Andrew Santa
Ashton Carruthers
Ken Mills
Nathan Brannan
Zifeng Yuan

# Design Document

**Programming Languages:**

- HTML5/CSS
- JS
- Node.js
- SQL

**Tools:**

- VSCode
- Sublime Text
- Xampp
- MySQL

**Classes and Data Types:**

- Admin inherits from User:
    - No unique fields
    - createSchedule(Date start, Date end) : Schedule
        - Calls the Schedule constructor that calls the Shift constructor repeatedly
    - updateAvailability(Employee) : Pair(Time, Time)
    - overwriteSchedule(Schedule sched) : Schedule
        - Calls Shift.setStaffing(Employee[])
- Schedule:
    - Content : Shift[]
    - getSchedule() : Shift[]
    - Schedule(Date start, Date end) : Schedule
        - constructor
        - Calls Shift constructor
- Shift:
    - date : Date
    - times : Pair<Time start, Time end>
    - staffingRequirements : Pair<string, int>
        - pair of position, count
    - timeOffRequests : Employee[]
    - Staffing : Employee[]
    - clockTimes : Tuple<Employee, Time, Time>[]
    - Shift(Date, Pair<string, int>) : Shift
        - constructor

        - getStaffing() : Employee[]
        - setStaffing(Employee[])
- User:
    - username : string
    - password : hashed string
    - thisEmployee : Employee[]
    - myPermissions : Permissions
    - getPermissions()
    - setPermissions(bool[])
- Permissions:
    - roleName : string
    - permissions : bool[]
- Employee:
    - firstName : string
    - lastName : string
    - employeeID : int
    - position : string
    - availability : Pair<Time, Time>[]
    - preference:  Pair<Time, Time>[]
    - onShift : bool
    - setAvailability(Pair<Time, Time>[])
    - getAvailability() : Pair<Time, Time>[]
    - setPreferences(Pair<Time, Time>[])
    - getPreferences(Pair<Time, Time>[])
    - getTimeOffRequests(Employee) : Date[]
    - setTimeoffRequest(Date)
    - getSchedule(Date) : Schedule
    - getOnShift() : bool
    - setOnShift(bool)
    - getPosition() : string
    - setPosition(string)
    - worksOnDay(Date) : Shift[]

**Define File Structure:**

- EmployeeSchedulerApp
    - App
        - Data
            - Db.js
        - Index.js
    - Lib
        - Assets
            - Css
            - Js
        - Util

- Public
    - Css
    - Img
    - Js
- Src
- Test
- tools
    - node_modules
- package.json

**Define how the program will interact with other systems:**

The program will start as a web-based application. This means the code will need to include functionality for many different browsers. The entire storage system will be database oriented. Login information, schedules, employee data, etc. will be stored in databases. MySQL is the foundation of our relational database system, but when scaling, a cloud-based database will be implemented through something like AWS. This will allow people to retain all their schedule information in the cloud and not have to store it locally.

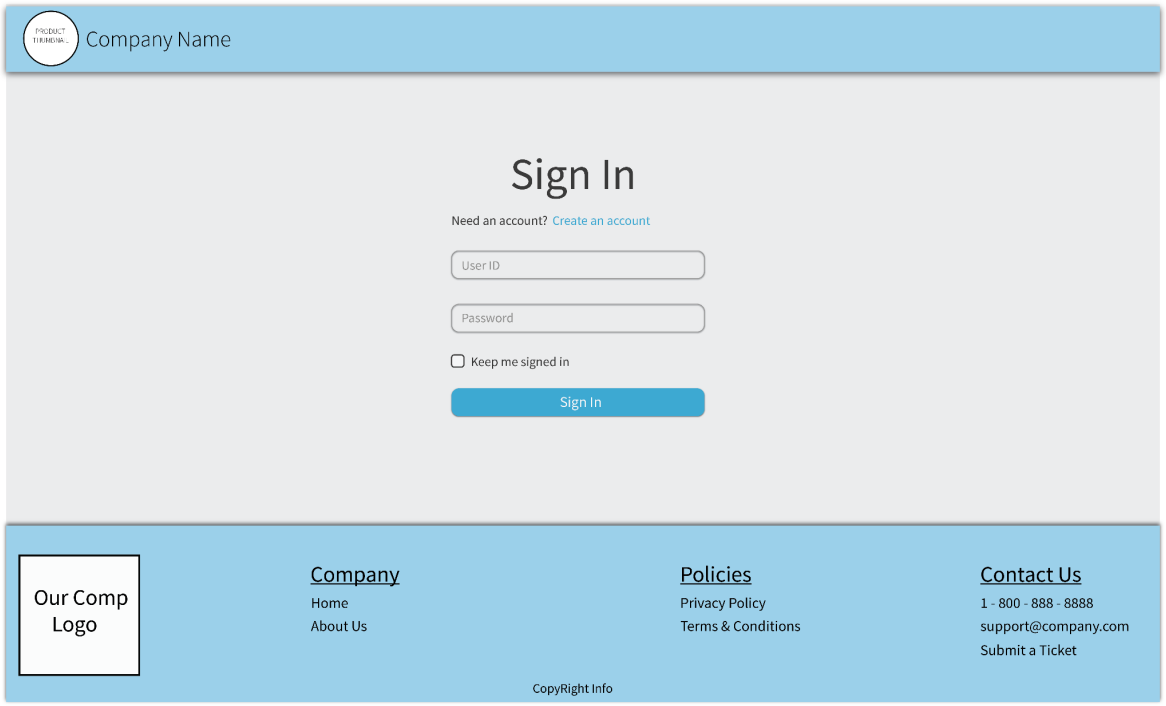
When using the program, a login page will require creation of an account or input of login credentials. These would be done through simple text boxes and drop down menus in which the information will be saved securely. The other important aspect is the schedule input. A variety of menus will be used. The input will generally be drop down menus, but sliders for schedule setting will also be included. Once selections are made, selecting “save” will save the information to a database. Lastly, when admins are setting up the initial job titles, a text box will allow input. If saved, the job title can be used in conjunction with an employee to set their job title via a drop down box.

**Define how a program will be deployed and operated:**

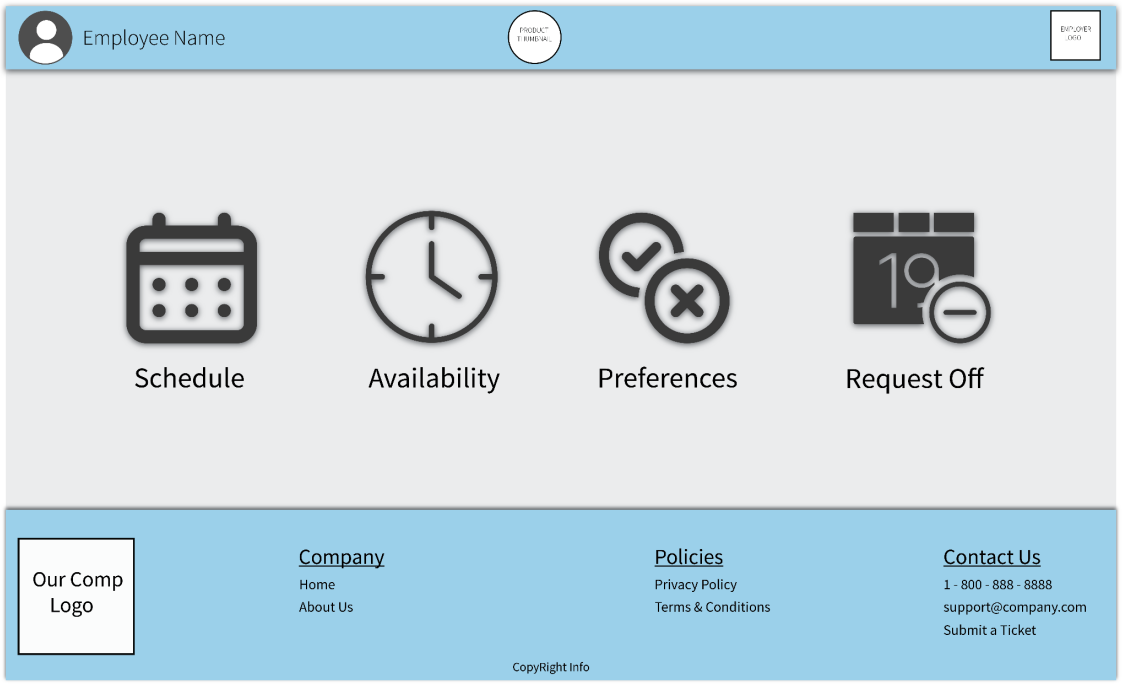
During deployment, we will roll out a couple different versions of the software. We will find some companies that would be available to “beta” test our software. We will continually roll out new software with more features as we go. This will allow us to continue on fixing bugs and gather feedback for further improvements. The first iteration will be a basic scheduler, including other features like preferences, which will come at a later date. Once we finish developing a fully functional product, with everything working on a suitable production level, it will be deployed in “full release”. This will be our official launch in which companies can officially purchase our software. Initially, companies will have to have a local database to handle schedule storage, but as we continue to iterate on the product, cloud-based storage will be more viable. Coinciding with this change, if we find the product to be scalable, the product will be transitioned to a subscription base.

**UI Design (Key Components):**

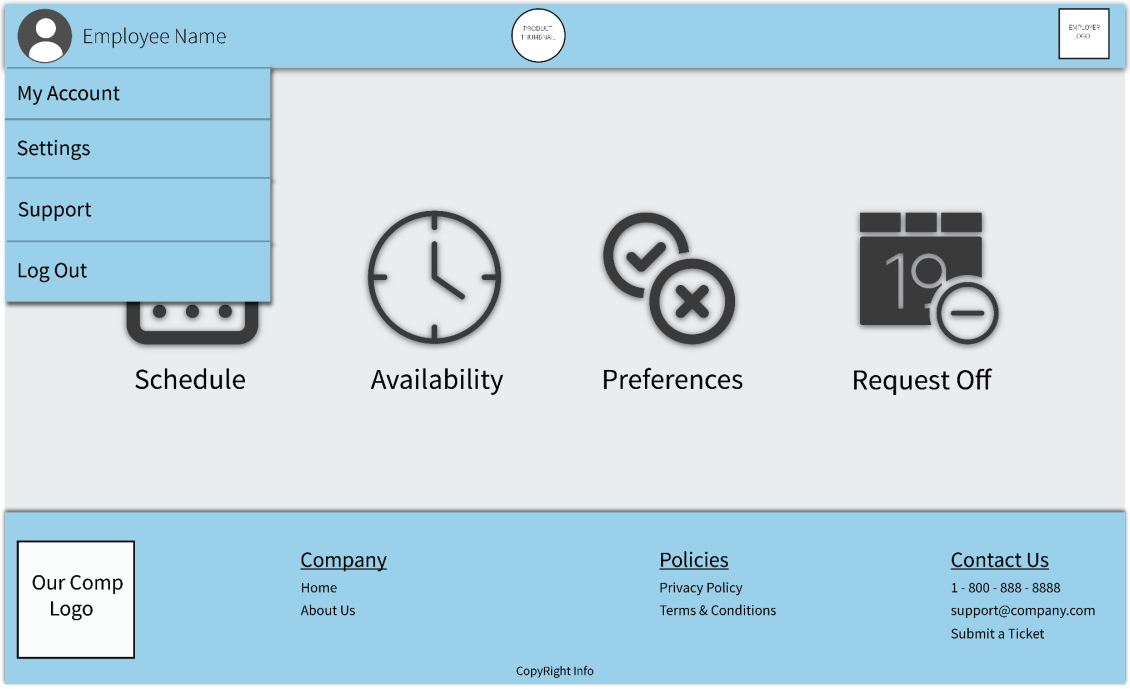
1. Login Page




2. Main Menu

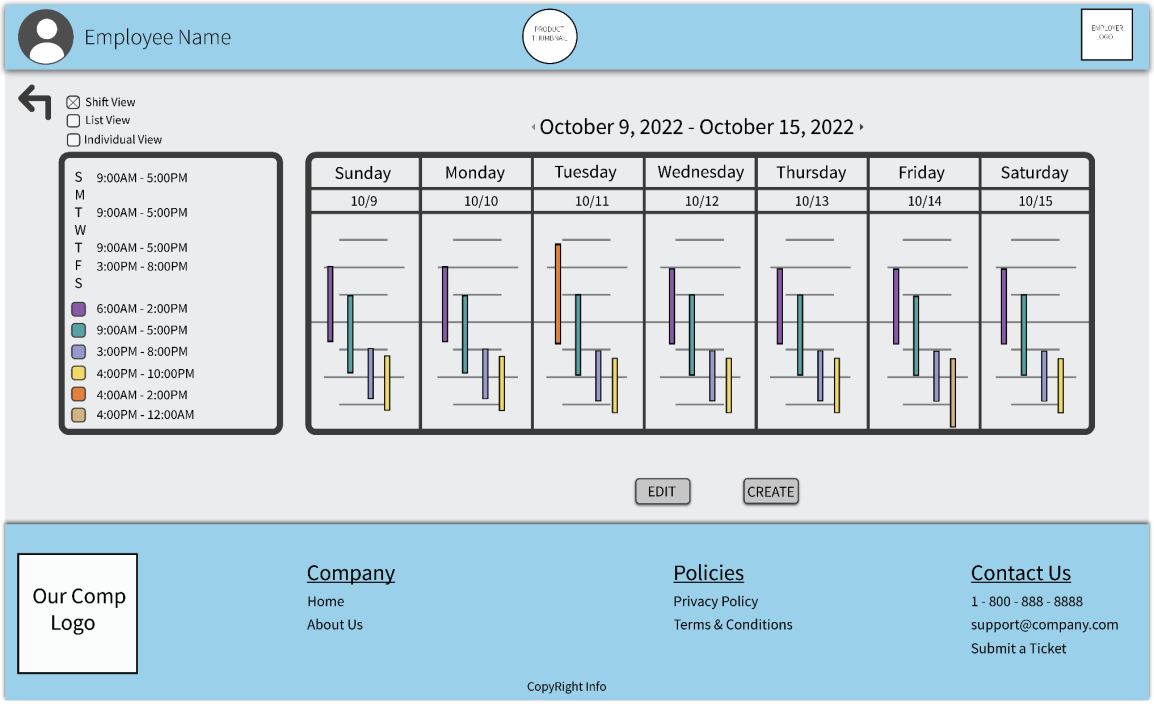



Landing Page with links to all other necessary pages

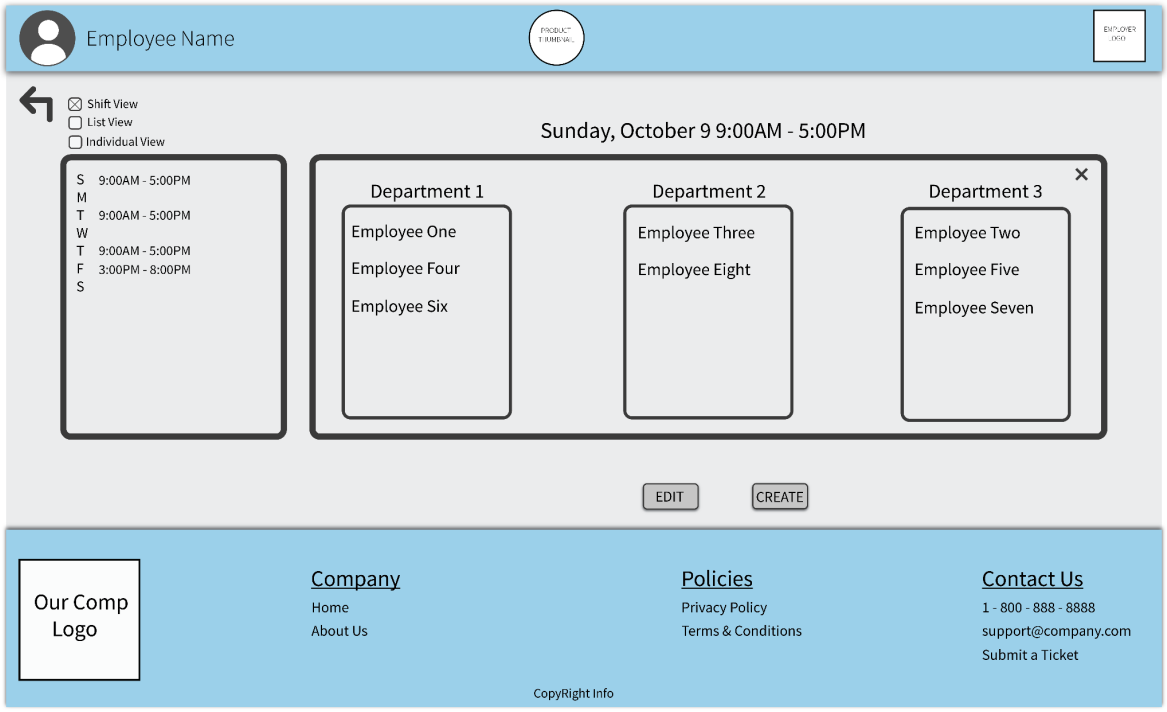



Example of dropdown menu that spawns when user clicks their name.

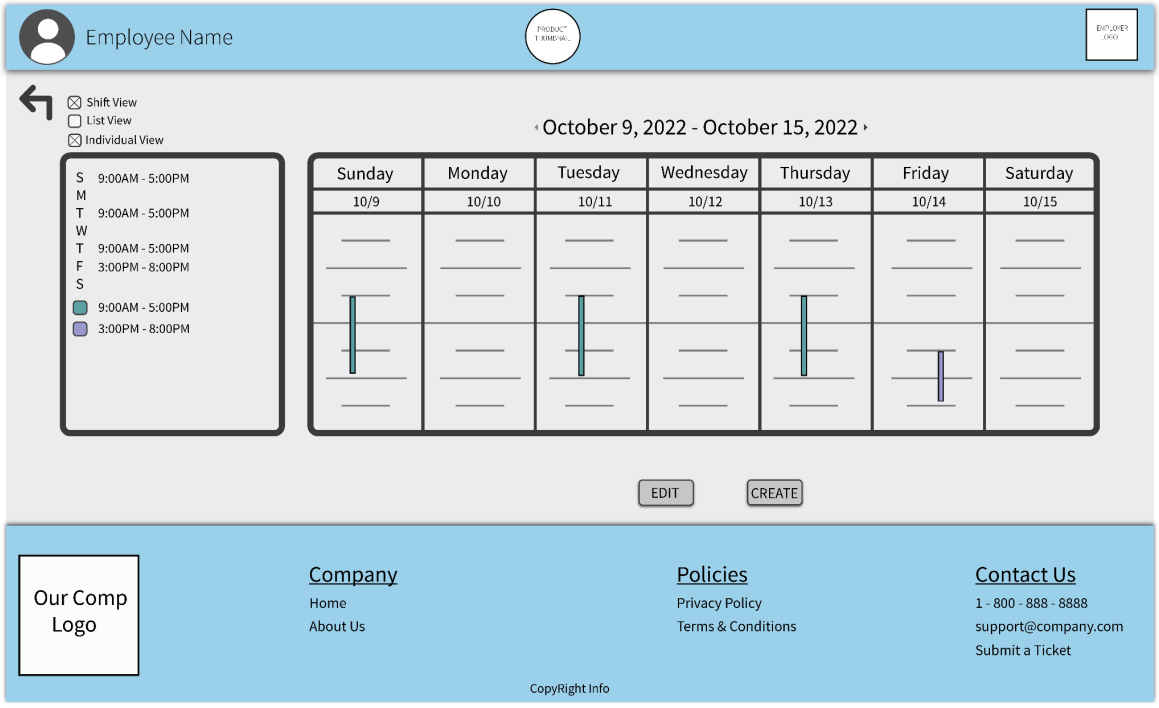
3. Schedule



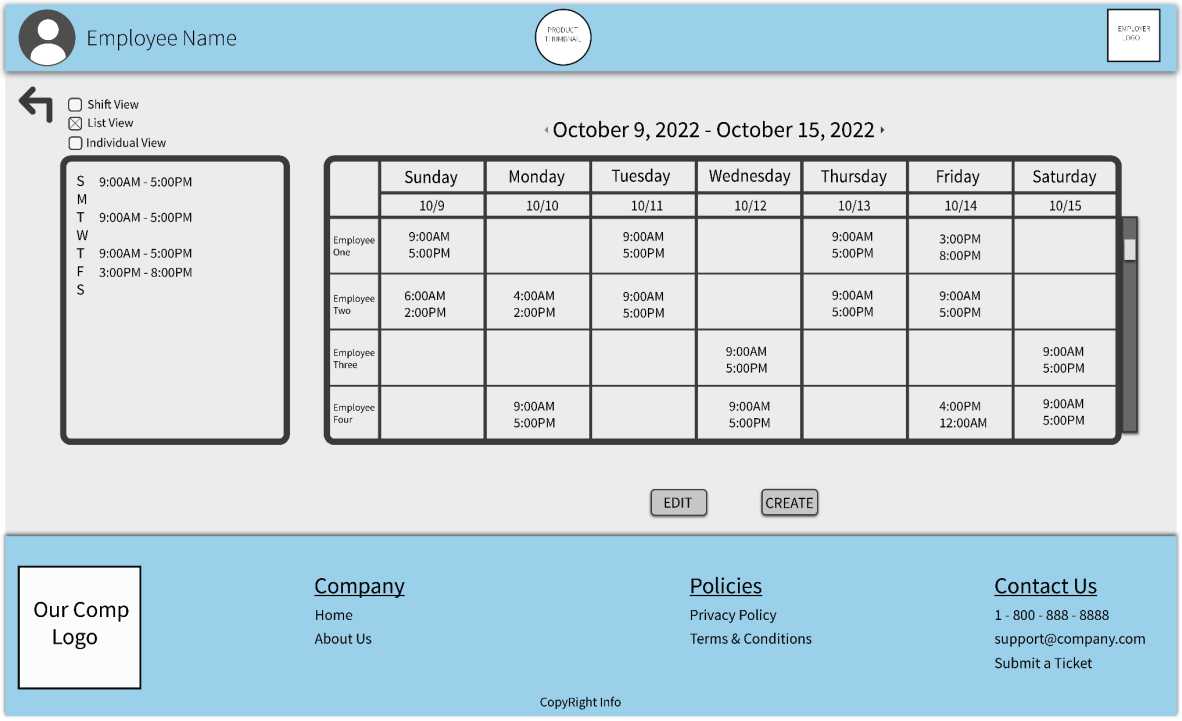
Default view of schedule. Shifts are listed per day in order of start time, a legend on the left includes actual times for shifts, as well as always displaying viewer's schedule. Edit and Create buttons only viewable to users with permission.



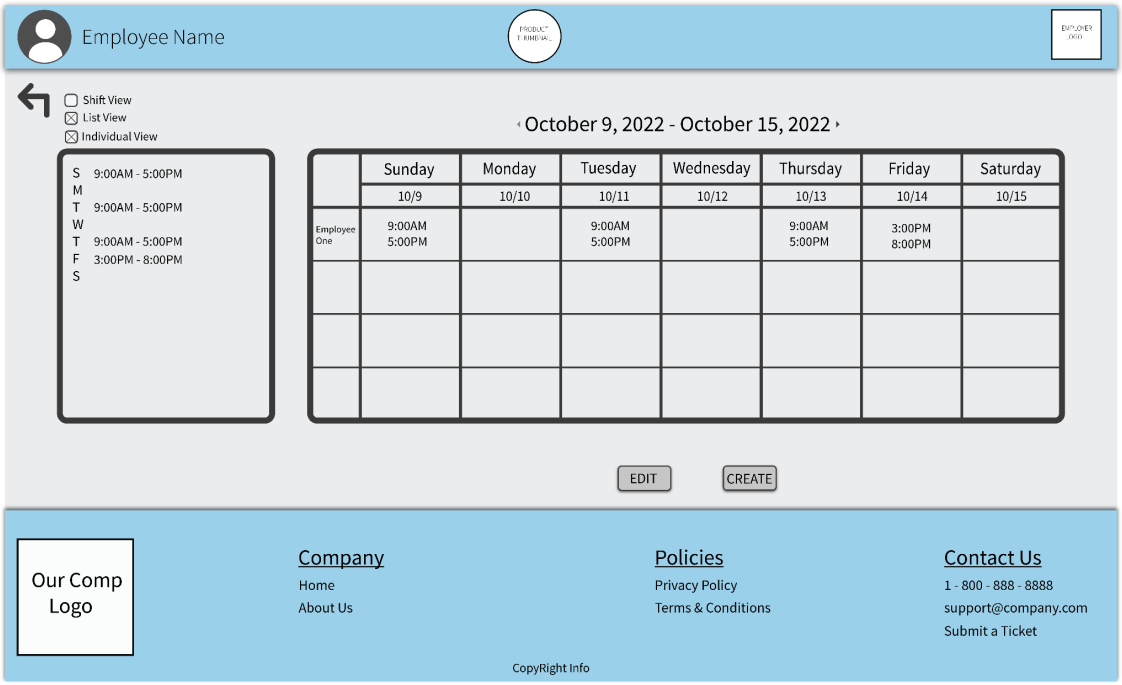

Upon clicking any shift on the schedule while in Shift View, the schedule graphic is replaced with details about the specific shift.



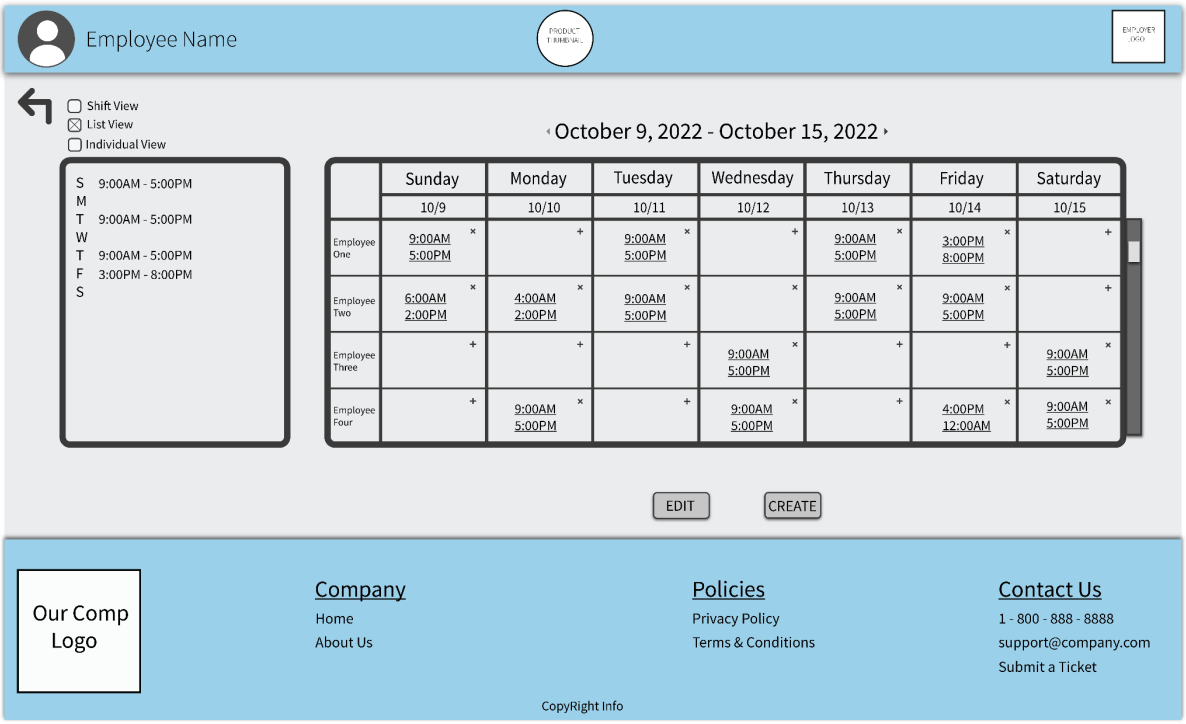

When Individual View is selected, only shifts that viewer is on displays




Schedule displays as scrollable list of employees and their shifts when List View is selected. This is implemented to increase accessibility.

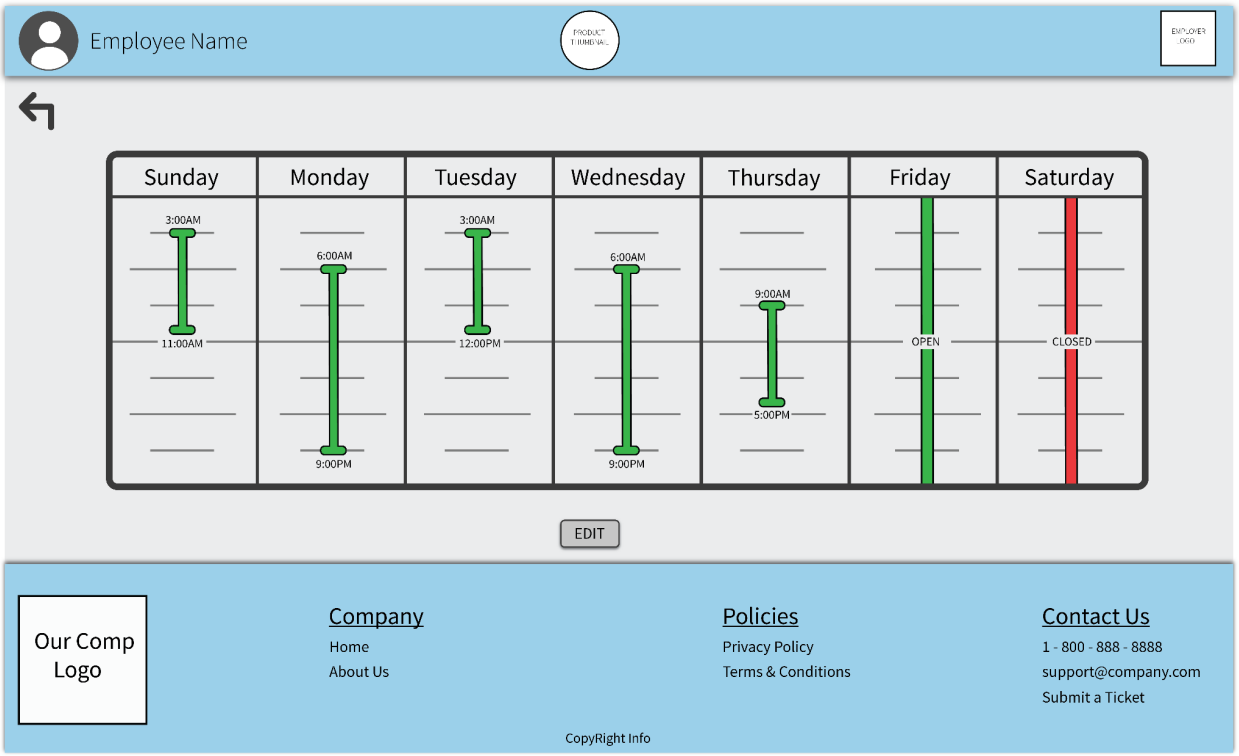


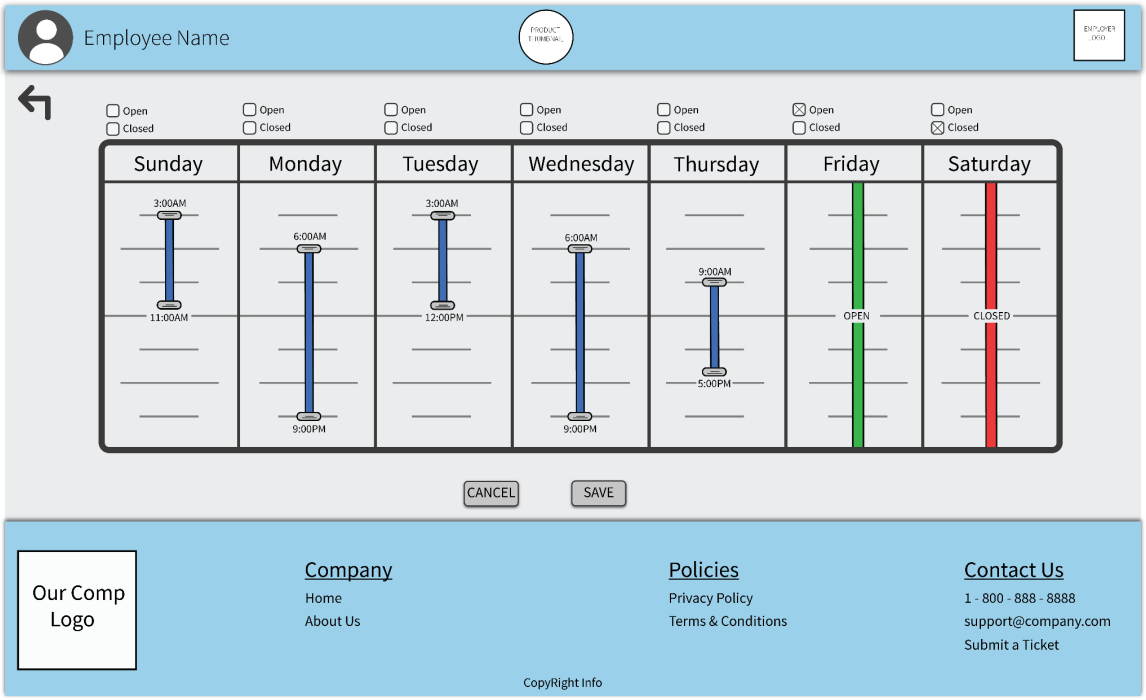

Individual View for List View




Creating or Editing schedule leads to same window (this window). Scheduler restricted to List View for editing to assure simplicity. Scheduler can add/remove shifts entirely, or click on a start/end time to enter a new value.

4. Availability



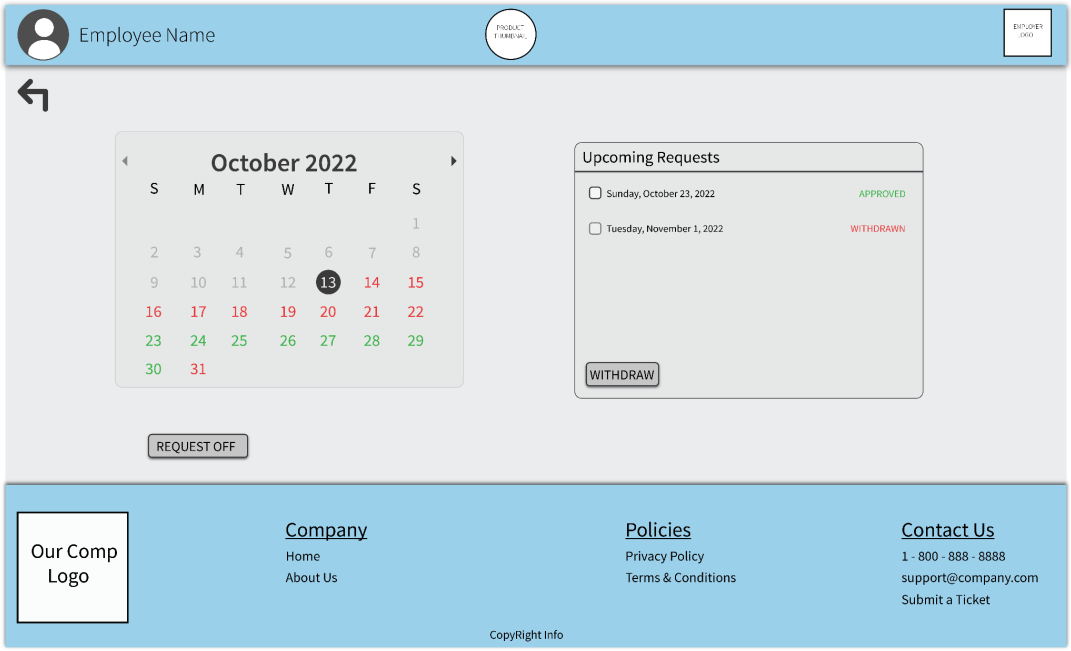



Upon clicking edit, sliders appear, allowing user to input new availability range. Checkboxes provided to simplify open/closed availability. Sliders move in 1 hour increments.

5. Preferences

   Preferences will be a variation of the Schedule Shift View. An edit button will be present to update preferences. Upon clicking a shift, the black outline turns either green or red, signifying preferred/non-preferred (black is neutral). If color block is pressed in Legend, all shifts of that time outline green/red (as well as the block). Selection is cyclical (click through until right color) and both a save and cancel button are present. When not editing, shifts are displayed with color outline of matching preference. Did not make a mockup as most changes from Schedule Shift View to this are animations and can't be properly conveyed.

6. Request Off




Days on calendar that are green are available to request off, while red is unavailable. Buttons added to submit request, as well as withdraw approve request. No need to incorporate denial of request as they are first come, first serve. Requests not allowed when schedule for week is already posted, regardless of allotted requests left.

**Explanation of Minimal Prototype:**

A minimum viable product would include user accounts, employee availability and preferences, schedule generation, and employee position differentiation. Users can sign in, define their availability and shift preferences, and view their schedule. In addition, Users with elevated permissions can generate a schedule and define role/count parameters for each shift.