Real-Time Workforce Allocation
–
Technical Documentation

Table of Contents

1. Introduction

Welcome to the technical documentation for the Real-Time Workforce Allocation (RTWA) application. RTWA is a powerful tool designed to streamline the management of workforce shortages and surpluses in an efficient and real-time manner.

1.1. Overview of RTWA:

RTWA facilitates the effective allocation of human resources within the organization by allowing users to post surplus or demand for workers from their respective departments. Whether it's a surplus of employees available for temporary assignment or a need for additional workers to fill shortages, RTWA provides a platform for seamless communication and coordination between departments.

1.2. Key Features:

- *Surplus Management:* Users can post surplus employees from their department, indicating the number of employees available, along with details such as date, time, shift, functions, competences, and department.

- *Demand Management:* Users can post demands for additional workers, specifying the number of employees required, along with relevant details like date, time, shift, functions, competences, and department.

- *Real-Time Updates:* The application ensures real-time updates, allowing users to instantly view and respond to surplus or demand postings.

- *Efficient Coordination:* RTWA facilitates efficient coordination between departments, enabling them to quickly fulfill surplus or demand requirements without delay.

- *Enhanced Resource Utilization:* By effectively managing workforce allocations, RTWA optimizes resource utilization within the organization, minimizing idle time and maximizing productivity.

1.3. Target Audience:

This documentation is intended for developers, testers, stakeholders, and any other individuals involved in the development, deployment, and maintenance of the RTWA application.

1.4. History:

Throughout this documentation, we will provide detailed insights into the architecture, functionality, deployment, and maintenance of the different versions of the RTWA application.

1.5. [Version 0.1.0]:

In this version the main focus was adding the email service and app authorization. Other changes include updating the popup form fields with data stored in the database, creating separate environments for development, staging and production, bug fixes and other small improvements in the UI and code refactoring.

- Added email notification service
- Added user access levels authorization
- Created of 3 separate environments (development, staging, production)
- Updated the popup form fields with information from the database
- Added form validation for the popup fields
- Modified the history database tables to include accepted workers
- Deleting the entries from the RequestTable when the number of employees in the offer is exhausted
- Added constraints, new tables and updated database to accommodate the new features
- Explored ways to implement future updates to the application (automated deployment, IDM integration, etc)
- Reload and redirect to the offer/demand pages after accepting workers
- UI improvements (back button on details page, displaying user role in side menu, etc)
- Updated documentation

2. Project Structure

2.1. Frontend

2.2. Backend

2.3. Database

3. Features

3.1. E-mail Service

The Email Notification Service within the RTWA (Real-Time Workforce Allocation) application facilitates the automated sending of email notifications to relevant stakeholders upon the creation of offer or demand personnel entries. This service enhances communication and coordination within the organization by promptly informing designated recipients about surplus personnel availability or demand.

3.1.1. Functionality:

Upon the creation of a new surplus personnel entry through the RTWA application, the Email Notification Service triggers the generation and dispatch of email notifications to predefined recipients. These notifications contain essential details regarding the surplus personnel entry, including the creator's name, department, shift, functions, competences, start date, end date, total days, and total employees.

3.1.2. Implementation:

To send email notifications to targeted groups for each entry created, the SendMailNotification function is invoked from the RequestTables controller. The necessary variables for this function are defined within the CreateSurplus function in the same file.

Initially, we must establish the required variables for the email service (*From, SendTo, CC, Subject, Body, IsBodyHtml, Server*) as parameters. Subsequently, these variables will be initialized within the CreateSurplus function, where the SendMailNotification function will be executed.

3.1.3. Parameters:

The SendMailMessage() function accepts the following parameters:

- *From:* Sender's email address.
- *SendTo:* Recipient's email address(es).

- *CC:* Carbon copy recipient's email address(es) (optional).
- *Subject:* Subject line of the email notification.
- *Body:* HTML-formatted body content of the email notification.
- *IsBodyHtml:* Boolean flag indicating whether the body content is in HTML format.
- *Server:* SMTP server address for email transmission.

3.1.4. Security:

To ensure secure email transmission, the Email Notification Service utilizes SSL encryption when communicating with the SMTP server. Additionally, credentials are provided for SMTP server authentication to prevent unauthorized access to the email sending functionality.

Detailed configuration and management of email settings, including sender, recipient, and SMTP server details, are maintained within the application's database and can be updated as required to accommodate organizational changes or preferences.

3.2. User Access Level Restrictions (Authorization)

The User Access Level Restriction feature within the RTWA (Real-Time Workforce Allocation) application enables the enforcement of role-based access control, restricting users' access to specific functionalities based on their assigned roles.

3.2.1. Functionality:

This feature ensures that users can only access functionalities within the application that are permitted by their assigned roles. Users are assigned roles such as "viewer", "key_user", “standard_user” or "admin," and access to certain features is determined by these roles.

3.2.2. Implementation Overview:

- *ASP.NET API (AccountController):* The AccountController exposes an endpoint /api/account/userRoles that retrieves user roles based on the account ID. When a request is made to this endpoint with a valid account ID, the controller fetches the corresponding user roles from the database and returns them to the client. If user roles are found, they are returned with an HTTP status code 200 (OK). Otherwise, an appropriate error message is returned.

- *Angular Service (UserRolesService):* The UserRolesService in the Angular project is responsible for making HTTP requests to the /api/account/userRoles endpoint of the ASP.NET API. It provides a method getUserRoles(accountId: string) that accepts an account ID as a parameter and returns an Observable of string array containing user roles. This service is injected into components where role-based access control is required.
- *Angular Components:* The AppButtonComponent is an example component where role-based access control is implemented. Upon initialization (ngOnInit), the component retrieves the user's account ID from the session storage and calls the getUserRoles method of the UserRolesService to fetch the user's roles. Based on the user's role, certain functionalities (such as opening a popup) may be enabled or disabled. In the provided example, the button is disabled, and a tooltip is displayed if the user's role is "viewer."

3.2.3. Usage:

Developers can implement role-based access control in various components and functionalities of the application by leveraging the UserRolesService to retrieve user roles and conditionally enable or disable features based on the user's role.