TAS- System Architecture Overview

2024-04

Note :

“This document outlines the main components of the system, their interactions, the technologies used, and key features such as security measures and integration points.” Development team

Table of contents

1. DOCUMENT VERSION

VERSION	DATE	CHANGE	AUTHOR
1.0.0.	2024-04-02	First version	DAVKHARBAYAR M
1.0.1	2024-04-09	System Components update	DAVKHARBAYAR M

2. TITLE

PROJECT CODE **: TAS**

PROJECT NAME **: OTTAS**

3. INTRODUCTION

In the dynamic and demanding environment of the mining industry, managing the logistics of mine workers' rosters, accommodation, and travel involves intricate coordination and meticulous attention to detail. The TAS system emerges as an essential tool, specifically developed to simplify these processes by providing a centralized platform for handling all aspects of mine worker management.

Key Features

Employee Registration and Management: At the core of the **TAS** system is its ability to facilitate the easy registration of new employees and manage ongoing changes to employee profiles. This ensures that all worker details are accurately maintained and readily accessible.

Roster Management: The system offers comprehensive roster management capabilities, allowing for the efficient organization of work schedules. It supports various roster patterns and can accommodate changes in real-time, ensuring that staffing levels are optimized to meet operational demands.

Accommodation Management: TAS system provides a seamless process for managing accommodation for mine workers, from booking to check-out. Whether it's for short-term projects or long-term assignments, the system can handle multiple accommodation types and preferences, ensuring workers have suitable living arrangements that comply with their needs and company policies.

Travel Management: Handling both domestic and international travel requests, the TAS system simplifies the travel booking process. It supports the planning, approval, and management of travel itineraries, ensuring that employees reach their destinations safely and efficiently. The system accommodates a variety of travel preferences and requirements, including transportation, lodging, and itinerary planning.

Report Management System

A key component of the TAS system's comprehensive feature set is the dynamic Report Management System. This subsystem is engineered to cater to the diverse reporting needs of the mining industry, offering customizable reporting solutions that provide insights into roster management, accommodation bookings, and travel arrangements.

Dynamic Reporting Capabilities

The dynamic nature of the Report Management System allows users to generate reports based on real-time data, ensuring that decision-makers have access to the most current information. With its user-friendly interface, the system supports the creation of customized reports tailored to specific requirements, enabling the analysis of various aspects of mine worker management, such as:

Roster Analysis: Detailed reports on roster patterns, employee availability, shift coverage, and compliance with working hour regulations.

Accommodation Utilization: Insights into accommodation usage, occupancy rates, and costs, helping to optimize accommodation management and planning.

Travel Logistics: Comprehensive overviews of travel arrangements, including costs, preferences, and compliance with travel policies.

Employee Management: Reports on employee registrations, profile changes, and other HR-related information, providing a clear view of the workforce dynamics.

4. System Components

4.1 TAS-API

URL: https://api.ottas.corp.riotinto.org

Description: The main instance running on an IIS server, responsible for handling business logic and serving as the backend API.

Technology: .NET Core 6.0 C#, implementing a clean architecture pattern.

Key Features: SSL certificate for secure HTTPs communication, REST API for interaction with the FrontEnd and report server, and integration with Active Directory for user rights management.

Database: SQL Server 2019, hosted on tasdb. Provides data storage and retrieval for API operations.

4.2 TAS-REPORT

URL: https://reportapi.ottas.corp.riotinto.org

Description: A dedicated .NET Core web API report server running on an IIS server. Generates and manages reports.

Technology: .NET Core 6.0 C#, with a focus on reporting capabilities.

Key Features: SSL certificate for secure communication. Utilizes a separate SQL Server for report data storage.

Database: Dedicated SQL server for the report database, ensuring isolated management and scalability of report data.

4.3 FrontEnd System

URL: https://ottas.corp.riotinto.org

Port: 80 (HTTP, with an assumption of HTTPS redirection for secure communication)

Description: The user interface for interaction with the TAS-API.

Technology: ReactJS, known for its efficiency and flexibility in building interactive UIs.

Key Features: Communicates with TAS-API via a REST API. Optional use of SignalR for real-time bi-directional communication between the FrontEnd and TAS-API. SSL certificate for secure communication.

4.4 Database Servers

TASDB (Main Database): Hosted on SQL Server 2019, provides data storage for TAS-API. Ensures data integrity and supports complex queries.

Report Database: Separate SQL Server instance for storing and managing report data. Enhances performance and security for reporting functions.

Port: 1433 (default port for SQL Server)

4.5 Email Service

Port: 25 (SMTP for email sending)

Description: The email service facilitates communication with users, sending automated notifications regarding roster changes, travel arrangements, and other system alerts. It plays a crucial role in keeping all stakeholders informed.

Security Considerations: Despite using port 25, the system should implement SMTP authentication and encryption (where possible) to secure email communications.

5. SYSTEM AND LOGICAL ARCHTECTURE

The system architecture of the TAS system is structured to ensure reliability, scalability, and security, addressing the complex needs of managing mine workers. It consists of several key components and services, each designed to handle specific aspects of the system's functionality:

Components:

Web Servers: Hosts the TAS-API and TAS-REPORT, serving as the system's backbone by processing business logic, handling API requests, and generating reports.

Database Servers: Two SQL Server instances—one for the main TAS database (tasdb) and one for the TAS reports database. These servers store, retrieve, and manage all data related to rosters, accommodation, travel, and reports.

Frontend Application: Built with ReactJS, providing a user-friendly interface for interacting with the TAS system. It communicates with the backend via RESTful APIs.

Active Directory Integration (Optional): For managing user access and rights, leveraging existing corporate AD infrastructure for authentication and authorization.

Email Notification System: Automated email communications through an internal mail service, supporting notifications for roster changes, travel arrangements, and system alerts.

Real-time Communication Service: Utilizes SignalR for real-time updates between the frontend and backend, enhancing user interaction and system responsiveness.

Logical Architecture:

The logical architecture focuses on the TAS system's data flow and processing logic, illustrating the interaction between its components:

User Interface (UI) Layer: The frontend application, through which users interact with the system. It sends and receives data from the backend via secure, RESTful API calls over SSL.

Application Layer: Contains the TAS-API logic, handling requests from the UI layer, executing business logic, and interacting with the database. It also includes the TAS-REPORT service for generating dynamic reports.

Data Access Layer: Manages communication with the database servers, abstracting the underlying database operations from the application logic. Ensures efficient data retrieval and storage.

Database Layer: Comprises the SQL Server instances where all system data is stored, including user information, rosters, accommodation details, travel plans, and report data.

Security and Authentication: Implemented at multiple layers, including SSL encryption for data in transit, Active Directory integration for authentication, and role-based access control within the system to ensure data integrity and security.

Notification and Real-time Communication: The email notification system and SignalR service are integrated across the architecture, providing timely updates and facilitating synchronous communication between the frontend and backend.

Conclusion:

The TAS system’s architecture is designed to be robust, secure, and capable of handling the complexities of roster, accommodation, and travel management for mine workers. By separating concerns across different layers and ensuring secure data exchange and real-time interactions, the system provides a comprehensive solution that meets the operational and management needs of the mining industry. This architecture not only supports current functionalities but also offers the flexibility to incorporate future enhancements and integrations.

6. PHYSICAL ARCHTECTURE

Browser Compatibility: The TAS application is designed to be compatible with a range of web browsers, including Google Chrome, Microsoft Edge, Opera, and Mozilla Firefox. This ensures that users can access the application from their preferred web browser without compatibility issues.

Client-Server Communication:

Users interact with the TAS application through their web browser. When they perform an action (e.g., submitting a form to register a new travel request), the browser sends this request to the server using the HTTP/HTTPS protocol.

HTTPS is preferred for its secure encryption of data in transit, protecting sensitive information such as login credentials and personal data.

Processing Requests on IIS:

The request from the client's browser is received by the Application Server running IIS (Internet Information Services), Microsoft's extensible web server software. The version specified for the TAS application is IIS 10.0 or higher, chosen for its reliability, manageability, and scalability.

IIS processes the incoming request in a sequential order. It performs the necessary actions required by the request, which often involves CRUD (Create, Read, Update, Delete) operations on the database.

The IIS server interacts with the database to execute these operations, retrieving or updating data as needed by the application logic.

Response to Client:

Once the IIS server completes the requested action and any database interactions, it compiles a response. This response is then sent back to the client's browser using the HTTP/HTTPS protocol.

The web browser receives the response and processes the data to present the results to the user. This could be the confirmation of a successful operation, the requested data displayed in the user interface, or an error message if something went wrong.

Request-Response Pattern:

This interaction between the client and server follows the request-response messaging pattern, a foundational concept of web applications. The client (web browser) sends a request to the server, the server processes the request, and then the server sends back a response.

This sequence of message exchanges continues with each user action until all requests are fulfilled, culminating in the presentation of the requested data or service outcome to the user.

This interaction between the client and server follows the request-response messaging pattern, a foundational concept of web applications. The client (web browser) sends a request to the server, the server processes the request, and then the server sends back a response.

This sequence of message exchanges continues with each user action until all requests are fulfilled, culminating in the presentation of the requested data or service outcome to the user.