7Factor Staffing Tool

A Major Qualifying Project submitted to the faculty of Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science.

In cooperation with 7Factor Software, LLC

Submitted 3/1/2024

By:

Sean McNamara

Nicholas Borrello

Nandita Kumar

Project Adviser:

Professor Joshua Cuneo

Abstract

Our team was assigned to aid 7Factor, an Atlanta-based software development company, to improve upon a staffing tool application system developed by previous WPI students that tracks employees' hours. We aimed to encapsulate specific methods behind several different Auth0 roles. This was a necessary step to improve the overall security of the system and the potentially sensitive data within it. Our additions to the application restrict data so people in one role cannot access data from another.

Acknowledgments

The following people provided the opportunity to expand our skills and experience. Without their great help, we may not have succeeded.

Professor Joshua Cuneo (Project Advisor)
Jeremy Duvall (7Factor CEO)
Allen Brooks (Director of Software Engineering)

In addition, our work was built off the work of previous groups.

Keval Ashara
Othniel Bondah
Nicholas Li

Dyllan Cole
Noah Olson
Andrew Whitney
Evan Llewellyn

Panagiotis Argyrakis
Rafael Pimentel
Nicholas Wood

Table of Contents

List of Figures

Executive Summary

Software Integration

7Factor Software, headquartered in Atlanta, specializes in developing cloud architectures, custom systems, and applications for businesses that require technology that can keep pace as they continue to expand. They have implemented a staffing tool to enhance their operations and support their growth. In preparation for launch, the app's security was upgraded due to the sensitive nature of the data it will handle. Our contribution to the app was to create security roles for different user groups. Previously, all users had access to all data, regardless of their position within the company. Now, we have established distinct security roles for each user group, such as standard employees and project managers, limiting their access to only the required data. This ensures that sensitive data remains secure and only accessible to those who need it.

Design

The most crucial goal of our project was keeping data secure on the staffing application. Specifically, we wanted to change users from being able to access anything to only being able to access what they need to see. Without these measures, it would be easy for a malicious actor to access all the data in the app. Such a data breach would be incredibly detrimental to the company. Our group worked on the tool’s API to implement the new roles alongside new permissions. Roles and permissions were added using Auth0, a tool that makes setting up software security easy. Additionally, we began implementing a settings page on the application's front end, which can serve as the foundation for more features in the future.

Results

Our team achieved our aim of enhancing the security of the staffing tool. We implemented Auth0 roles as a means of boosting the application's security. First, we determined the roles to be incorporated and assigned the corresponding permissions. Following that, we employed an IDE to commence the coding process. We tested the security measures to guarantee their functionality. Once improvements to the system's overall security were made, our group began developing a settings page for the application. Our coding efforts were productive, successfully enhancing the application's security and providing the basis for a settings page that future teams can build upon.

Future Work

To improve the staffing tool for prospective users, a couple of key improvements could be made. The first pertains to how data is transferred between the front and back end. Currently, the data retrieved from the backend is directly serialized, which can lead to unnecessary data being sent to the front end. This issue will only worsen as the application grows, potentially slowing the connection between the two ends. To address this, DTOs should be added to limit the data sent during requests. While not an immediate need, this will become critical as the application expands. The second issue is the lack of a search feature, which would allow users to find specific data within the application quickly. This feature will become even more crucial as the amount of data collected grows. Implementing these two improvements will significantly benefit 7Factor over the long term.

In addition to these key improvements, other future recommendations include implementing user-friendly features such as profile picture uploads to personalize user accounts and storing employee skills for the Skill Search page. These enhancements would not only streamline user interactions but also improve the platform's overall usability and effectiveness, ensuring it remains a valuable tool for 7Factor and its users in the long term.

Introduction

Founded in 2016 by Atlanta-based entrepreneur Jeremy Duvall, 7Factor is a cloud-native software solutions company. The Staffing Tool simplifies time tracking for clients with diverse rates, projects, and employees. This MQP project expands upon previous work on the Staffing Tool by implementing Role-Based Access Control (RBAC), enhancing security and data integrity.

Background

7Factor

7Factor is a Georgia-based company dedicated to delivering exceptional software solutions to its clients. Their expertise lies in providing top-notch general software and cloud solutions to large, global companies. Notably, they have worked with clients such as Delta, HandyTrac, and Songsplits. The name "7Factor" is inspired by their seven missions, including "Teach and Elevate," "Automate Everything," "Do No Harm," "Curiosity," "Equality and Diversity," "Good Things," and "Love."

Previous Groups Accomplishments

Our team was the fourth group to contribute to the development of the staffing tool, and we were not the only ones working on it simultaneously. To ensure our success and avoid any complications, we collaborated closely with the other team involved in the project. Over the past three years, multiple teams have worked on the foundation of the tool we built upon. The first team created the base that served as the framework for the rest of the tool. The following team focused on improving the user interface and developed an analytics page. They aimed to make it easier to use and navigate while enhancing the tool's analytical capabilities. Lastly, last year’s team focused on creating a responsive design for mobile and tablet devices. They concentrated on adapting the analytics to mobile and tablet devices while ensuring the user interface was easy to navigate.

Previous Groups Future Work

The initial staffing tool team had several features that needed to be completed, with six crucial areas requiring attention. Their proposed solutions included the addition of front-end support for editing rate cards, employees, and projects, restricting the project list on the time entry page, integrating front and back-end systems, converting clients into a database object with front-end handling capabilities, enabling lazy-fetched objects on the front end, and implementing search employee and project views. The second team also had its own ideas, including adding Auth0 roles on the server side and implementing data transfer objects.

Methodology

Development Methodology

Our team opted not to follow a specific method of organization. Instead, we collaborated to develop a basic plan for moving forward. Our first step was determining how many roles were necessary and creating a week-by-week timeline detailing which tasks would be tackled and when various parts of the report would be completed. Once everyone was set up to do their respective tasks, we focused on getting the code up and running on each computer. The application was built using five tools: Java and JavaScript for coding, NodeJS and NPM for the front end, and Git for version control. To run the code, we used a docker container to manage the three repositories. As the project progressed, we encountered unexpected issues that required us to revise our timeline and delay some of our work.

Development Tools

GitHub

Our team utilized GitHub as a valuable tool to maintain our data. It acted as a version control system for our app-building process. Every time a team member made updates to the code, they would push it to the GitHub repository. The repository would then accept and record the changes or request assistance with merging them successfully. Since it recorded all changes, we could revert to any previous state if needed, though we were fortunate not to require this feature. GitHub proved an excellent tool as our team worked on the application alongside another team. It allowed for easy integration of the two groups' code, split into three repositories: UI, API, and database. These divisions were highly beneficial in our work as the other team worked on different application parts, minimizing the frequency of conflicting updates to the repositories.

Google Suite

Google Suite, also known as G Suite, played a vital role in our development process. It provided us with many tools that made collaborating much more manageable. All the files and documents were stored online, meaning anyone with the proper permissions could access them from anywhere. Our productivity was greatly enhanced thanks to G Suite. We started by creating a shared folder that all of us could edit. The content was organized into sections, with separate folders for written reports and previous MQP reports. Additionally, we had loose files, such as our timeline, which we created at the start of the project. By putting it in G Suite, we could update it as necessary without sending it to the whole team. The folder containing the report's components was essential as it allowed us to keep track of each other's progress and stay on schedule. Lastly, we also had a folder with last year's reports, which made it easy to access all the information we needed in one place.

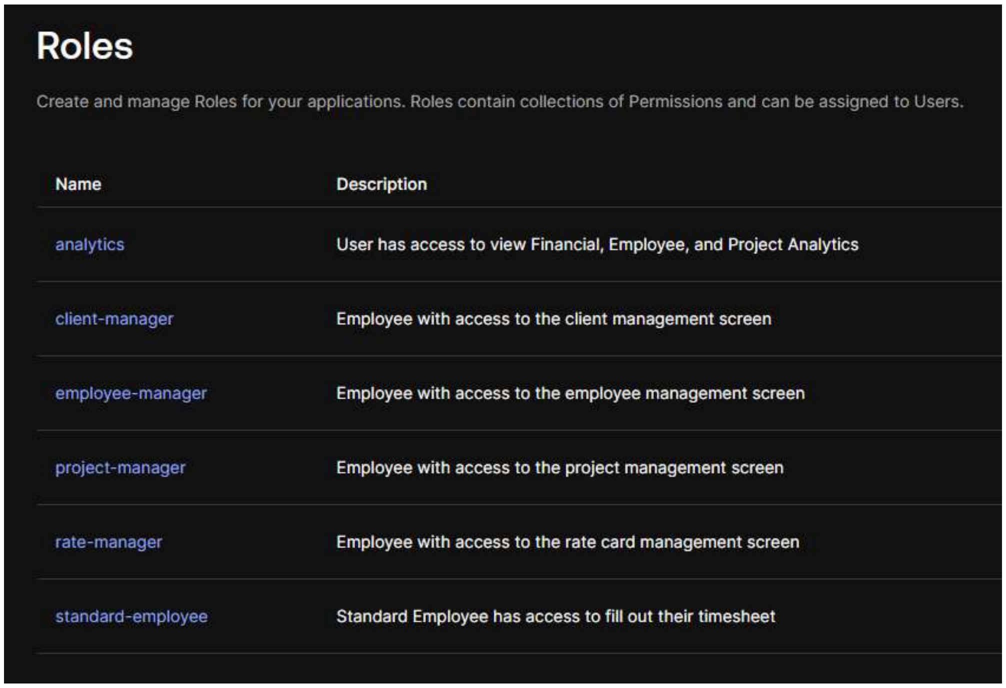
Software Tools

Auth0

To secure the app, our group used Auth0 as security. Auth0 is an identity access management provider. Identity access management providers are a framework for ensuring users access to what they should have access to, no more and no less. Auth0 lets us group users into different roles with different permissions for each role. This accomplished the primary goal of the project. Assigning roles to each user meant that we would not need to create permissions for each user, and any newly added users could easily have their roles added. Administrators can also mix and match roles to create users with multiple roles. Auth0 enhances security by attaching a token to every message from the front end to the back end. The token is verified in the backend, which evaluates whether the request is accepted or rejected.

Results

Our primary goal for this project was to improve the application's current security measures. We were given six different Auth0 roles, five of which corresponded to a particular management screen within the application. The Auth0 roles can be seen in the figure below.



Roles

Create and manage Roles for your applications. Roles contain collections of Permissions and can be assigned to Users.

Name	Description
analytics	User has access to view Financial, Employee, and Project Analytics
client-manager	Employee with access to the client management screen
employee-manager	Employee with access to the employee management screen
project-manager	Employee with access to the project management screen
rate-manager	Employee with access to the rate card management screen
standard-employee	Standard Employee has access to fill out their timesheet

Figure 1: Auth0 Roles

For most of our project duration, we focused on implementing Role Based Access Control (RBAC) within the application. In order to implement RBAc, we needed to create new permissions to check for when making requests to the API. The first step in our development process was to create these permissions necessary for protecting our methods.

Creating Permissions

Our initial strategy for implementing RBAC involved the incorporation of new permissions into the Staffing-Tool-Api through Auth0. Within the API's REST controllers, individual methods are created to facilitate handling requests pertaining to getting, editing, creating, or deleting content from the database. For each of these actions we needed to create a permission to limit access to said action. Each permission we added aligned with methods/actions within these REST controllers.

- **create**: Indicates the ability to create a new resource instance. For example, "create:employee" allows a user to create a new employee entry in the system, "create:client" allows creating a new client, and so on. Users with this permission can initiate adding new data into the system.

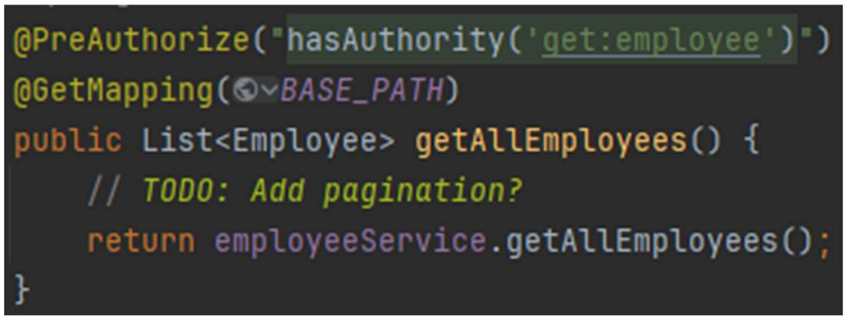
- **edit**: Indicates the ability to modify an existing resource instance. For instance, "edit:employee" allows editing the details of an existing employee, "edit:client" permits modifying client information, and so forth. Users with this permission can update the existing data in the system.

- **get**: Indicates the ability to retrieve or fetch information about a resource. "get:employee" allows fetching details of employees, "get:client" retrieves client information, and similar for other resources. Users with this permission can access existing data from the system.

- **delete**: Indicates the ability to remove an existing resource instance. For example, "delete:project" allows deleting a project from the system, "delete:rate" permits removing a rate entry, and so on. Users with this permission can initiate the deletion of data from the system.

Implementing Permissions

After establishing the necessary permissions, we implemented the `'@PreAuthorize'` annotation within our REST Controller methods. This annotation regulates access by stipulating the requisite permissions for each method, thereby enhancing overall security. We systematically incorporated the `'@PreAuthorize'` annotation for each REST Controller to enforce access control, ensuring that only users with the appropriate permissions could execute specific actions within the API. This approach effectively governs user privileges, bolstering the security framework of our system. Below is an example of the `'@PreAuthorize'` annotation implemented into the `EmployeeController.`



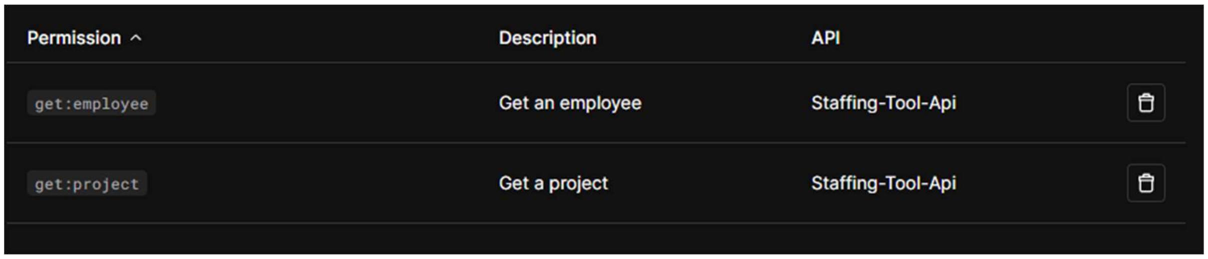
```java
@PreAuthorize("hasAuthority('get:employee')")
@GetMapping(BASE_PATH)
public List<Employee> getAllEmployees() {
    // TODO: Add pagination?
    return employeeService.getAllEmployees();
}
```

Figure 2: `'@PreAuthorize'` *Example*

After adding the `'@PreAuthorize'` annotation to all REST Controllers and their methods, we then determined the permissions needed for each user role. This involved carefully analyzing what different roles should be allowed to do in the system. The following sections will delve into the necessary permissions for each role within the system.

Analytics Role

This role allows users to view financial, employee, and project analytics, allowing them to analyze crucial data for informed decision-making.



Permission ^	Description	API
`get:employee`	Get an employee	Staffing-Tool-Api
`get:project`	Get a project	Staffing-Tool-Api

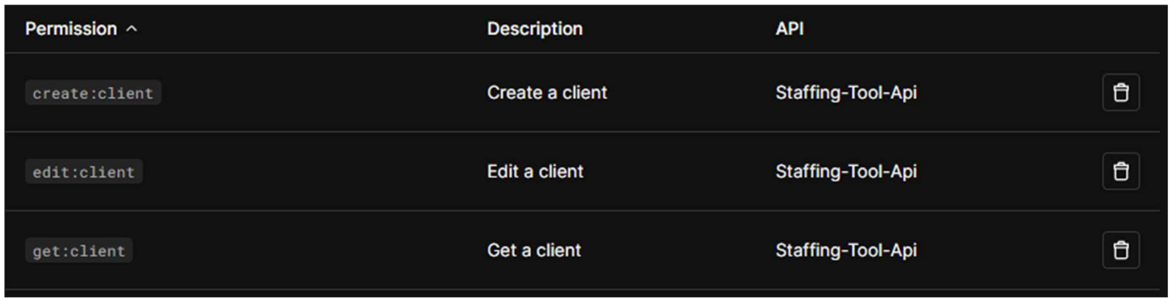
Figure 3: Analytics Permissions

Permissions:

- **get:employee**: Allows access to all employee information.
- **get:project**: Allows access to all projects and their details.

Client Manager Role

Users assigned to this role have access to the client management screen, enabling them to manage client information efficiently within the application.



Permission ^	Description	API
`create:client`	Create a client	Staffing-Tool-Api
`edit:client`	Edit a client	Staffing-Tool-Api
`get:client`	Get a client	Staffing-Tool-Api

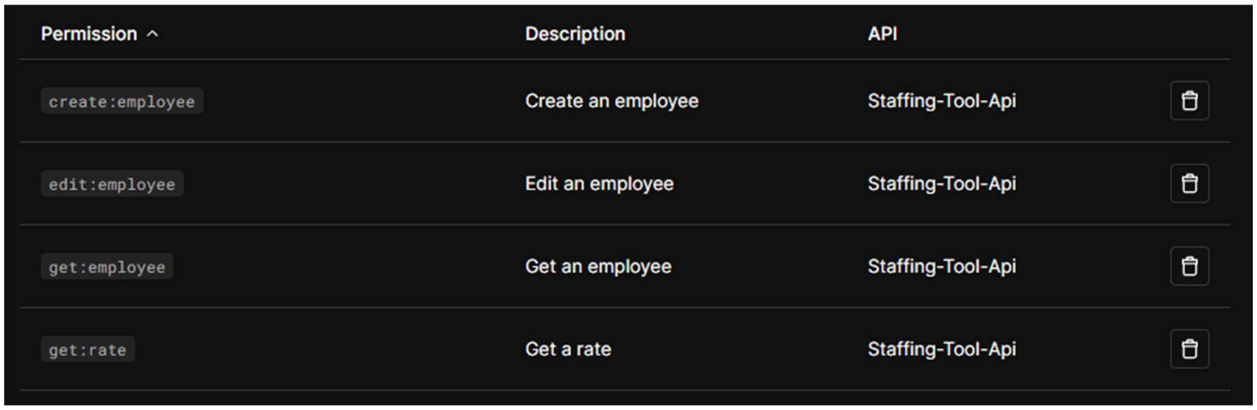
Figure 4: Client Manager Permissions

Permissions:

- **get:client:** Provides access to all client information.
- **create:client:** Grants permission to create new client entries in the system.
- **edit:client:** Enables modifying existing client data.

Employee Manager Role

This role provides employees with access to the employee management screen, allowing them to oversee and manage employee-related tasks and information effectively.



Permission ^	Description	API	
`create:employee`	Create an employee	Staffing-Tool-Api	
`edit:employee`	Edit an employee	Staffing-Tool-Api	
`get:employee`	Get an employee	Staffing-Tool-Api	
`get:rate`	Get a rate	Staffing-Tool-Api	

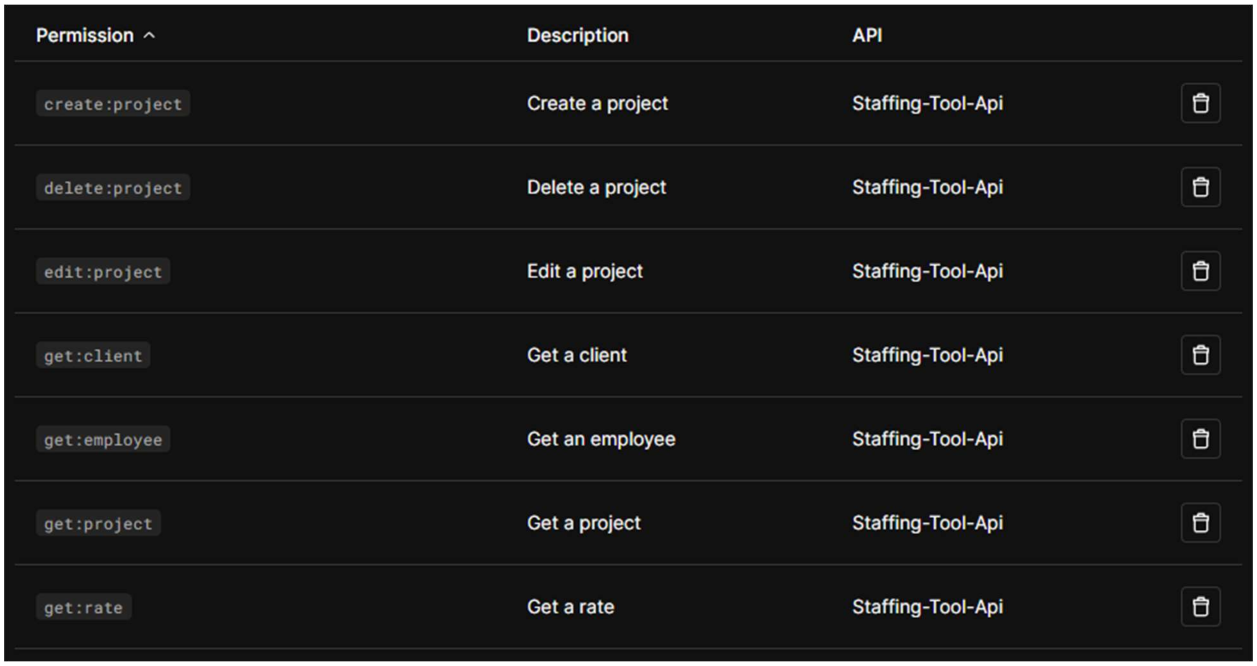
Figure 5: Employee Manager Permissions

Permissions:

- **create:employee:** Grants permission to create new employees in the system.
- **edit:employee:** Enables modifying existing employee data.
- **get:employee:** Allows access to all employee information.
- **get:rate:** Allows access to all rate card information.

Project Manager Role

Employees assigned to this role have access to the project management screen, empowering them to manage projects efficiently and oversee project progress.



Permission ^	Description	API
`create:project`	Create a project	Staffing-Tool-Api
`delete:project`	Delete a project	Staffing-Tool-Api
`edit:project`	Edit a project	Staffing-Tool-Api
`get:client`	Get a client	Staffing-Tool-Api
`get:employee`	Get an employee	Staffing-Tool-Api
`get:project`	Get a project	Staffing-Tool-Api
`get:rate`	Get a rate	Staffing-Tool-Api

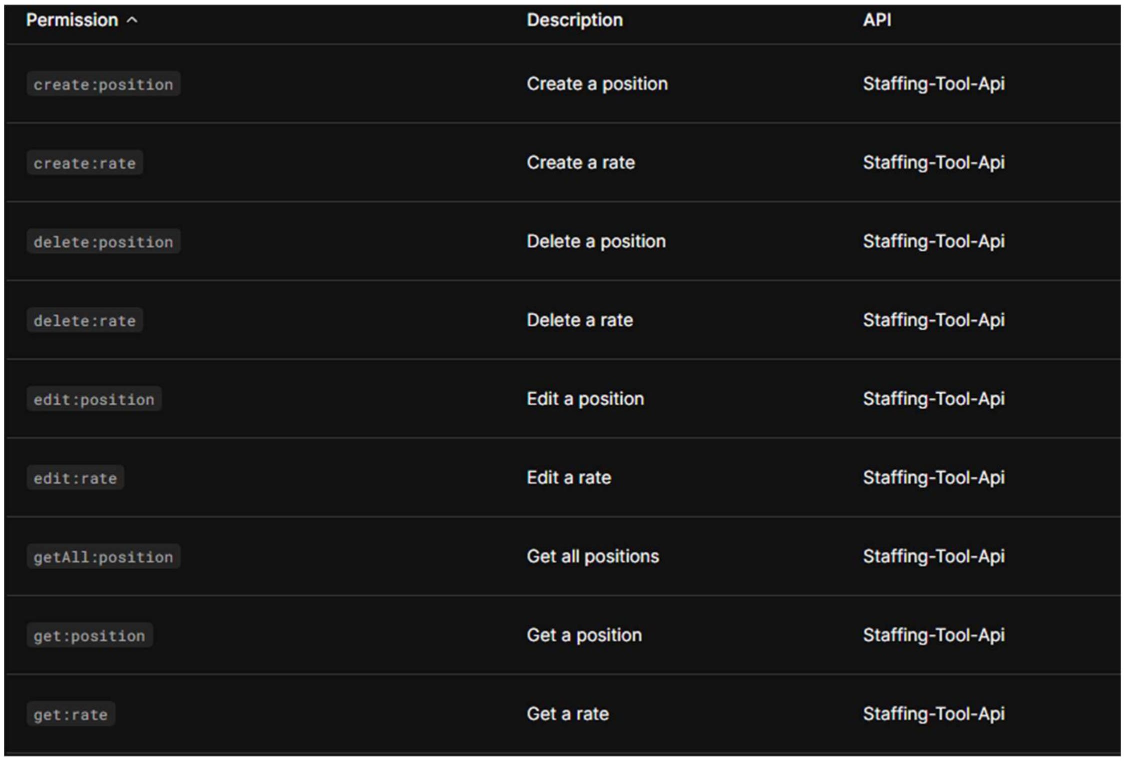
Figure 6: Project Manager Permissions

Permissions:

- **create:project:** Grants permission to create new projects within the system.
- **delete:project:** Allows deletion of projects from the system.
- **edit:project:** Enables modification of existing project details.
- **get:client:** Provides access to all client information.
- **get:employee:** Allows access to all employee information.
- **get:project**: Allows access to all projects and their details.
- **get:rate:** Allows access to all rate card information.

Rate Manager Role

Users with this role can access the rate card management screen, enabling them to manage and update rate card information within the application.



Permission ^	Description	API
`create:position`	Create a position	Staffing-Tool-Api
`create:rate`	Create a rate	Staffing-Tool-Api
`delete:position`	Delete a position	Staffing-Tool-Api
`delete:rate`	Delete a rate	Staffing-Tool-Api
`edit:position`	Edit a position	Staffing-Tool-Api
`edit:rate`	Edit a rate	Staffing-Tool-Api
`getAll:position`	Get all positions	Staffing-Tool-Api
`get:position`	Get a position	Staffing-Tool-Api
`get:rate`	Get a rate	Staffing-Tool-Api

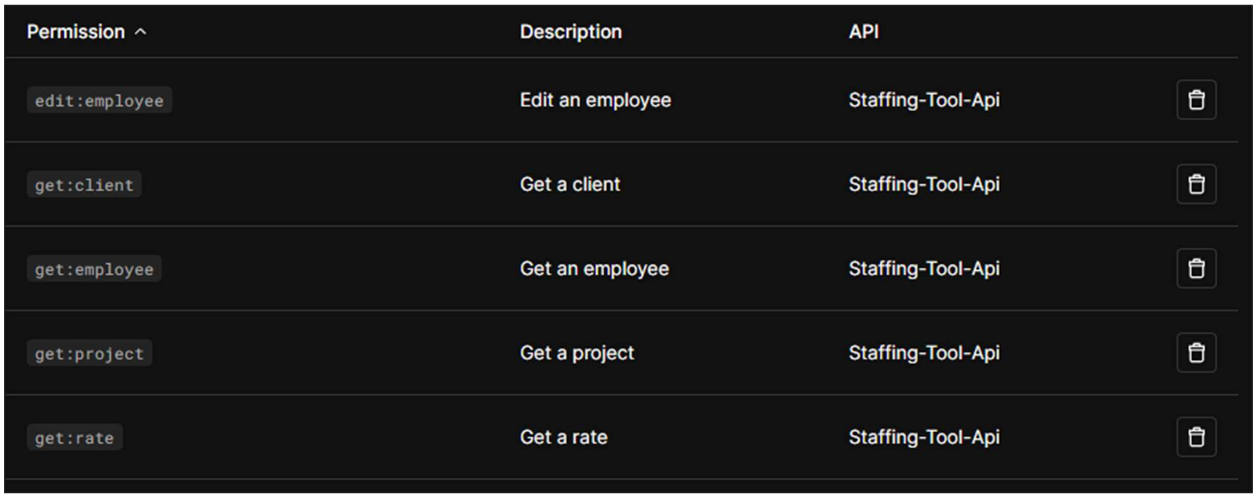
Figure 7: Rate Manager Permissions

Permissions:

- **create:position:** Allows users to add new positions to the system.
- **create:rate:** Allows users to add new rate cards to the system.
- **delete:position:** Allows users to delete existing positions.
- **delete:rate:** Allows users to delete existing rate cards.
- **edit:position:** Allows users to modify the details of existing positions.
- **edit:rate:** Allows users to update the details of existing rate cards.
- **get:position:** Grants users access to all positions.
- **get:rate:** Grants users access to all rate cards.

Standard Employee Role

Standard employees have access to fill out their timesheets, allowing them to log their work hours and activities within the system accurately.



Permission ^	Description	API	
`edit:employee`	Edit an employee	Staffing-Tool-Api	
`get:client`	Get a client	Staffing-Tool-Api	
`get:employee`	Get an employee	Staffing-Tool-Api	
`get:project`	Get a project	Staffing-Tool-Api	
`get:rate`	Get a rate	Staffing-Tool-Api	

Figure 8: Standard Employee Permissions

Permissions:

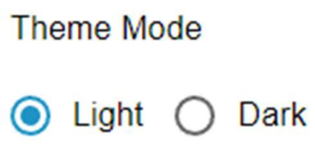
- **edit:employee:** Enables modifying existing employee data.
- **get:client:** Provides access to all client information.
- **get:employee**: Allows access to all employee information.
- **get:project**: Allows access to all projects and their details.
- **get:rate:** Allows access to all rate card information.

User Settings Page

Once we had successfully implemented all of the permissions for each role, our next goal was to implement a page so that users could make personal changes to the application and their employee profile. Our first step in this process was to create a new page on the UI front end of the application. This page would serve as the foundation for various settings that users can tweak to their liking.

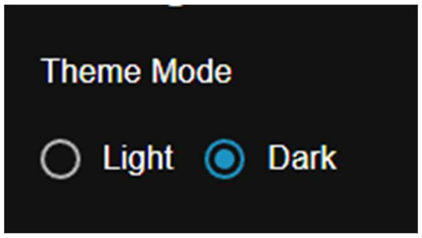
Light/Dark Themes

The first setting we implemented into the application was the ability to switch between a light and a dark theme. This feature offers users a customizable visual experience aligned with their preferences.


Theme Mode

[✓ Light] Dark

Figure 9: Theme Mode Toggle (Light)


Theme Mode

Light [✓ Dark]

Figure 10: Theme Mode Toggle (Dark)

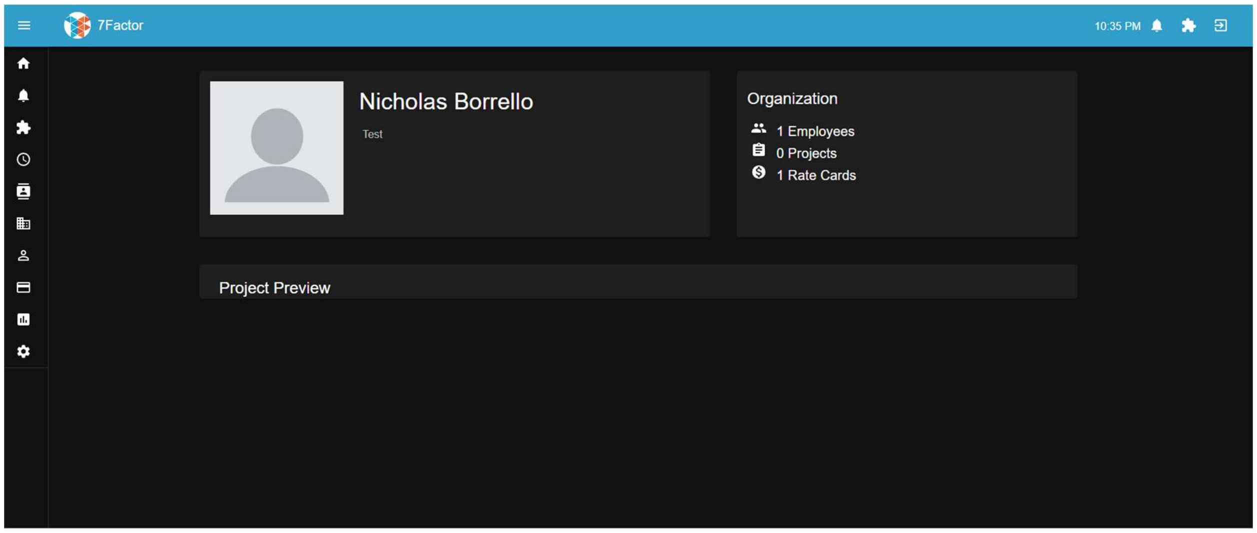
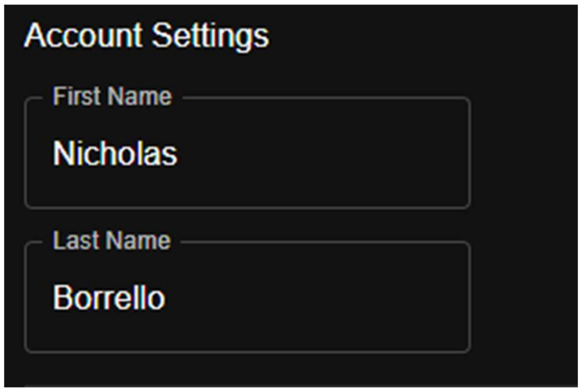


Figure 11: Home Page with Dark Theme

Account Settings

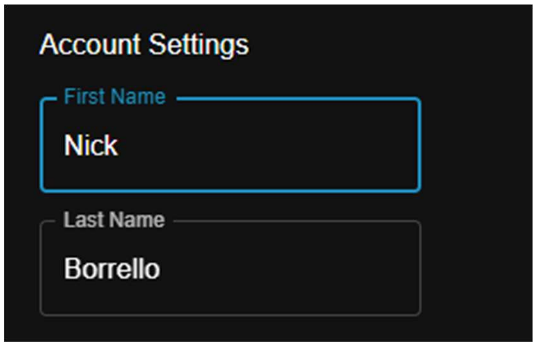
Following the implementation of the theme mode toggle, we introduced an account settings section to allow users to personalize their profile information. Users can conveniently modify their first and last names within this section to ensure their accounts reflect accurate information. In future project iterations, we hope that teams will build upon this section by providing users with even more personal settings, such as profile pictures.


Account Settings

First Name: Nicholas

Last Name: Borrello

Figure 12: Account Settings (Name Before)


Account Settings

First Name: Nick

Last Name: Borrello

Figure 13: Account Settings (Name After)

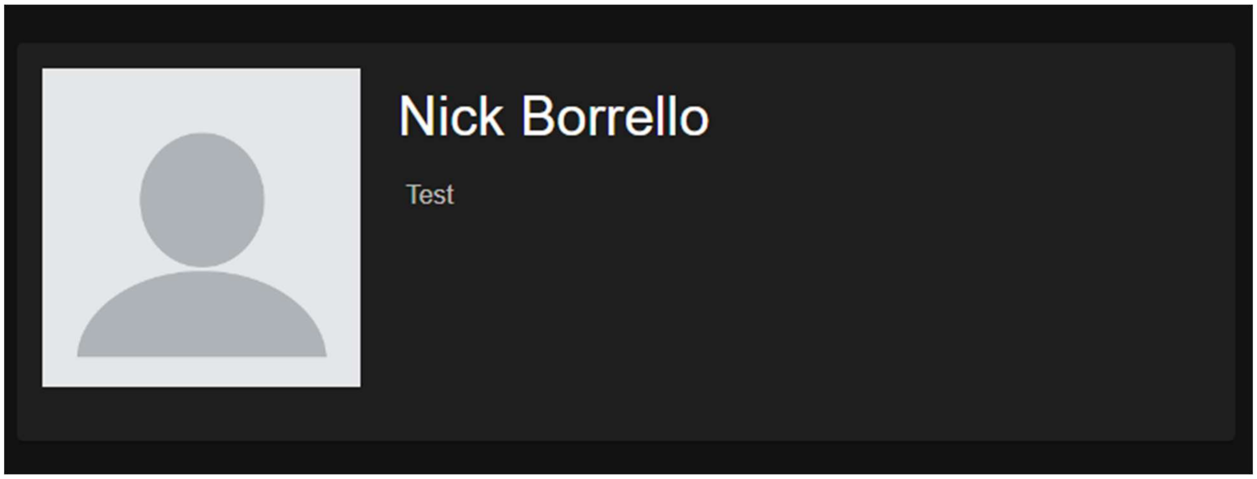


Figure 14: Updated name on Home Page (After change)

My Skills

The final setting we introduced was *My Skills*. This setting allows users to manage their skill sets directly within the settings page. This feature will enable users to customize their skillset by adding or removing skills as needed, allowing the Skill Search tool to be as up-to-date and accurate as possible. It is worth noting that this feature currently does not save the skills to the database. Future teams should consider implementing some method of storing an employee's skills within the database.

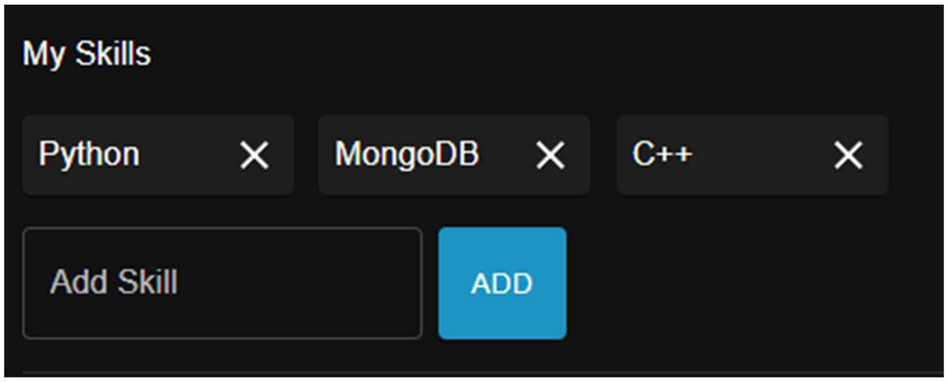


Figure 15: My Skills Setting

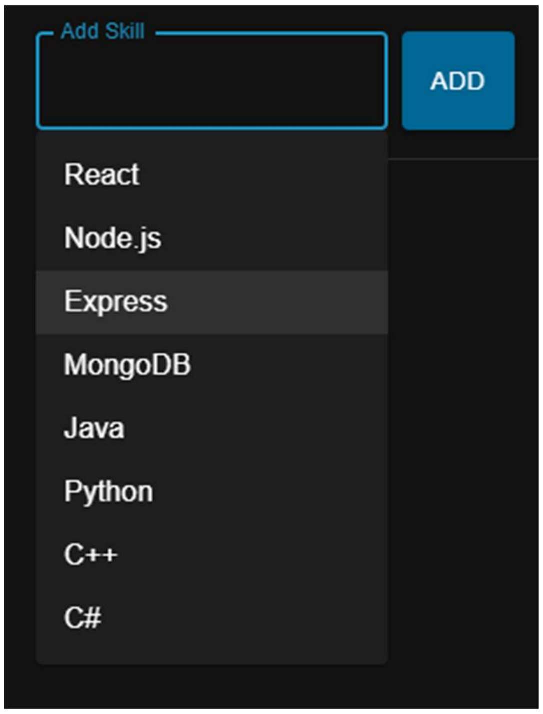


Figure 16: My Skills (Adding Skills)

Recommendations and Future Work

Due to time constraints, certain essential areas of development were not fully addressed during our project. However, it is crucial to acknowledge and credit the previous team for laying a solid foundation for the Staffing Tool platform. Their efforts provided us with a robust starting point for our work.

One area identified by the previous team was the inefficiency in data transfer, where complete server-side objects were sent for each API request, even when only specific subsets of information were required. To enhance performance, they recommended implementing Data Transfer Objects (DTOs) to send only relevant data, reducing user burden and improving efficiency. This optimization streamlines communication and enhances platform responsiveness effectively.

Improving the user interface was another area highlighted by the previous team. While the current platform operates smoothly on all devices, there is potential to enhance user experience further. They proposed integrating subtle animations such as hover impacts for cards and immersive page transitions in the sidebar and toolbar to improve user interaction. These visual enhancements could elevate customer satisfaction and make the platform more engaging.

While these recommendations provide a clear roadmap for future development, there are additional suggestions we would like to incorporate. First, adding a Profile Picture feature to the Account Settings would personalize the user experience and enhance user identification within the platform. Additionally, further implementing the *My Skills* setting to support storing skills for each employee in the database would greatly benefit the *Skill Search* feature. We encourage upcoming teams to build upon these suggestions, tailor them to project objectives and incorporate user feedback to elevate the Staffing Tool's capabilities further.

Conclusion

Our team has accomplished a significant feat by integrating Auth0 roles into 7Factor's staffing tool. Initially, this application had already undergone three years of development, leaving it partially completed. However, we utilized this as a starting point and planned and executed Auth0 roles to enhance the application's functionality. As a result, the overall security of the project has been dramatically improved, and users now have access to a settings page, bringing us one step closer to achieving our goals.

Bibliography

7Factor. (2024, January 3). Quality Engineered Custom Software Solutions | Cloud Native | 7Factor. 7Factor Software. https://www.7factor.io/

Auth. (n.d.). Auth0 overview. Auth0 Docs. https://auth0.com/docs/get-started/auth0-overview

Java documentation - get started. Oracle Help Center. (2023, January 31). https://docs.oracle.com/en/java/

MozDevNet. (n.d.). JavaScript: MDN. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript

Node.js V21.6.2 documentation. Index | Node.js v21.6.2 Documentation. (n.d.). https://nodejs.org/docs/latest/api/

Slamic, Tadej. “AUTH0.” Auth0, 7 Oct. 2021, auth0.com/blog/amp/spring-boot-authorization-tutorial-secure-an-api-java/.

Appendix

Setting Up Your Staffing Tool Development Environment

Software Requirements:

To get started, you will need the following software installed on your system:

- **Docker (4.12.0):** Used for containerized development environments.
- **Git (2.39.2):** Version control system for managing code changes.
- **Java (version 8, jdk 1.8.0_341):** Programming language used for the platform backend.
- **Node.js (v16.17.0):** JavaScript runtime environment for the platform frontend.
- **NPM (8.15.0):** Package manager for installing Node.js dependencies.

For Ubuntu Users:

If you are using Ubuntu Linux, you can install all the required software in one go using the following command:

```
sudo apt-get install docker npm git nodejs
```

Running the Development Environment:

1. **Acquire the code:** https://github.com/jackson-lundberg/7factorStaffingToolTeam2

2. **Start the environment:** We must start the database, API, and UI components in that order.
    - **Database:**
        - Open the `db` folder in your terminal.
        - Run the command: `./env/_up.sh`
    - **API:**
        - Open the `api` folder in your terminal.
        - Run the appropriate command based on your operating system:
            - Windows: `./gradlew.bat bootRun`
            - Other: `./env/_up.sh`
    - **Frontend:**
        - Open the `UI` folder in your terminal.
        - Navigate to the `src` directory: `cd src`
        - Install dependencies: `npm install`
        - Start the frontend server: `npm start`

Accessing the Development App:

- If everything is set up correctly, the frontend should launch in a new browser tab. This is the development build of the Staffing Tool.
- To fully access all features, you will need OAuth credentials from 7Factor. Once you have those, you can start developing and testing the platform.