9/11/2023

Celumusa Zulu

The software dude

Time Sheet System Architectural Document

Version: 1.0

Contents

[1. Introduction 3](#_Toc145350400)

[1.1 Purpose 3](#_Toc145350401)

[1.2 Scope 3](#_Toc145350402)

[1.3 Document Overview 3](#_Toc145350403)

[2. System Architecture Overview 3](#_Toc145350404)

[2.1 High-Level Architecture 3](#_Toc145350405)

[2.1.1 Presentation Layer (Frontend) 3](#_Toc145350406)

[2.1.2 Application Layer (Backend) 3](#_Toc145350407)

[2.1.3 Data Layer (Database) 3](#_Toc145350408)

[2.2 System Components 3](#_Toc145350409)

[2.2.1 Frontend Components 3](#_Toc145350410)

[2.2.2 Backend Components 4](#_Toc145350411)

[2.2.3 Database Components 4](#_Toc145350412)

[2.3 Integration Points 4](#_Toc145350413)

[2.3.1 Internal Integrations 4](#_Toc145350414)

[2.3.2 External Integrations (if applicable) 4](#_Toc145350415)

[3. Technology Stack 4](#_Toc145350416)

[3.1 Frontend Technologies 4](#_Toc145350417)

[3.2 Backend Technologies 5](#_Toc145350418)

[3.3 Database Management System 5](#_Toc145350419)

[4. Database Design 5](#_Toc145350420)

[4.1 Entity-Relationship Diagram (ERD) 5](#_Toc145350421)

[4.2 Database Tables and Schema 5](#_Toc145350422)

[5. Module Breakdown 5](#_Toc145350423)

[5.1 Authentication Module 5](#_Toc145350424)

[5.2 Time Entry Module 6](#_Toc145350425)

[5.3 Approval Workflow Module 6](#_Toc145350426)

[5.4 Reporting Module 7](#_Toc145350427)

[6. Scalability and Performance 7](#_Toc145350428)

[6.1 Load Balancing 7](#_Toc145350429)

[6.2 Caching Strategies 7](#_Toc145350430)

[6.3 Database Optimization 8](#_Toc145350431)

[7. Security Measures 8](#_Toc145350432)

[7.1 Data Encryption 8](#_Toc145350433)

[7.2 Authentication and Authorization 8](#_Toc145350434)

[7.3 Role-Based Access Control 8](#_Toc145350435)

[8. Deployment Plan 9](#_Toc145350436)

[8.1 Staging Environment 9](#_Toc145350437)

[8.2 Production Environment 9](#_Toc145350438)

[8.3 Deployment Workflow 9](#_Toc145350439)

[9. Architectural Diagrams 9](#_Toc145350440)

[9.1 System Architecture Diagram 9](#_Toc145350441)

[9.2 Workflow Diagram 10](#_Toc145350442)

[9.3 Data Flow Diagram 10](#_Toc145350443)

# 1. Introduction

## 1.1 Purpose

The purpose of this Architectural Document is to provide a comprehensive overview of the Time Sheet System's architecture, including system components, technologies used, and integration points.

## 1.2 Scope

This document covers the architectural aspects of the Time Sheet System, detailing its high-level structure, technology stack, database design, module breakdown, scalability, security measures, deployment plan, and architectural diagrams.

## 1.3 Document Overview

This document is intended to serve as a reference for architects, developers, and stakeholders involved in the Time Sheet System project. It provides insights into the system's design and lays the foundation for successful development and deployment.

# 2. System Architecture Overview

## 2.1 High-Level Architecture

The Time Sheet System follows a three-tier architecture, comprising:

### 2.1.1 Presentation Layer (Frontend)

The frontend is built using modern web technologies, including HTML, CSS, and TypeScript.

Angular is used as the core frontend framework, providing a responsive and interactive user interface.

UI libraries such as Material-UI are employed for consistent and visually appealing design.

### 2.1.2 Application Layer (Backend)

The backend is developed using .net core 7 for building RESTful APIs.

Authentication, time entry, approval workflow, and reporting are the key components of the application layer.

The backend is responsible for handling client requests, processing business logic, and interacting with the database.

### 2.1.3 Data Layer (Database)

The database management system used is PostgreSQL, a robust and scalable relational database.

The schema design includes tables for employees, projects, time entries, and user authentication data.

## 2.2 System Components

### 2.2.1 Frontend Components

User Interfaces: The frontend includes intuitive user interfaces for employee time entry, manager approvals, and reporting.

Client-Side Logic: TypeScript and angular handle client-side interactivity, ensuring a seamless user experience.

### 2.2.2 Backend Components

Server: The backend server is built using C#, providing RESTful API endpoints for communication with the frontend.

Application Logic: Business logic for time tracking, approvals, and reporting is implemented in the backend.

Authentication Logic: The authentication module ensures secure user login and session management.

Workflow Logic: The system includes workflows for submitting and approving time entries.

Reporting Logic: Reporting functionality is implemented to generate various reports based on user requirements.

### 2.2.3 Database Components

Database Server: The database server hosts the PostgreSQL database management system.

Database Management System: PostgreSQL manages data storage and retrieval efficiently.

Data Tables: The database schema includes tables for employees, projects, time entries, and user authentication data.

# 2.3 Integration Points

## 2.3.1 Internal Integrations

Communication between Frontend and Backend: The frontend communicates with the backend via RESTful APIs for user authentication, time entry submission, approvals, and reporting.

Data exchange between Backend and Database: The backend interacts with the database to store and retrieve employee, project, and time tracking data.

## 2.3.2 External Integrations (if applicable)

Integration with Third-Party Services: Integration with external services for single sign-on (SSO) or data exchange may be implemented based on specific requirements.

# 3. Technology Stack

## 3.1 Frontend Technologies

HTML, CSS, TypeScript: The frontend uses these core web technologies for building web pages and handling user interactions.

Angulatr Framework: Angular is chosen as the core frontend framework due to its component-based architecture, and strong community support.

UI Libraries: UI libraries like Material-UI are employed for consistent and visually appealing UI components.

## 3.2 Backend Technologies

.Net Core 7: .Net core is chosen as the backend runtime environment due to its non-blocking, event-driven architecture, which is well-suited for handling concurrent requests.

Web Api: web api is used to build the backend server, create RESTful APIs.

RESTful APIs: RESTful API endpoints are designed for communication between the frontend and backend, ensuring a standardized and scalable interface.

## 3.3 Database Management System

PostgreSQL: PostgreSQL is selected as the database management system for its reliability, ACID compliance, and support for complex queries. It provides a solid foundation for storing and retrieving time tracking data.

# 4. Database Design

## 4.1 Entity-Relationship Diagram (ERD)

[Insert Entity-Relationship Diagram (ERD) depicting the relationships between tables such as employees, projects, time entries, and user authentication.]

## 4.2 Database Tables and Schema

The database schema includes the following tables:

Employees Table: Stores employee information, including names, roles, and user authentication details.

Projects Table: Contains project details, such as project names, descriptions, and associated employees.

Time Entries Table: Records time entries, including timestamps, project references, task descriptions, and hours worked.

User Authentication Table: Manages user credentials, including usernames, hashed passwords, and authentication tokens.

# 5. Module Breakdown

## 5.1 Authentication Module

Description

The Authentication Module is responsible for user authentication, authorization, and session management.

Key Components

User Login and Registration

Token-based Authentication

Password Hashing

Role-Based Access Control (RBAC)

Secure Session Management

Authentication Workflow

User Registration: New employees can register by providing their details.

User Login: Registered employees can log in using their credentials.

Token Generation: Upon successful login, a secure authentication token is generated.

Role-Based Access: RBAC ensures that users have appropriate permissions based on their roles.

Session Management: Secure session management is implemented to maintain user sessions.

## 5.2 Time Entry Module

Description

The Time Entry Module allows employees to record their work hours and project/task details.

Key Components

Time Entry Form

Project and Task Selection

Validation and Error Handling

Data Persistence

Time Entry Workflow

User Authentication: Ensure that the user is authenticated.

Time Entry Form: Provide a user-friendly form for time entry.

Project and Task Selection: Allow users to select the project and task for time tracking.

Validation and Error Handling: Validate data to prevent errors and ensure accuracy.

Data Persistence: Store time entry data securely in the database.

## 5.3 Approval Workflow Module

Description

The Approval Workflow Module facilitates the review and approval of time entries by managers.

Key Components

Manager Notifications

Time Entry Review

Approval and Rejection

Audit Trail

Approval Workflow Diagram

[Insert Approval Workflow Diagram depicting the flow of time entry submissions, manager notifications, and approval/rejection processes.]

## 5.4 Reporting Module

Description

The Reporting Module provides users with the ability to generate and customize reports on employee work hours and project costs.

Key Components

Report Generation

Customization Options

Export Formats (e.g., PDF, Excel)

Data Visualization

Reporting Workflow

User Authentication: Ensure that the user is authenticated.

Report Selection: Allow users to choose the type of report to generate.

Customization: Provide options for customizing report parameters.

Data Retrieval: Retrieve relevant data from the database.

Report Generation: Generate reports in the selected format.

Export: Allow users to export reports in various formats.

# 6. Scalability and Performance

## 6.1 Load Balancing

Load Balancer Configuration

To ensure high availability and even distribution of traffic, a load balancer is configured to distribute incoming requests across multiple backend servers.

Handling High Traffic

The system is designed to handle a significant number of concurrent users.

Autoscaling mechanisms are in place to add or remove server instances based on traffic.

## 6.2 Caching Strategies

Caching Mechanisms

Caching is employed to reduce latency and improve response times.

Commonly accessed data, such as user profiles, is cached to minimize database queries.

Cache Invalidation

Cache invalidation mechanisms are implemented to ensure that cached data remains up-to-date.

Cache expiration policies are set to refresh cached data as needed.

## 6.3 Database Optimization

Indexing Strategies

Proper indexing of database tables is performed to optimize query performance.

Indexes are created on frequently queried columns to accelerate data retrieval.

Query Optimization

Complex queries are optimized to reduce database load and response times.

Query execution plans are analyzed and improved as needed.

# 7. Security Measures

## 7.1 Data Encryption

HTTPS for Secure Data Transmission

All data transmission between clients and the server is secured using HTTPS.

SSL/TLS certificates are implemented to encrypt data in transit.

Data Encryption at Rest

Data stored in the database is encrypted to prevent unauthorized access.

Encryption keys are managed securely to protect sensitive information.

## 7.2 Authentication and Authorization

Authentication Flow Diagram

[Insert Authentication Flow Diagram depicting the user login and token-based authentication process.]

Role-Based Access Control (RBAC)

RBAC is employed to assign roles to users and define their access privileges.

Roles include Employee, Manager, and Admin, each with specific permissions.

## 7.3 Role-Based Access Control

Role Definitions

Employee: Can enter time entries and view personal data.

Manager: Can review and approve time entries of assigned employees.

Admin: Has administrative privileges, including user management.

Access Control Policies

Access control policies are enforced to restrict unauthorized access to sensitive data and system functionalities.

Least privilege principles are followed to minimize security risks.

# 8. Deployment Plan

## 8.1 Staging Environment

Staging Server Configuration

A dedicated staging server is set up for testing and quality assurance.

Staging environment mirrors the production environment's configuration.

Testing Procedures

Rigorous testing procedures, including unit tests, feature tests, and integration tests, are performed in the staging environment.

Testing covers functionality, security, performance, and scalability.

## 8.2 Production Environment

Production Server Configuration

Production servers are configured with robust security measures and scalability options.

Server and database configurations are optimized for production workloads.

## 8.3 Deployment Workflow

Version Control and Continuous Integration

Version control systems (e.g., Git) and CI/CD pipelines are used for code management and automated deployments.

Deployment pipelines ensure consistency and reliability in the release process.

Deployment Steps

Deployment to production follows a well-defined workflow, including staging environment testing, database migrations, and data seeding.

Rollback procedures are established to handle deployment failures.

# 9. Architectural Diagrams

## 9.1 System Architecture Diagram

[Insert System Architecture Diagram depicting the high-level architecture, including the presentation, application, and data layers.]

## 9.2 Workflow Diagram

[Insert Workflow Diagram illustrating the flow of key processes, such as time entry submission, manager approvals, and reporting.]

## 9.3 Data Flow Diagram

[Insert Data Flow Diagram illustrating how data flows within the system, from user input to database storage and retrieval.]