Final Project Design- Team 6

Abby Davidow, Grant Schnettgoecke, Archana Ramakrishnan, Adam Soelter, Alfonso Martello

Project Name: What's Due When (WDW)

Project Synopsis:

What's Due When web application lets students see upcoming schoolwork for all their classes in one place and allows professors to add/edit their class schedules.

Project Description:

College students are constantly barraged by emails, announcements, and handouts from a wide variety of different sources. Keeping track of this deluge can be overwhelming, especially for freshmen just entering the college world. This information overload can add unnecessary stress to students' lives and managing it can take substantial time and resources. What's Due When (WDW) aims to solve this problem by providing a central application that faculty and students can use to control the flow of information. Faculty can use WDW to add events (such as notes, due dates, or comments) directly to their students' calendars on either a weekly, monthly, or semester basis. Students can use WDW to edit or add events to their own calendar with the extra benefit of having all events relevant to them gathered in one convenient place. The result will be an online platform (specifically, a website designed using React.js) where students and faculty can have schedules delivered in a manageable and convenient way.

Project Milestones:

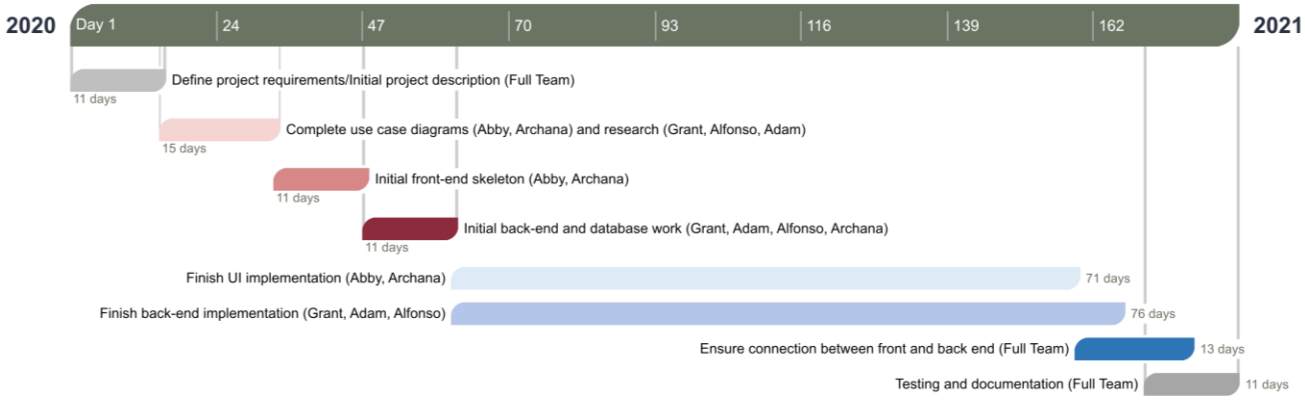
Fall Semester

- 10/5/20 - Define project requirements/Initial project description
- 10/23/20 - Complete use case diagrams and research
- 11/20/20 - Initial front-end skeleton

Spring Semester

- 2/12/21 - Initial back end and database work/Final project report
- 2/26/21 - Finish UI implementation
- 3/5/21 - Finish back-end implementation
- 3/16/21 - Ensure connection between front and back end
- 3/23/21 - Testing and documentation

Team 6 Gantt Chart



Project Budget:

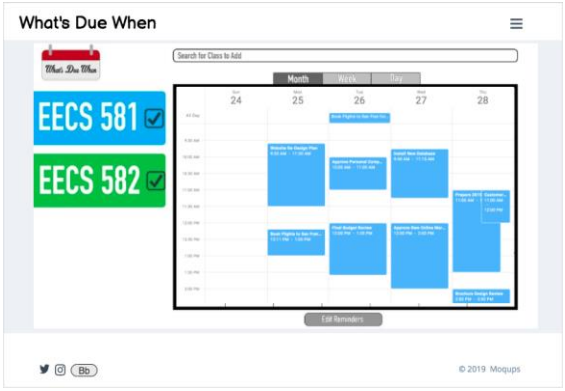
The possible costs incurred will be for the domain name, web hosting service, and the database. A one-year registration from *Namecheap for Education* costs $0.00 for whatsduewhen.me. We will host our MongoDB database using MongoDB Atlas, which provides 5GB of storage for free (we do not anticipate using more than 5GB at this time). We plan to host our website using AWS, which offers both a free-tier and 12-months of free services. Thus, we don't anticipate any hosting costs. Special free training in React JS and databases was used during October/early November to start implementing the project. In short, we have no costs.

Preliminary Project Design:

How the Software Works

Home Page

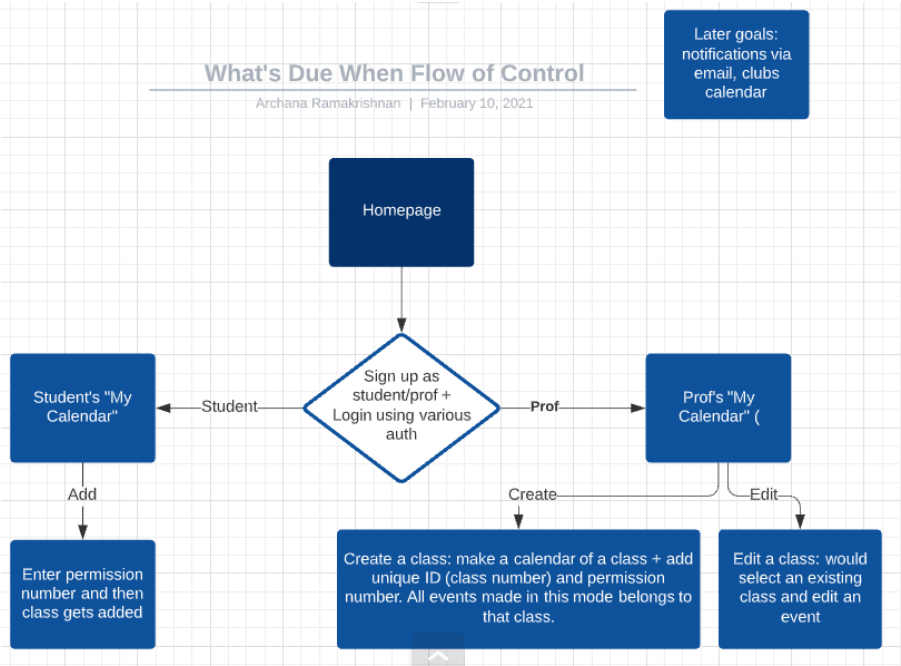
Main Page



Our website will have five main UI views for users. The first view for any user will be the login page. Faculty and student users will have separate homepages, however, the view for both

users will display a calendar containing events for all their classes as shown in the mock designs above. On this page, they will be able to adjust their view and choose other editing options. The difference between faculty and student views is the additional option to create a new class and change existing classes. Clicking on these options will bring up a form view where all required input will be entered in order to create a new course. When creating a new course, the professor will essentially be creating a calendar designated for the class. To edit, they will make changes to an existing class calendar. To create an event, users will double click the calendar to bring up a built-in event form. The control flow chart below further describes how the website will be structured.

Control Flow

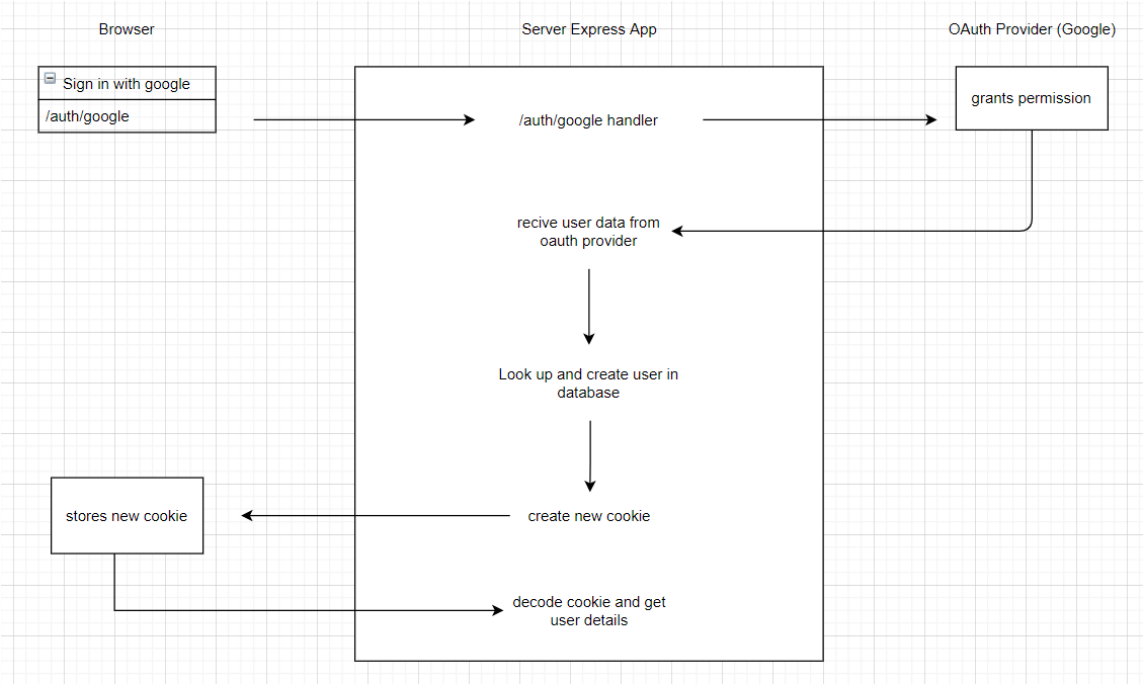


For the front end of our website, we will be using React JS library to build our UI. We decided on React due to its fast-learning curve and re-usability in terms of components. A couple of our team members have some experience with React and found it to be straightforward to learn and it has helpful developer tools. The reusability will be key for our design since the student and faculty views will require many similar components.

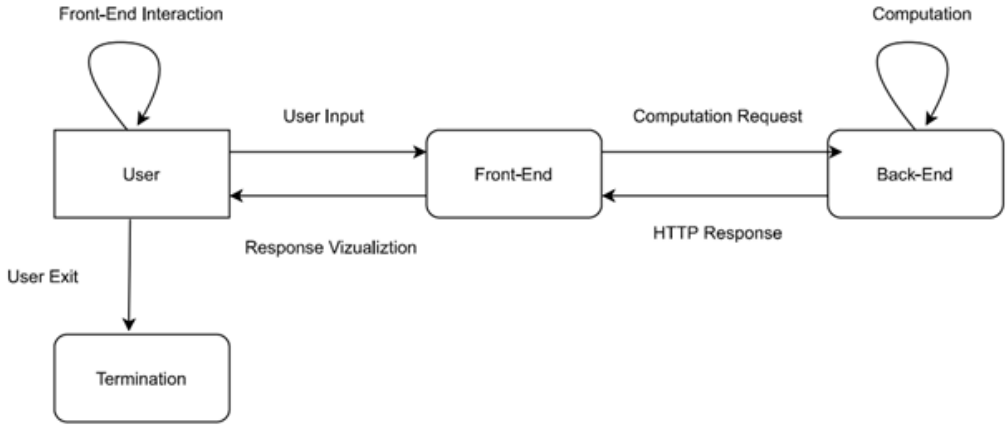
Since a major aspect of our UI consists of a calendar, the React Scheduler library provides us with many helpful built-in functionalities. Their calendar provides multiple views (month, week, and day), integrated navigation between dates, and a default form for adding events.

The diagram below shows what is happening behind the scenes when authenticating a user with a social login. The user will first be directed to a custom endpoint (/auth/google). Then express will take over and handle what happens when the user is at that route. They will be redirected to a consent page provided by whatever social they choose to login with. On the consent page, the user can view what kind of information our application is requesting. Once they grant us permission, they will then be redirected again to another custom route. At this point we now have the users profile information. Next, we need to verify if this user has logged in with us before. We'll do a lookup in our database and check if the user exists. If there is no record of that user, we will create a new one. Otherwise, they have visited us before, and we

can move on to the next step. Next, our app will create a new cookie and send it to the browser. Whenever the user makes any request while logged in, express can decode the cookie and know who made the request.



Another key component of our, and any, web project is the backend. The backend is the portion of the code that runs on a server, performs computations, and responds to user input remotely. The user input is generated by the user by interacting with our ReactJS front-end before being transmitted through an HTTP protocol to our back end. After the back end performs the necessary computations requested by the user, it generates a response which it transmits back through an HTTP protocol. This process continues until the user stops interacting with the application. This process is summed up in the figure shown below.



This server client architecture allows us to move computations away from possibly resource limited user devices such as cell phones and tablets and towards the more performant machines that run the server. Additionally, it allows us to share common data in one process (the back end) rather than duplicating the same data across multiple processes/devices on the

front end. For example, many students are enrolled in the same class. So, it would be wasteful to store information about that class on every user device. Instead, we can store one copy of that information on the server and let every user access that information over the internet when that information is needed by the user.

For the back end of our project, we intend to utilize NodeJS. NodeJS is a cross platform backend for web applications that allows the user to run JavaScript code outside of the browser. One of the main benefits we receive from using NodeJS, in conjunction with our ReactJS frontend, is that we can write the front end and back ends in the same language. Additionally, NodeJS is designed to be more performant than traditional back-end models because it makes use of a single threaded event loop rather than a traditional request/response model. This allows it to avoid the context switching overhead that is common in other back-end API's.

For our database, we will use MongoDB. MongoDB stores its data inside of JSON files so changing the structure of the data is simple and quick, which would allow us to easily change how our project functions in the future. Our data will be split into three sections: student information, professor information, and class information. The student information will contain identifying details, the course ids of the courses they are enrolled in, and details about custom events they have added to their own calendar. The professor information will contain identifying details, the course ids of the courses they are teaching, and the details about the custom event they have added to their own calendar. Course information will contain details about assignments and class meetings (and a unique course id for identification purposes). Additionally, courses will contain a randomly generated permission number that is created when a professor makes a course. Students can add a course to their list of enrolled courses only if they have the permission number for the course from their professor.

Design Constraints

Technical Constraints

Programming language: We are using the MERN stack. It stands for MongoDB, Express, ReactJS and NodeJS. Express is a framework that works with Node.js's web server to organize the functionality of the application. It simplifies routing, renders dynamic HTTP objects, and adds utilities to the HTTP objects. ReactJS is a library to build interfaces using reusable components. NodeJS is a backend environment that facilitates returning content from the server to the client.

Upon looking at the differences between a traditional RDBMS and MongoDB, which uses noSQL, we made some observations. NoSQL databases are built to have flexible schemas and specific data models. MongoDB will be fast and highly scalable as they use a document model containing key-value pairs rather than relational tables.

Use of specific libraries**:** Node Package Manager (NPM) has a react scheduler package that we could implement for the user interface. This package provides a calendar view to pick days, months or years. It is fast, lightweight and easy to style. NPM also provides a react-google-calendar-api to manage a google calendar and node-outlook to integrate the Outlook calendar. We may opt into using one of these if we are able to introduce some level of customization to fit our use case. It might help to stay consistent if we use google/outlook authentication.

Operating system or platforms supported: We aim to build the web application to be accessible through a desktop or mobile browser. The proper rendering of a lot of the front-end components in React will determine the platforms supported. React Scheduler supports the latest versions of all major browsers: Google Chrome, Mozilla Firefox, Safari, Opera, and Microsoft Edge.

Business Constraints

Schedule: The project will be ready for deployment by the end of April 2021. The timeline leading up to that goal involves planning, communicating with the stakeholders, iterative prototyping, implementing the application, and getting user feedback. We picked the project idea in late September 2020. In early October, we outlined the requirements by meeting with the professor who introduced this idea. Following that, we researched the technologies we would use and planned the design through use case, database schema, and wireframe diagrams. Over the winter, we began our initial implementation with production picking up in earnest after the spring semester began. Following the completion of a minimal viable product, we will iterate on that product after some feedback to integrate the student and professor use case by March. March and April will be our last sprint of enhancement and bug fixes.

Team composition and make-up: We have a team with diverse skill sets and interests. Grant and Alfonso will work as an intermediary between the front end and back-end sides of the application. Adam will focus on perfecting the back-end functionality, while Alfonoso will support the database management and configurations. Abby and Archana will primarily work to design and enhance the user interface and experience on the front end. These roles will complement each other and shift as our priorities shift through the project timeline.

Ethical Issues:

A major ethical issue that we will have to account for as we implement our project is ensuring students' rights as laid out in the Family Educational Rights and Privacy Act (FERPA) are protected. Students have the right to choose what is disclosed about their education records. This includes grades, the classes they take, or any other identifiable information from their education record. If we are going to store information about the classes a student is taking in our database, we need to take appropriate measures to protect their information from possible hacks into our database. One way we could increase the security of our database is by double hashing the information we store, so if there is a breach, the data that would be vulnerable is protected by a second hash. We are additionally planning on relying on the built-in security features provided by MongoDB Atlas (the service we are using to host our database).

Another minor ethical issue that we need to abide by is ACM Code of Ethics and Professional Conduct rule 1.3 "Be honest and trustworthy." The concern for this rule is that our project is being made to make time management easier for students. However, in achieving this goal we also need to convince faculty to use our product. Thus, we have the two-pronged job of designing our software with students and teachers in mind. We will need to be upfront about the purpose behind our project and its capabilities throughout the development process. We are considering several ideas to make our software more faculty friendly, but we should not advertise these ideas as features until we know they will be implemented.

Intellectual Property Issues:

An intellectual property issue that could arise is if we plan to integrate our project with KU. They would be responsible for storing information regarding students' class schedules. Because of this, KU might not allow us the right to some of the code. There are a couple of routes that we could go. We could decide that we want to have the rights to everything that we create and have it as a stand-alone application. KU would then decide if they wanted to buy our software or not. We would then want to make sure we copyright the code that contains the core functionality of our application. Another route we could take is to work with KU. We would sign a contract stating what our property is and what property they will have. The code that will be directly related to helping KU professors and students will be theirs. We will still have rights to the code that is responsible for the main functionality of the web app. Lastly, we may decide that we want to make our project open-source and let anyone use our application. This would allow anyone to use our code and modify it however they like. Since anyone will be able to use our application, we would not have to worry about any intellectual property issues.

Change Log:

- We have decided not to integrate with blackboard because KU is moving away from the platform soon and doing so would add complexity without adding much benefit.
- We decided to use MongoDB instead of MySQL since we have chosen to implement our website with the MERN stack.