Team United Airlines
Michigan State University
Training Scheduling and Optimization System II
Project Plan
Fall 2019

United Airlines Contacts

Amadou Anne
Craig Bennett
Rick Brown
Jaime Hill
Lynda McDaniel
Tom Wilson

MSU Capstone Members

Laura Danila
Andrew Ferguson
Josh Pezeshki
Jack Soenke

Table of Contents

Executive Summary

United Airlines is a major American airline that is headquartered in Chicago, Illinois. They operate both domestic and international flights from over 350 airports and have seven hubs. They operate around 4,600 flights a day using a fleet of 1,300 aircrafts. To be able to operate these flights, the airplanes need to be properly maintained. This requires having adequately trained maintenance personal. The United Technical Operations division currently has a staff of 45 instructors; they deliver 700 courses a year to over 7,000 students from a catalog of 90+ courses.

The current process has been used for multiple years and involves manually scheduling these classes. Currently there is one person that goes in and looks at spreadsheets to find when and where classes can be scheduled, and they upload that information to multiple websites. This process needs upgrading, as it is time-consuming and easily susceptible to human error. It can also lead to suboptimal use of instructor time.

Using MSU Team United Airlines spring 2019's project as a baseline, the fall 2019 team will complete a web application to schedule maintenance instructors, classes, and locations. The new application will be able to optimize a schedule given a list of available rooms and instructors, and it will then have to be approved by a human. Using the current system, scheduling a single class requires extensive expertise of the system and classes, multiple application and web pages open from several sources, and in total takes over 15 minutes. Adding that same class to the schedule in the new web application can be done in 2 minutes from a single website with basic knowledge of the system. This application drastically speeds up the scheduling process while saving both user time and training time to use this system.

Functional Specifications

The current system is comprised of multiple applications and websites, forcing data and tasks necessary to schedule courses to be scattered across multiple locations. This setup requires thorough manual checking to ensure consistency between the different subsystems. However, any conflicting schedule information cannot be automatically detected as there is no centralized system. This problem is solved by the all-in-one approach of the new system. It is more user friendly and helps save time by having all the information needed consolidated into one website.

The main goal of this project is a schedule optimizer. A scheduler will input a list of classes, locations and start time, and with one button, the application creates an optimal schedule. Within the given time frame, each class will be assigned a classroom location and an instructor. It needs to be able to filter by different attributes, such as different room features, or if the instructor has scheduled time off. The optimizer will also prioritize different things, such as teaching a class with a local instructor and assigning classes earlier within the time frame. This helps avoid having instructors have unnecessary travel, which saves both time and money.

Additionally, instructors will be able to view a calendar of all their scheduled classes and activities recorded (ex. vacation time or travel time), while guests will be able to view the overall class schedule. Non-guest users will be able to request courses. The application also has other features to automate work. For example, the updated application will include the ability to assign/edit a class using a drag-and-drop calendar and a selector to enable/disable weekend days when scheduling classes. There are also more notifications that alerts users if they are trying to manually add classes that create conflicts. This portal will simplify the process for everyone involved.

All features will be available on both the website and a mobile version of the website. This has already been started by the Spring 2019 team, and our team will continue to build upon and improve this. While the website will be useful for some of the larger, more time intensive aspects of scheduling, having a mobile version allows for easier changes and on-the-go use. Additionally, the new app will now be available in mobile browsers, and work in IE, Edge, Firefox, Chrome, and Safari on desktop.

Design Specifications

Overview

Training Scheduling and Optimization System II is an application designed to help schedule and organize classes for United's training system. With the website supporting both desktop, tablet, and smartphone screen sizes, the user can easily schedule a class with an instructor and attendees, as well as modify any existing classes that haven't started yet. With the optimizer, the user selects which classes they want to schedule and the time frame, and the optimizer then figures out what the best class schedule is for that range.

User Interface (UI)

Calendar Page

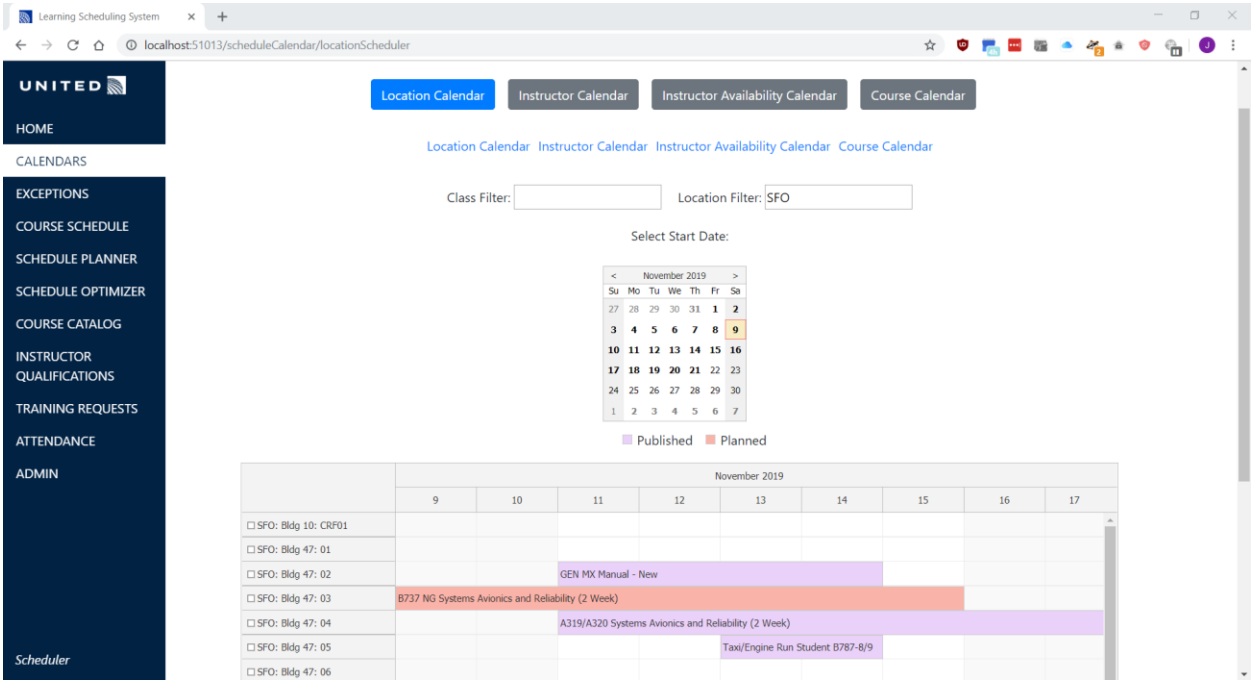

Figure 1: Calendar Type Selection, Filters, and Start Date Selector

Seen above in Figure 1, the calendar page view is where a user can see the current published schedule as well as the planned schedule if they have access rights to the schedule planner. The schedule information can be seen by either the location, instructor, monthly, or course calendars. Additionally, users can filter the classes shown on each calendar by typing parameters like "SFO" into the location filter and the calendar results will be restricted to classes only occurring at the San Francisco facility. The user can also look at previous and upcoming classes easily by selecting a start date.

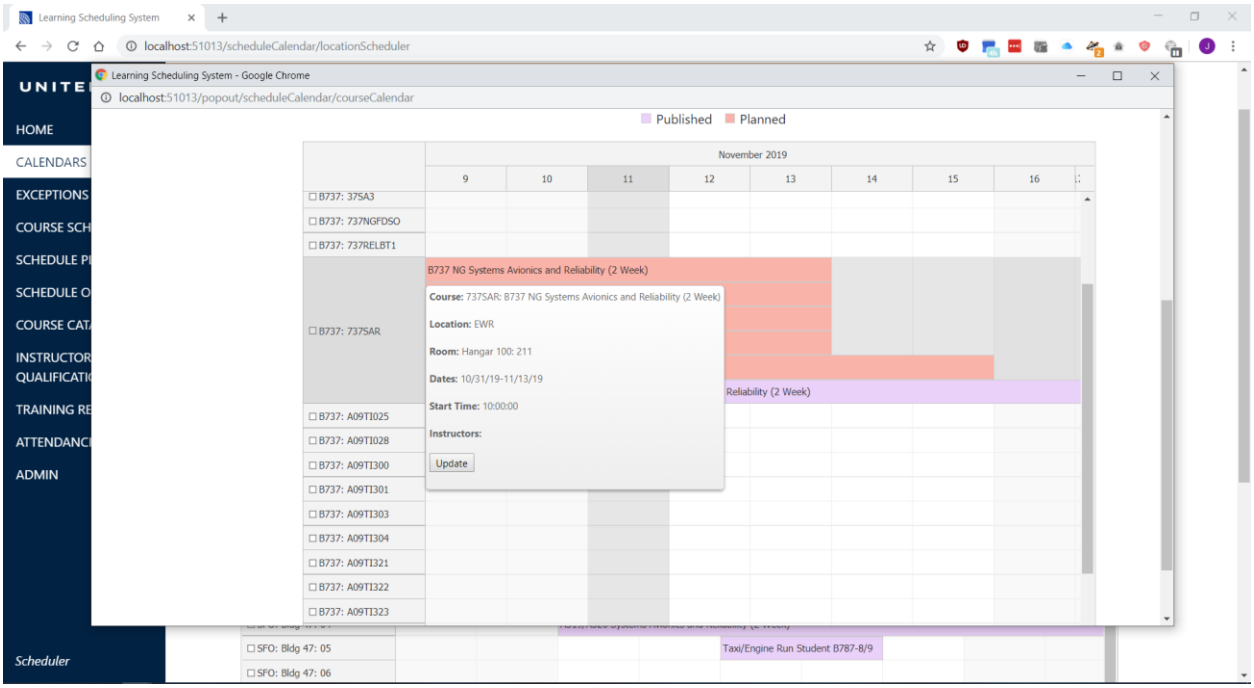

Figure 2: Pop-out Course Calendar

As can be seen in Figure 2, users can also view the calendars from a pop-out window. From the current view as a scheduler user, planned courses can be seen in a light red color and published courses can be seen in a purple color. When hovering over a class, the details specific to that class are displayed (class, location, instructor, start time, classroom). Additionally, if the user selects the update button from this hover over window, they will be able to conveniently and quickly make a change to the scheduled class.

Schedule Optimizer Page

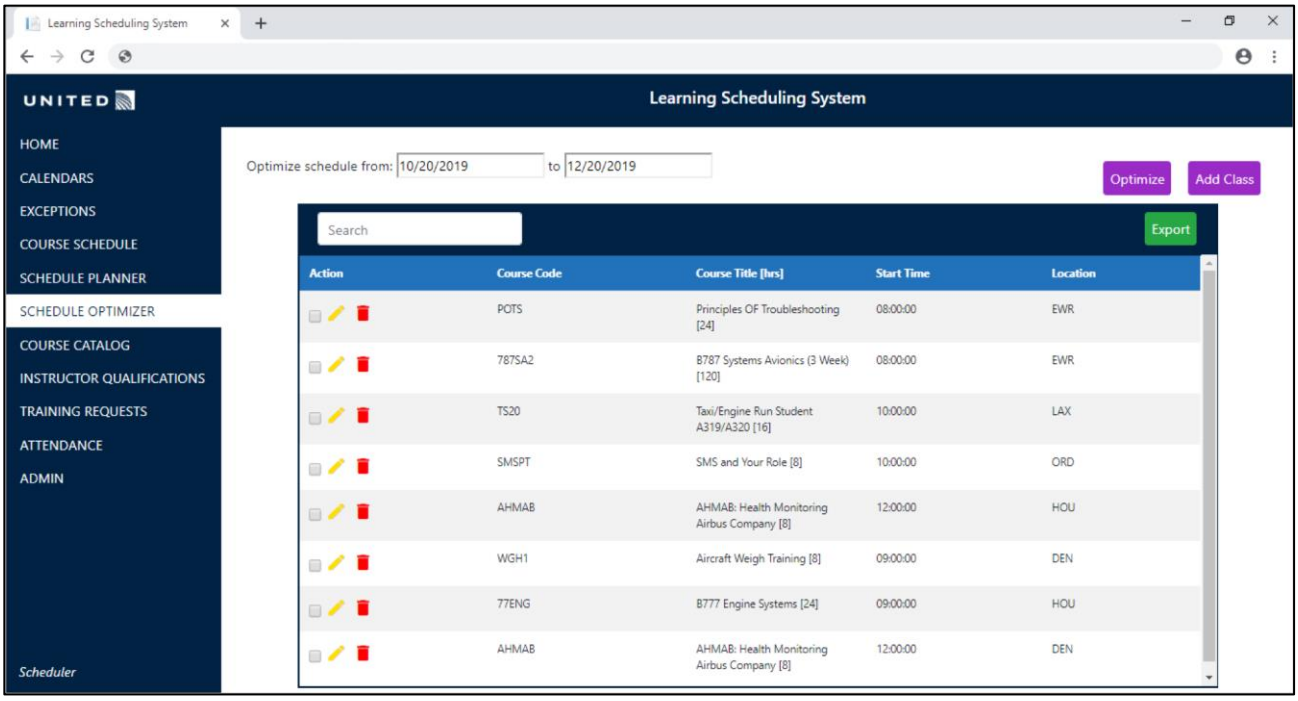

Figure 3: Scheduler Optimizer Page

Figure 3 shows the page view for the schedule optimizer. When the scheduler adds a class, it will appear on the list; it can be edited or deleted later from the action buttons. The scheduler will need to input a start and end date to run the optimizer on the classes listed below and when ready will hit the purple optimize button, which will proceed to a new screen with the results.

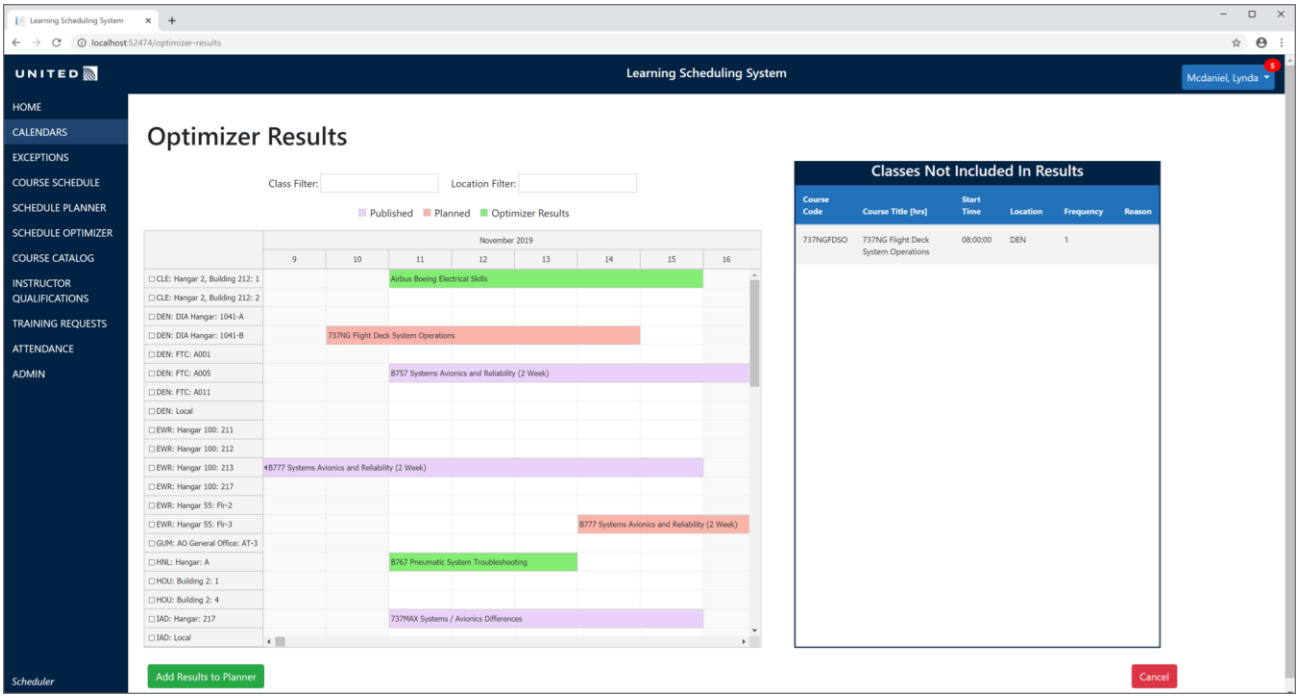

Figure 4: Optimized Schedule Page

Figure 4 shows the page view for the optimizer results screen. The screen includes a calendar, a table of classes not able to be optimized, and a results table. The calendar is like the course calendar shown in Figure 2, but it also includes results from the optimizer in green. When the screen first appears, the user sees the above image, and if they scroll down, they will see the results table, which includes all classes able to be optimized. The user will be able to select which optimized classes they want to add to the schedule planner, and then hit a button to add them.

Resource and Schedule Conflict Alerts

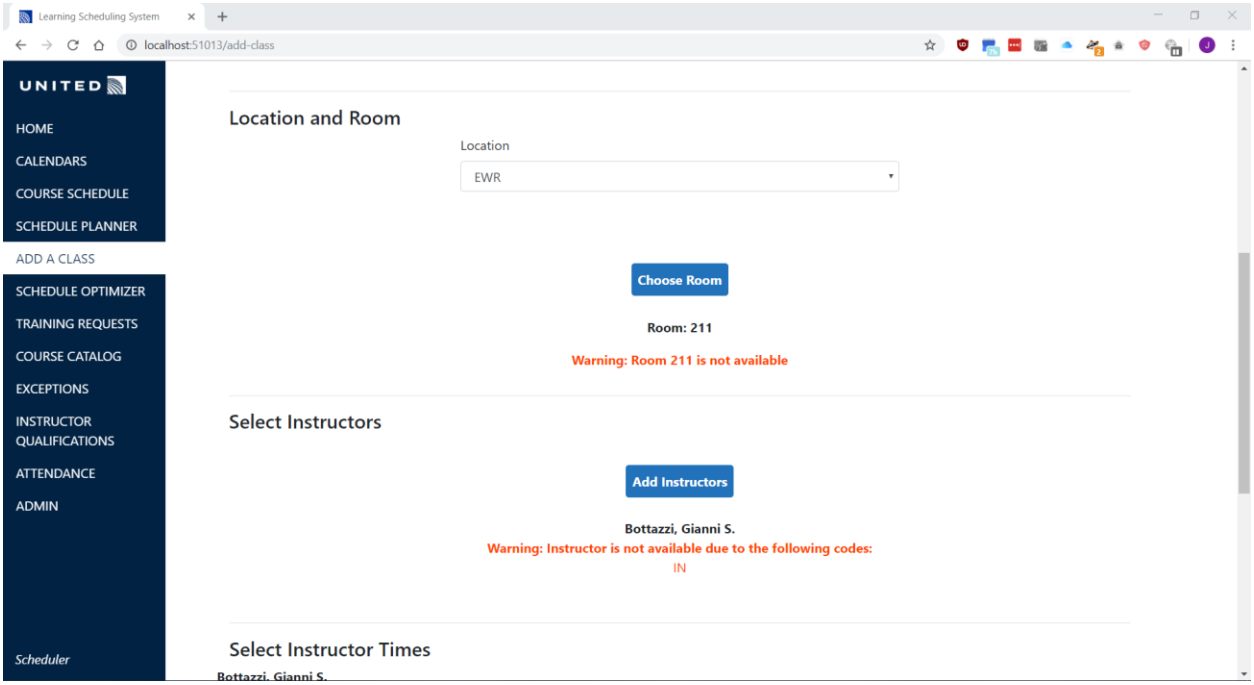

Figure 5: Resource Conflict Alerts in the Add a Class Modal

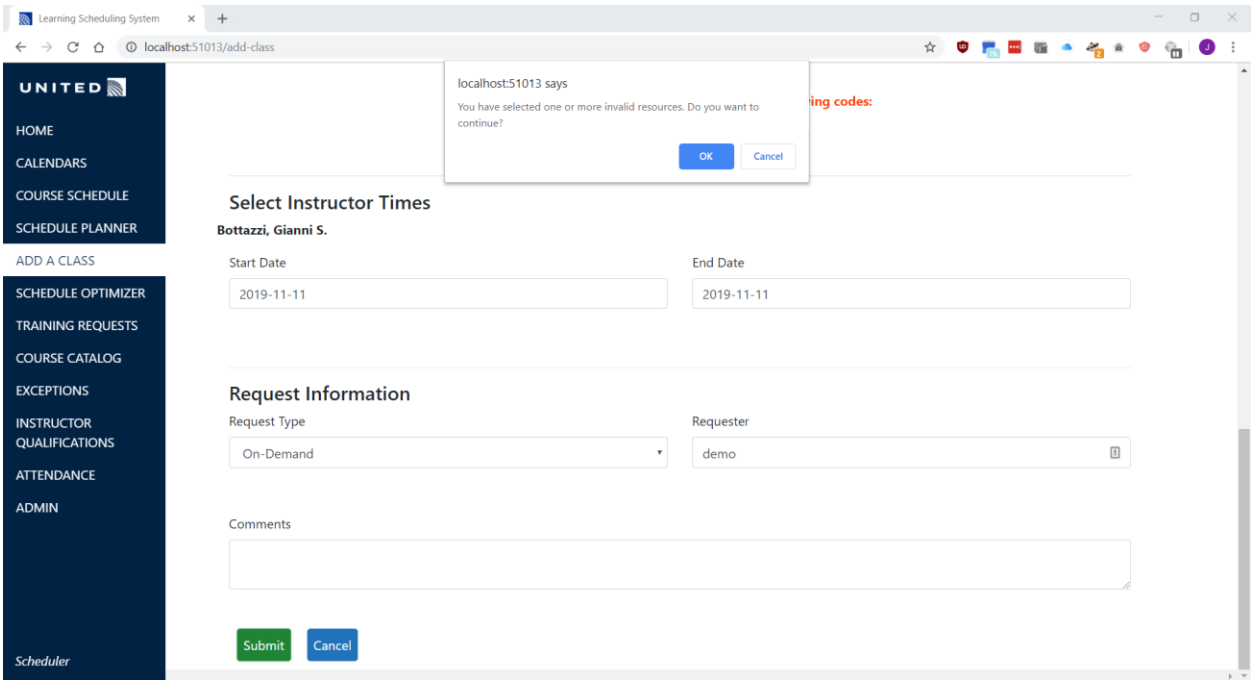

Figure 6: Confirmation Modal When Submitting a Class with Conflicts

Another design feature is the inclusion of more feedback and validation prompts when a user is scheduling a class. From Figure 5, the user will see that if there is a potential conflict when scheduling, an alert will appear informing the user of this conflict and whether they approve of

the conflict. In addition, from Figure 6, when selecting either a course or instructor, the rows will be colored to make it clearer if the instructor or room has a conflict.

Mobile View

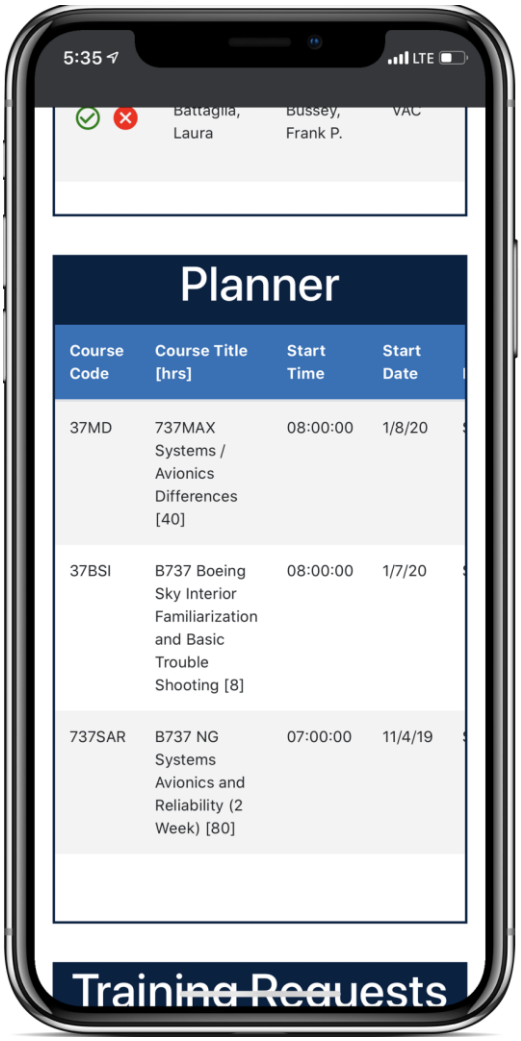

Figure 7: View of home page

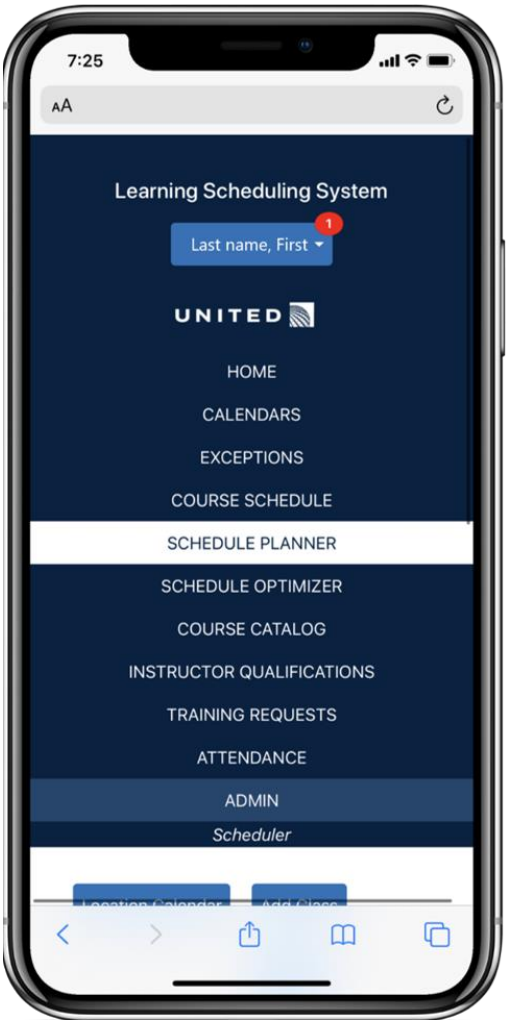

Figure 8: View of menu

Figure 7 shows the home screen the user will see after they login to the application. They will be able to see what classes are scheduled today, exception requests, the classes that are planned, and the training requests as well as details about them. Although this shows less than the desktop version, it will be cleaner. The other calendars will still be available through the menu.

Figure 8 shows a view of the menu from a mobile browser. By using specialized media queries, the user will be able to access the website from a variety of devices and have access to all the same functionality of the main desktop website.

Access levels

	Guest (default)	Coordinator	Instructor	Supervisor	Manager/ Scheduler	Application Admin
View course catalog, scheduled classes, locations	✓	✓	✓	✓	✓	✓
Request courses		✓	✓	✓	✓	✓
Enroll students up to set number of days before class start date		✓	○ (for their class)	✓	✓	✓
View instructor scheduling assignments			✓	✓	✓	✓
See rosters of classes			○ (for their class)	✓	✓	✓
Calendar view of their class			✓	✓	✓	✓
See instructor info			○ (work activities)	○ (default to their instructors)	○ (default to all instructors)	✓
Ability to schedule classes, edit unpublished schedule, run optimizer					✓	✓
View only unpublished plan				✓	✓	✓
Lock/unlock classes					✓	✓
Manage class rosters at any time					✓	✓
Manage all other user roles					○ (except admin)	✓
View/edit application settings						✓

Table 1: Summary of User Access Levels

✓ = access granted ○ = access granted with condition

On the previous page, the table shows the access levels for all the different users in the application. There are several changes of note from the previous application including: managers/schedulers being able to manage roles for guests, coordinators, instructors, and supervisors; coordinators being able to request a course, view the schedules and have the ability to enroll students in courses; and a new admin role that can view and edit the application settings as well as manage all the user roles.

Technical Specifications

System Architecture

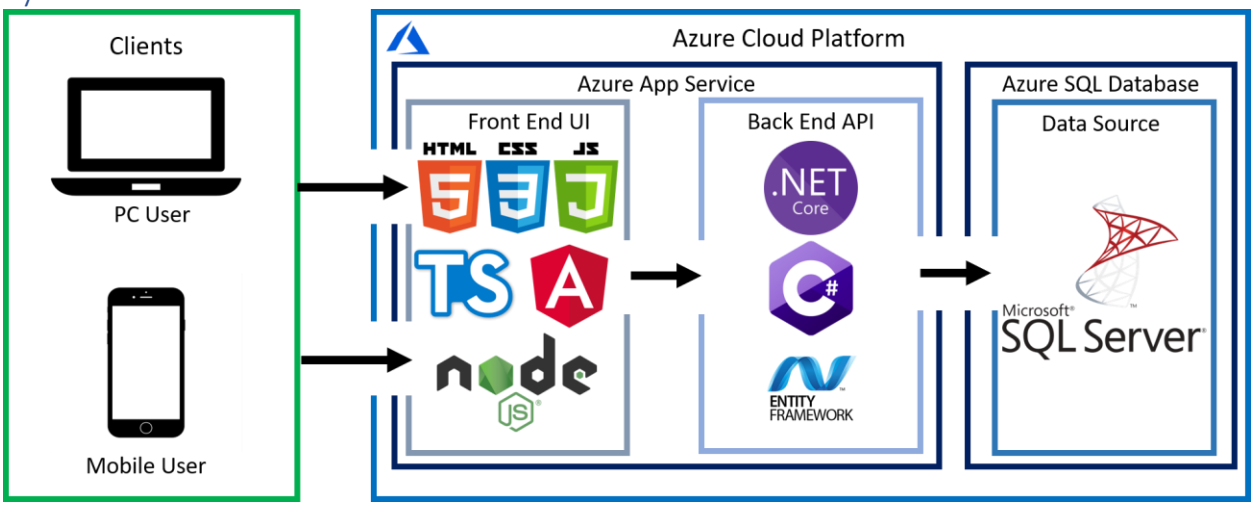

Figure 9: System Architecture Diagram

An end user can access the application from a browser on a desktop, tablet, or smartphone. If accessing from a smartphone or tablet, the front-end Angular components will utilize CSS media queries to provide a more optimized user interface for a smaller screen. Before accessing the website, the user will be required to log in using a form-based system. If Oracle Access Manager (OAM) authentication is turned on, the website will integrate with OAM to identify users. From there, the web application is hosted as an App Service on Azure. Being a single page application, alongside the UI built with Angular, the back-end controllers and APIs are built with ASP.NET Core. To access the data source, .NET Core uses the Entity Framework to create, read, update, and delete data from the database. The data source itself is a Microsoft SQL Server hosted on Azure.

System Components

Cloud Platform

Azure:

Microsoft's cloud computing platform Azure will be used to host and build the web application and data source. Specifically, the main services that will be leveraged with Azure are the Azure SQL Service, App Service, and App Service Plan. Through Azure, these solutions enable the mitigation of having to locally host the Microsoft SQL Server and ease the process of migrating the solution to an on-premise server for United Airlines.

Security:

Eventually when the application is moved into production, it will utilize United Airline's on-premise Oracle Access Manager (OAM) for authentication. As this is an on-premise solution, it

is being researched if it can be integrated with the development environment within Azure. Alternatively, a separate user authentication can be built for the application to work with OAM.

Web Application

Visual Studio 2019:

A powerful IDE from Microsoft that will be used to develop the ASP.NET Core web application with Angular. It also allows for the integration of the open-source NuGet package, Entity Framework, which works as an object-relational mapping framework with the SQL Server.

Navigation:

Navigation within the web applications is accomplished through Angular. Angular features the Angular Router to navigate between web pages and components by interpreting instructions from the URL.

Layout:

The web application will continue to utilize the front-end development framework, Bootstrap 4.2, for quick development of response web pages and all the components.

Form Submissions and Actions:

All data form submissions, actions, and retrievals will be delivered between the front-end and the MS SQL Server through the Entity Framework which enables a comprehensive object-relational mapping.

Mobile Web View:

Separate HTML and CSS files will be created for each of the Angular components. These will be utilized to generated UIs specifically for viewing on a smartphone. By only creating separate view components for the mobile web site, code reuse of the web application is maximized. Therefore, team development can be prioritized on feature implementations.

Schedule Optimization

Algorithm Overview

To achieve the goal of optimizing a list of classes for the schedule plan, a recursive tree will be used. The tree starts by selecting a class from the list of all the classes to be optimized, starting with the longest class. From there, it creates child nodes based off all the possible ways that this class could legally be scheduled. Then, the algorithm will recursively be called from each child node by asking them to grab another class from the list of classes to optimize. If a node is unable to schedule the next class legally, it will discard that class from the list and try the next available one. This process continues until all nodes find themselves with no more classes to consider. Then they return their schedules to their parents, and each parent determines which child's schedule is the best based off the number of classes that were scheduled and the user's selected optimizer parameters.

Optimizer Parameters

Default

By default, the optimizer will run with no additional parameters set. For this setting, the optimizer will return a schedule with the highest number of classes that were successfully scheduled. If the optimizer had to consider between five answers that all had successfully scheduled ten out of ten inputs, it would select one randomly.

Instructor Least Total Travel Distance

For this, the optimizer will be set to minimize the total instructor travel distance by prioritizing selecting a local instructor for a class. As the branches are building out the different possible schedules, they will have a score representing the total distance instructors will have to travel to satisfy the schedule. For example, if one class is being taught in Denver but is using a foreign instructor from New Jersey, then the total instructor travel distance for that schedule will include the distance in miles between the Denver facility and the instructor's home facility in New Jersey. In the end, the optimizer will return the schedule with the most classes scheduled and then the lowest total instructor travel distance. This prioritization will help to ensure that the local instructors are being utilized and that instructors will minimize their time they are required to travel to teach.

Instructor Longest to Teach

This optimizer prioritization will tell the optimizer to select the instructor who hasn't taught the course in the longest time. Every time a course is scheduled, it will accumulate for that schedule the total time between the last time that instructor taught the course and when the class is scheduled to start. In the end, the optimizer will select the schedule with the largest time value. This prioritization will ensure that instructors are being kept up to date on their instructor qualifications and that a single instructor will not teach all instances of a course if others are available.

Instructor Most Specialized

What this optimizer prioritization will do is select the instructor for a course that has the least amount of total instructor qualifications. For example, there are two available and qualified instructors to teach a course named instructor A and instructor B. Instructor A is qualified to teach 20 other courses in addition to this one, while instructor B can only teach this course. Under this prioritization, the optimizer will select instructor B because he is the most specialized instructor regarding this course. Each potential schedule being built will accumulate the total amount of qualifications all the chosen instructors have. In the end, the optimizer will select the schedule that has the least amount of total qualifications of its instructors. This prioritization in effect will promote a larger utilization of instructors as ones who can only teach one or two classes are more likely to be picked.

Speed

For this prioritization, the optimizer will not be necessarily optimizing the inputs to find the best schedule. Instead, it will be prioritizing speed of the results by just finding a possible schedule

as fast as possible from the inputs. Here, the optimizer will go through the list of inputs and assign the first available instructor and room to each class. It will not be building a recursive tree to compare multiple results as it is only creating one result. This will guarantee a good, legal answer but not necessarily the best answer.

Algorithm Bounding

To minimize the computations required by this algorithm, bounding must be used extensively. One negative bounding that will be implemented is that if a node is unable to schedule the next class legally, and a solution has already been found by another end node that is legal and includes all classes, then the node simply terminates and doesn't bother branching anymore. This is because it is impossible for any of that terminated node's children to come up with a solution more optimal than the one already found.

Risk Analysis

Handoff of Final Code Between the Team and the Client

Difficulty: Low
Description: The handoff of code between the MSU team and United was a big issue at the end of last semester. Due to how the database was set up for the MSU team, there were complications with integrating the application with United's servers.
Mitigation: In the hopes of avoiding the same issues as last semester's team, we are using more cloud-based solutions from the start of the project. This should allow for easier transfer of ownership. Additionally, we will also give the client a beta version of the code in the end of November, so that we can change any issues before the final handoff is made.

Time to Run for the Schedule Optimizer

Difficulty: Hard
Description: Creating a recursive tree to represent all possible schedules within a time frame will cost a great amount of time to compute.
Mitigation: Bounding must be used extensively on the recursive tree to minimize the number of branches that are created to stop unnecessary schedules from being considered. Additionally, preparing the list of classes to optimize by sorting them by class length and other metrics can ensure the tree is built in a way that maximizes bounding.

Oracle Access Manager (OAM)

Difficulty: Medium
Description: United Airlines uses OAM for their user identity security. Azure has its active directory that is uses, and our team is unsure if it is possible to integrate OAM with the web application if it's hosted on Azure.
Mitigation: We will rebuild the login page for the web application to utilize Azure Active Directory to model the application closer to how authentication will work within United's systems. Additionally, we will continue to research the possibility of integrating OAM with the web application, which would require the least amount of migration in the end.

Accuracy of Data

Difficulty: Medium
Description: The data we are using is accurate up until June 2019. We have accurate courses and instructors from United, but we are creating our own mock schedule. We are also creating activities for instructors, such as when they are on vacation or teaching, which will be used as schedule optimization criteria and the resources needed for each classroom. There is a possibility of getting a current schedule, but it is uncertain how the data will be uploaded to the website. Additionally, data is missing for classroom resources.
Mitigation: For now, the group will communicate with the client what our mock data looks like, and what changes we make to it. Confirm with them that our mock data is a good representation of what currently exists, so that features can be created as they are intended to be used. Once the team is given current data, we will work on uploading it to the site.

Schedule

Week 1 (8/28 - 9/1)

- Initial meeting with group members
- First call with clients

Week 2 (9/2 - 9/8)

- Review spring 2019 code base
- Research about technologies that will be used

Week 3 (9/9 - 9/15)

- Status report document and presentation
- Start project plan rough draft
- Set up Azure web app and SQL database with existing code base

Week 4 (9/16 - 9/22)

- Finish project plan document and project plan presentation
- Visit client in Chicago

Week 5 (9/23 - 9/29)

- Project Plan Presentation
- Setup custom domain and research OAM integration with Azure
- Add mobile friendly components for home, calendar, course schedule, course catalog, course request, and activity pages

Week 6 (9/30 - 10/6)

- Setup web page, controller, and base classes for the optimization feature
- Add additional calendar views to calendar page and extend functionality
- Finish adding mobile and tablet responsiveness
- Add mobile friendly views components for attendance, instructor qualifications, planner, optimizer, and admin pages

Week 7 (10/7 - 10/13)

- Work on Alpha Presentation
- Create ability for optimizer to check possible schedules for classroom assignment
- Design result screen for optimizer
- Add ability to move and edit classes from calendars

Week 8 (10/14 - 10/20)

- Alpha Presentation
- Expand legality checking for optimizer to include instructor
- Work on conflict notifications for adding or editing a class to a schedule planner

Week 9 (10/21 - 10/27)

- Create separate login that uses OAM
- Add legality check for release rate in the optimizer algorithm

Week 10 (10/28 - 11/3)

- Implement optimizer prioritization for instructors who haven’t taught the longest and instructors who are most specialized for each class
- Continue working on conflict notifications

Week 11 (11/4 - 11/10)

- Meetings with client in East Lansing
- Add automatic log out after period of inactivity and other security features
- Add ability to move optimizer results to the schedule planner
- Increase speed of optimizer
- Start creating the recursive structure for optimizer

Week 12 (11/11 - 11/17)

- Work on Beta Presentation
- Finalize functional feature implementations, including conflict notifications
- Finish recursive optimizer algorithm

Week 13 (11/18 - 11/24)

- Beta presentation
- Project video filming
- Work on testing, debugging, and polishing of web app in desktop and mobile views

Week 14 (11/25 - 12/1)

- Finish project video
- Work on testing, debugging, and polishing of web app in desktop and mobile views

Week 15 (12/2 - 12/8)

- Submit project video
- Submit All Deliverables
- Design Day