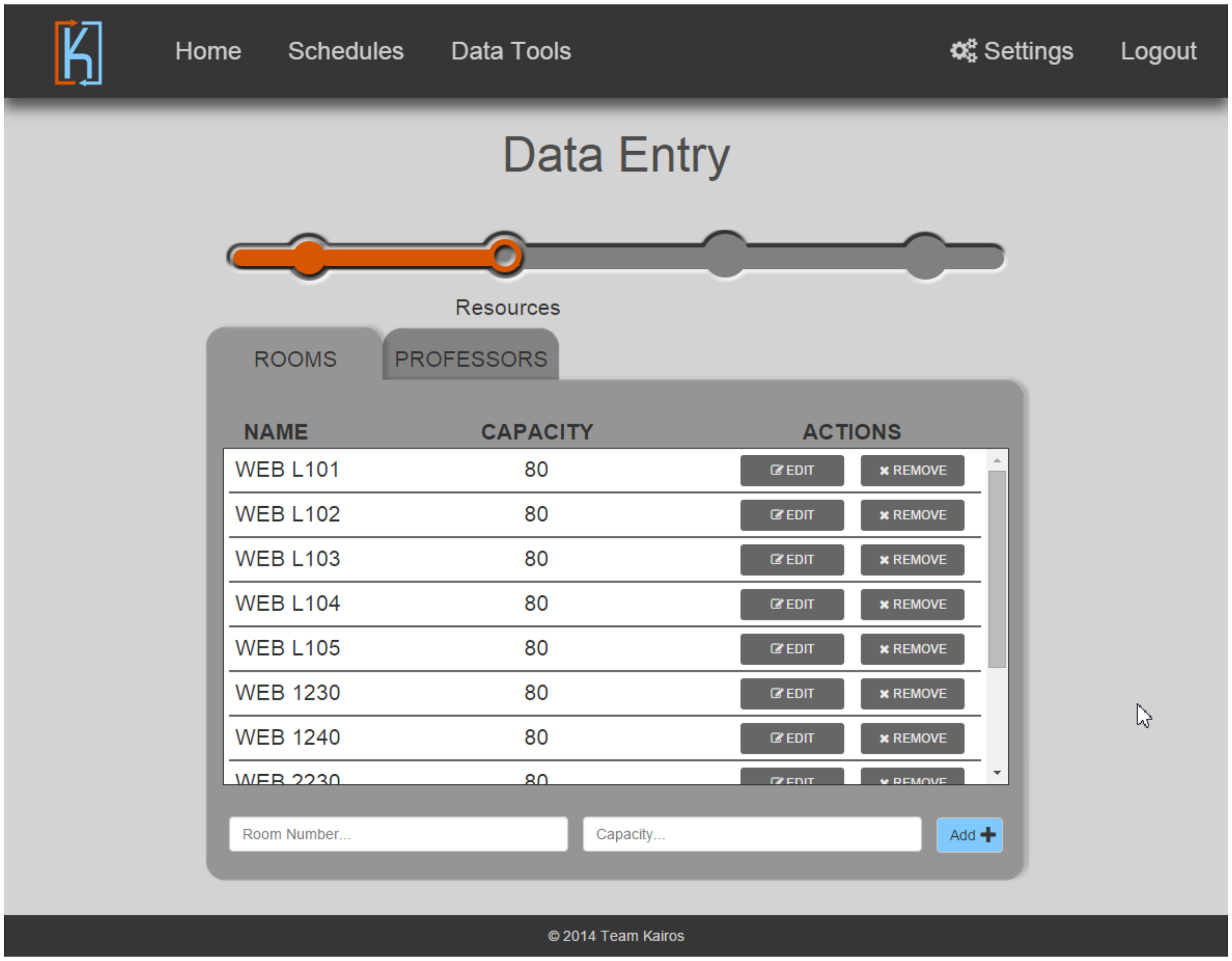


Figure 4: User data entry screen for classes



NAME	CAPACITY	ACTIONS
WEB L101	80	EDIT REMOVE
WEB L102	80	EDIT REMOVE
WEB L103	80	EDIT REMOVE
WEB L104	80	EDIT REMOVE
WEB L105	80	EDIT REMOVE
WEB 1230	80	EDIT REMOVE
WEB 1240	80	EDIT REMOVE
WEB 2230	80	EDIT REMOVE

Figure 5: User data entry screen for rooms

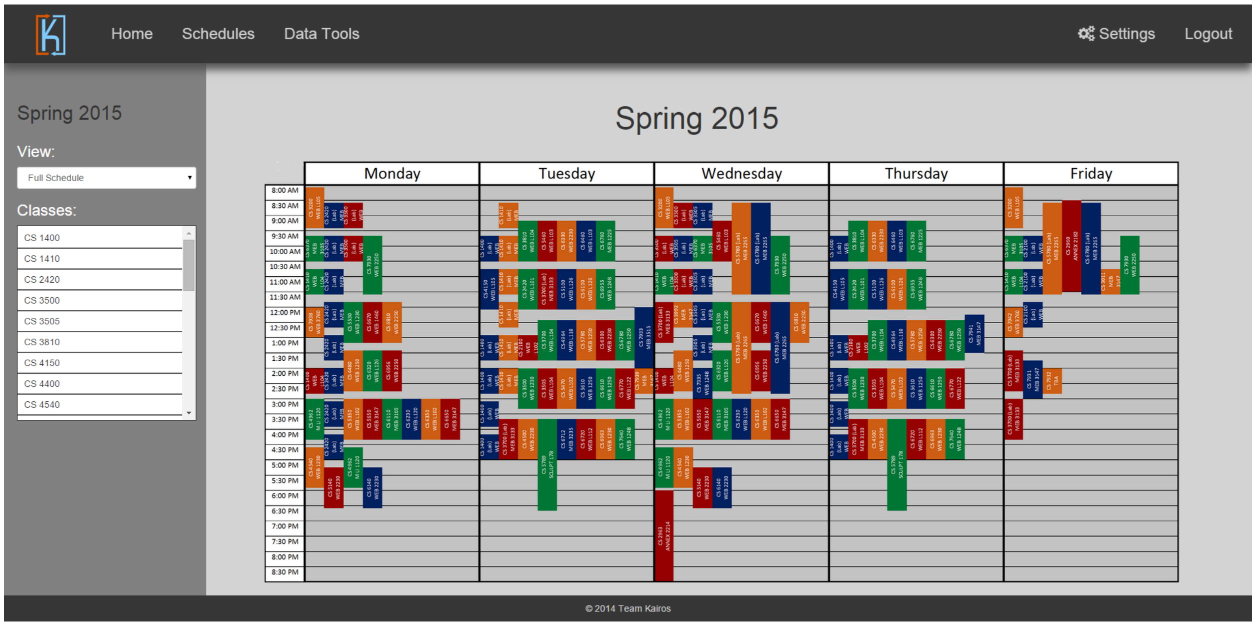


Figure 6: Visualization of a schedule

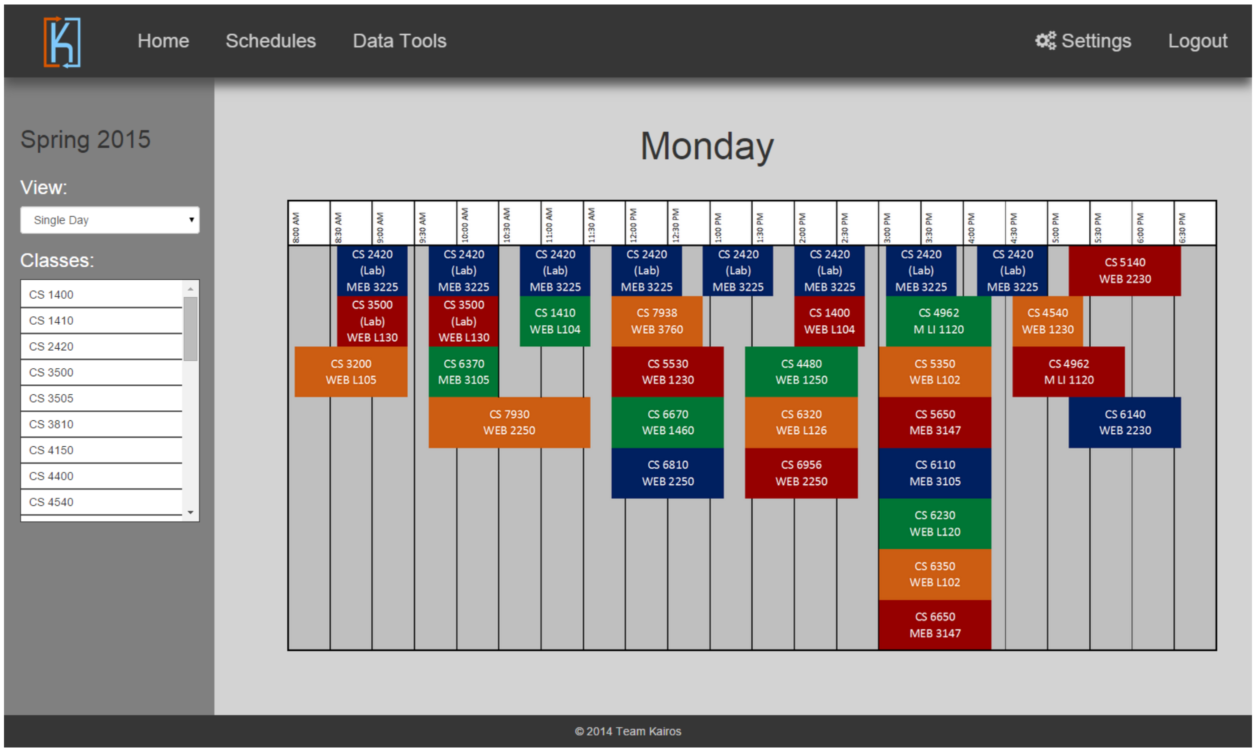


Figure 7: Single day visualization of a schedule

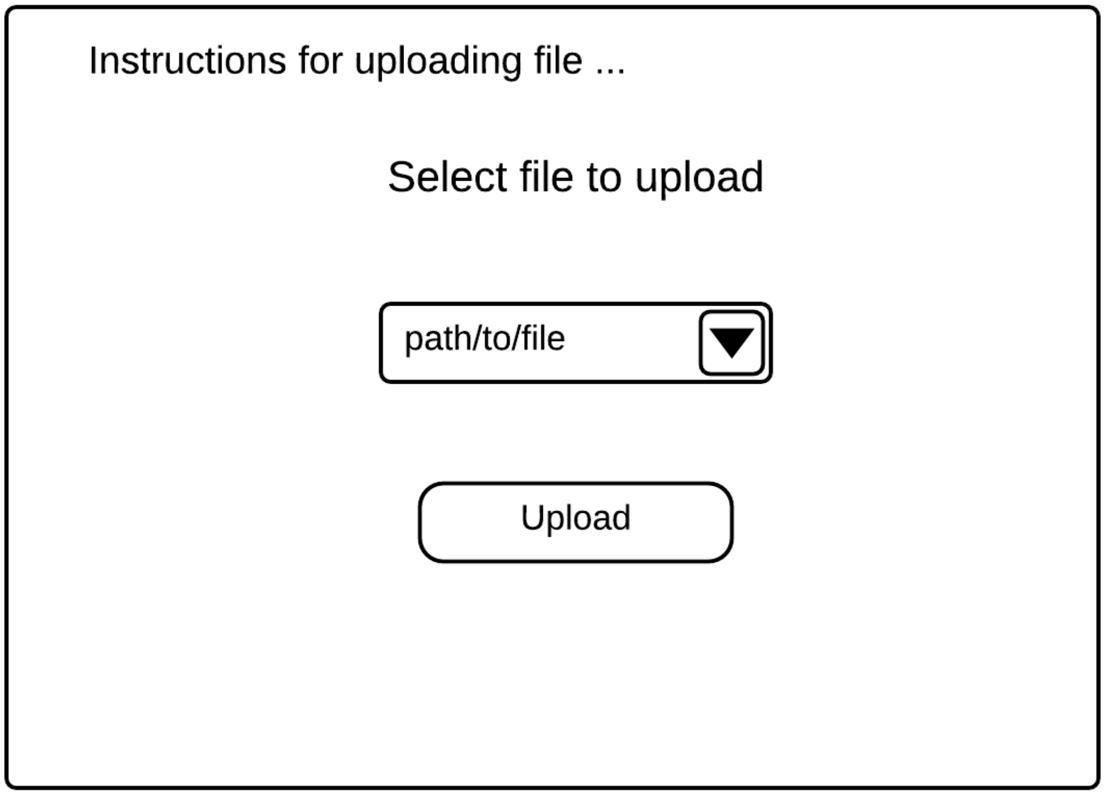


Figure 8: Wireframe of UI for user to upload a CSV file

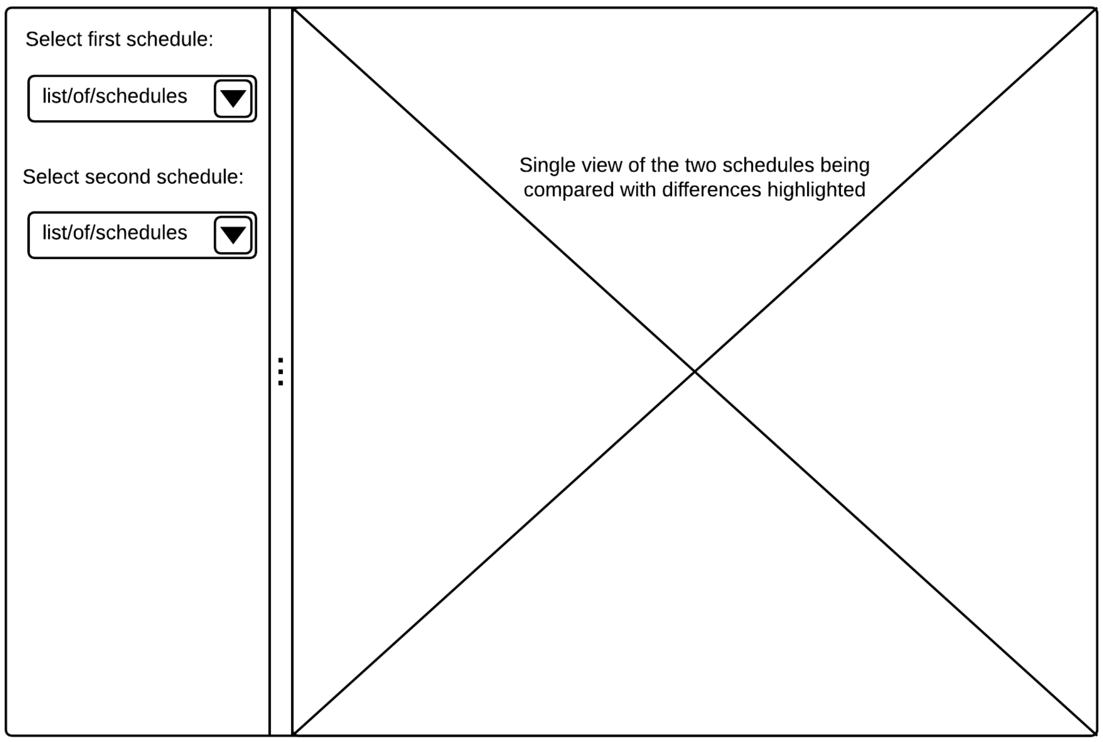


Figure 9: Wireframe of UI for user to view differences between two schedules