Table of Contents

# HR Website System - Business Documentation

**Document Version**: 1.0  
**Date**: October 1, 2025  
**Prepared by**: Development Team  
**Project**: HR Website Enterprise Application

## Table of Contents

1. [Executive Summary](#executive-summary)
2. [System Overview](#system-overview)
3. [Business Requirements](#business-requirements)
4. [System Architecture](#system-architecture)
5. [User Roles and Permissions](#user-roles-and-permissions)
6. [Functional Specifications](#functional-specifications)
7. [Business Workflows](#business-workflows)
8. [Security Framework](#security-framework)
9. [Data Management](#data-management)
10. [System Administration](#system-administration)
11. [Technical Specifications](#technical-specifications)
12. [Implementation Timeline](#implementation-timeline)
13. [Testing Strategy](#testing-strategy)
14. [Deployment Guide](#deployment-guide)
15. [User Training Materials](#user-training-materials)
16. [Maintenance and Support](#maintenance-and-support)
17. [Appendices](#appendices)

## 1. Executive Summary

### 1.1 Project Overview

The HR Website is a comprehensive Human Resources management system designed to streamline timesheet management, employee oversight, and approval workflows within the organization. Built on enterprise-grade Spring Boot architecture, the system provides secure, role-based access to HR functions while maintaining data integrity and audit capabilities.

### 1.2 Business Objectives

* **Efficiency**: Reduce manual timesheet processing time by 80%
* **Accuracy**: Eliminate data entry errors through validation
* **Compliance**: Ensure audit trail for all timesheet operations
* **Security**: Implement role-based access control
* **Scalability**: Support growing organization requirements

### 1.3 Key Benefits

* Automated timesheet approval workflows
* Real-time visibility into employee work hours
* Comprehensive reporting capabilities
* Secure JWT-based authentication
* Enterprise-level architecture for future expansion

## 2. System Overview

### 2.1 System Purpose

The HR Website serves as a centralized platform for: - Employee timesheet submission and management - Manager approval and oversight functions - Administrative user management - Comprehensive reporting and analytics

### 2.2 Core Functionality

* **Employee Portal**: Submit, edit, and track timesheets
* **Manager Dashboard**: Review, approve, and reject timesheets
* **Authentication System**: Secure login with JWT tokens
* **Audit Trail**: Complete history of all system operations

### 2.3 Technology Foundation

* **Backend**: Spring Boot 3.2.0 with Java 17
* **Database**: PostgreSQL 17.6 for data persistence
* **Security**: JWT authentication with Spring Security
* **API**: RESTful web services with JSON responses
* **Architecture**: Layered enterprise architecture

## 3. Business Requirements

### 3.1 Functional Requirements

#### 3.1.1 User Management

* **FR-001**: System shall support user registration with role assignment
* **FR-002**: System shall authenticate users with email and password
* **FR-003**: System shall generate JWT tokens for session management
* **FR-004**: System shall support EMPLOYEE and MANAGER roles

#### 3.1.2 Timesheet Management

* **FR-005**: Employees shall submit daily timesheets with hours and project details
* **FR-006**: System shall validate timesheet data before submission
* **FR-007**: Employees shall edit pending timesheets
* **FR-008**: System shall prevent future date timesheet submissions

#### 3.1.3 Approval Workflow

* **FR-009**: Managers shall view all employee timesheets
* **FR-010**: Managers shall approve or reject timesheets
* **FR-011**: System shall track timesheet status changes
* **FR-012**: System shall prevent unauthorized access to manager functions

### 3.2 Non-Functional Requirements

#### 3.2.1 Performance

* **NFR-001**: System shall respond to API calls within 2 seconds
* **NFR-002**: System shall support concurrent users (up to 100)
* **NFR-003**: Database queries shall be optimized for performance

#### 3.2.2 Security

* **NFR-004**: All API endpoints shall require authentication
* **NFR-005**: System shall use HTTPS for all communications
* **NFR-006**: JWT tokens shall expire after 24 hours
* **NFR-007**: System shall validate all input data

#### 3.2.3 Reliability

* **NFR-008**: System shall maintain 99.5% uptime
* **NFR-009**: Database shall be backed up daily
* **NFR-010**: System shall log all operations for audit purposes

## 4. System Architecture

### 4.1 Architecture Overview

The HR Website follows a layered enterprise architecture pattern with clear separation of concerns:

┌─────────────────────────────────────┐  
│ Frontend/API Layer │  
│ (REST Controllers) │  
├─────────────────────────────────────┤  
│ Business Logic Layer │  
│ (Services) │  
├─────────────────────────────────────┤  
│ Data Access Layer │  
│ (Repositories) │  
├─────────────────────────────────────┤  
│ Data Persistence Layer │  
│ (PostgreSQL Database) │  
└─────────────────────────────────────┘

### 4.2 Component Description

#### 4.2.1 API Layer (Controllers)

* **AuthController**: Handles user registration and authentication
* **TimesheetController**: Manages employee timesheet operations
* **ManagerController**: Provides manager-specific functionality

#### 4.2.2 Business Logic Layer (Services)

* **AuthService**: Authentication and authorization logic
* **TimesheetService**: Timesheet business rules and validation
* **UserService**: User management operations

#### 4.2.3 Data Access Layer (Repositories)

* **UserRepository**: User data operations
* **TimesheetRepository**: Timesheet data operations

#### 4.2.4 Cross-Cutting Concerns

* **Security**: JWT authentication and authorization
* **Validation**: Input validation and business rule enforcement
* **Logging**: Comprehensive operation logging
* **Exception Handling**: Centralized error management

## 5. User Roles and Permissions

### 5.1 Employee Role

**Permissions:** - Submit personal timesheets - View personal timesheet history - Edit pending timesheets - Delete own pending timesheets

**Restrictions:** - Cannot access other employee data - Cannot approve/reject timesheets - Cannot access manager functions

### 5.2 Manager Role

**Permissions:** - All employee permissions - View all employee timesheets - Approve/reject any timesheet - View user management information - Access system-wide reports

**Responsibilities:** - Review timesheet accuracy - Approve legitimate work hours - Reject invalid or incomplete timesheets - Monitor team productivity

## 6. Functional Specifications

### 6.1 User Registration and Authentication

#### 6.1.1 Registration Process

1. User provides name, email, password, and role
2. System validates input data
3. System encrypts password
4. System creates user account
5. System returns confirmation

#### 6.1.2 Login Process

1. User provides email and password
2. System validates credentials
3. System generates JWT token
4. System returns token and user information

### 6.2 Timesheet Management

#### 6.2.1 Timesheet Submission

1. Employee selects date
2. Employee enters hours worked (1-24)
3. Employee specifies project
4. Employee adds optional notes
5. System validates all fields
6. System saves timesheet with PENDING status

#### 6.2.2 Timesheet Editing

1. Employee selects pending timesheet
2. Employee modifies allowed fields
3. System validates changes
4. System updates timesheet
5. Timesheet remains in PENDING status

### 6.3 Manager Approval Workflow

#### 6.3.1 Approval Process

1. Manager views pending timesheets
2. Manager reviews timesheet details
3. Manager decides to approve or reject
4. System updates timesheet status
5. System logs approval action

## 7. Business Workflows

### 7.1 Daily Timesheet Workflow

[Employee Submits Timesheet]   
 ↓  
[System Validates Data]  
 ↓  
[Timesheet Status: PENDING]  
 ↓  
[Manager Reviews Timesheet]  
 ↓  
 [Approve/Reject?]  
 ↓ ↓  
[APPROVED] [REJECTED]  
 ↓ ↓  
[End] [Employee Resubmits]

### 7.2 Weekly Approval Workflow

[Weekly Timesheet Collection]  
 ↓  
[Manager Reviews All Pending]  
 ↓  
[Bulk Approval Process]  
 ↓  
[Status Updates Applied]  
 ↓  
[Reporting Generated]

## 8. Security Framework

### 8.1 Authentication Strategy

* **Method**: JWT (JSON Web Tokens)
* **Expiration**: 24 hours
* **Algorithm**: HS256
* **Storage**: Client-side (secure storage recommended)

### 8.2 Authorization Model

* **Role-Based Access Control (RBAC)**
* **Endpoint Protection**: All API endpoints require valid JWT
* **Method-Level Security**: Spring Security annotations
* **Data Filtering**: Users see only authorized data

### 8.3 Security Best Practices

* Password encryption using BCrypt
* Input validation and sanitization
* SQL injection prevention
* Cross-Site Request Forgery (CSRF) protection
* Secure HTTP headers

## 9. Data Management

### 9.1 Database Schema

#### 9.1.1 Users Table

CREATE TABLE users (  
 id BIGSERIAL PRIMARY KEY,  
 name VARCHAR(255) NOT NULL,  
 email VARCHAR(255) UNIQUE NOT NULL,  
 password VARCHAR(255) NOT NULL,  
 role VARCHAR(50) NOT NULL,  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

#### 9.1.2 Timesheets Table

CREATE TABLE timesheets (  
 id BIGSERIAL PRIMARY KEY,  
 user\_id BIGINT REFERENCES users(id),  
 date DATE NOT NULL,  
 hours DECIMAL(3,1) NOT NULL,  
 project VARCHAR(255) NOT NULL,  
 notes TEXT,  
 status VARCHAR(50) DEFAULT 'PENDING',  
 created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  
 updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  
);

### 9.2 Data Validation Rules

#### 9.2.1 User Data

* **Name**: 2-255 characters, required
* **Email**: Valid email format, unique
* **Password**: Minimum 6 characters
* **Role**: EMPLOYEE or MANAGER only

#### 9.2.2 Timesheet Data

* **Date**: Valid date, not in future
* **Hours**: 1-24 decimal hours
* **Project**: 2-255 characters, alphanumeric with allowed symbols
* **Status**: PENDING, APPROVED, or REJECTED

### 9.3 Backup Strategy

* **Frequency**: Daily automated backups
* **Retention**: 30 days
* **Location**: Secure backup directory
* **Recovery**: Tested restore procedures

## 10. System Administration

### 10.1 Application Management

#### 10.1.1 Starting the Application

java -jar target/hr-website-0.0.1-SNAPSHOT.jar

#### 10.1.2 Health Monitoring

curl http://localhost:8081/actuator/health

#### 10.1.3 Log Management

* **Location**: logs/ directory
* **Files**: hr-website.log, hr-website-error.log
* **Rotation**: Daily rotation with 30-day retention

### 10.2 Database Administration

#### 10.2.1 Backup Operations

# Automated backup  
.\backup-database.bat  
  
# Manual backup  
pg\_dump -U hruser -h localhost hrdb > backup\_filename.sql

#### 10.2.2 Restore Operations

# Automated restore  
.\restore-database.bat backup\_filename.sql  
  
# Manual restore  
psql -U hruser -h localhost hrdb < backup\_filename.sql

## 11. Technical Specifications

### 11.1 System Requirements

#### 11.1.1 Server Requirements

* **Operating System**: Windows/Linux/macOS
* **Java Runtime**: JRE 17 or higher
* **Memory**: Minimum 512MB RAM, 1GB recommended
* **Storage**: 100MB application, 1GB+ for logs and backups
* **Network**: HTTP/HTTPS ports (8081 default)

#### 11.1.2 Database Requirements

* **Database**: PostgreSQL 17.6+
* **Memory**: 1GB RAM minimum
* **Storage**: 10GB minimum for production
* **Connections**: Support for 20+ concurrent connections

### 11.2 Performance Specifications

* **Response Time**: <2 seconds for API calls
* **Throughput**: 100 concurrent users
* **Database**: Optimized queries with indexing
* **Caching**: In-memory caching for frequent operations

### 11.3 Integration Capabilities

* **API Format**: RESTful JSON APIs
* **Authentication**: JWT token-based
* **Database**: JDBC connectivity
* **Monitoring**: Spring Boot Actuator endpoints

## 12. Implementation Timeline

### 12.1 Phase 1: Foundation (Completed)

* ✅ Project structure setup
* ✅ Database design and implementation
* ✅ Basic authentication system
* ✅ Core entity models

### 12.2 Phase 2: Core Features (Completed)

* ✅ User registration and login
* ✅ Timesheet CRUD operations
* ✅ JWT security implementation
* ✅ Basic validation

### 12.3 Phase 3: Enterprise Features (Completed)

* ✅ Service layer architecture
* ✅ DTO implementation
* ✅ Advanced validation
* ✅ Exception handling
* ✅ Logging and monitoring

### 12.4 Phase 4: Manager Features (Completed)

* ✅ Manager dashboard endpoints
* ✅ Approval/rejection workflow
* ✅ User management features
* ✅ Status-based filtering

### 12.5 Phase 5: Production Readiness (Current)

* 🔄 Comprehensive testing
* 🔄 Documentation completion
* 📋 Deployment procedures
* 📋 Training materials

## 13. Testing Strategy

### 13.1 Testing Levels

#### 13.1.1 Unit Testing

* **Scope**: Individual methods and classes
* **Tools**: JUnit 5, Mockito
* **Coverage**: 80%+ code coverage target
* **Focus**: Business logic validation

#### 13.1.2 Integration Testing

* **Scope**: API endpoints and database interactions
* **Tools**: Spring Boot Test, TestContainers
* **Coverage**: All REST endpoints
* **Focus**: End-to-end workflows

#### 13.1.3 Manual Testing

* **Scope**: User interface and workflows
* **Tools**: Postman, curl commands
* **Coverage**: All user stories
* **Focus**: User experience validation

### 13.2 Test Data Management

* **Test Users**: Predefined employee and manager accounts
* **Test Scenarios**: Common and edge case workflows
* **Data Cleanup**: Automated test data cleanup procedures

## 14. Deployment Guide

### 14.1 Pre-Deployment Checklist

* Java 17 runtime installed
* PostgreSQL 17.6+ installed and configured
* Database user and permissions configured
* Application properties updated
* Firewall rules configured for port 8081
* SSL certificates installed (if using HTTPS)

### 14.2 Deployment Steps

#### 14.2.1 Database Setup

1. Install PostgreSQL 17.6
2. Create database and user
3. Run database initialization script
4. Verify connection

#### 14.2.2 Application Deployment

1. Build application JAR file
2. Transfer to target server
3. Configure application.properties
4. Start application service
5. Verify health endpoints

#### 14.2.3 Post-Deployment Verification

1. Test authentication endpoints
2. Verify database connectivity
3. Check log file creation
4. Test basic workflows
5. Monitor system metrics

### 14.3 Environment Configurations

#### 14.3.1 Development Environment

* Local PostgreSQL instance
* Debug logging enabled
* Development security settings
* Sample data loading

#### 14.3.2 Production Environment

* Dedicated database server
* Info-level logging
* Production security settings
* Backup procedures activated

## 15. User Training Materials

### 15.1 Employee Training

#### 15.1.1 Getting Started

1. **Account Setup**: Registration process and initial login
2. **Dashboard Overview**: Navigation and main features
3. **Submitting Timesheets**: Step-by-step timesheet creation
4. **Editing Timesheets**: Modifying pending submissions
5. **Viewing History**: Accessing historical timesheet data

#### 15.1.2 Best Practices

* Daily timesheet submission
* Accurate hour reporting
* Detailed project descriptions
* Timely submission for approval

### 15.2 Manager Training

#### 15.2.1 Manager Dashboard

1. **Accessing Manager Features**: Role-based navigation
2. **Reviewing Timesheets**: Bulk and individual review
3. **Approval Process**: Best practices for timely approval
4. **Rejection Handling**: Providing feedback for improvements

#### 15.2.2 Reporting and Analytics

* Team productivity metrics
* Approval status tracking
* Historical reporting
* Export capabilities

### 15.3 Administrator Training

#### 15.3.1 System Administration

1. **Application Management**: Starting, stopping, monitoring
2. **Database Maintenance**: Backup and restore procedures
3. **Log Analysis**: Troubleshooting using log files
4. **Performance Monitoring**: Using Actuator endpoints

#### 15.3.2 Troubleshooting Guide

* Common error scenarios
* Database connection issues
* Authentication problems
* Performance optimization

## 16. Maintenance and Support

### 16.1 Preventive Maintenance

#### 16.1.1 Daily Tasks

* Monitor application logs
* Check system health endpoints
* Verify database connectivity
* Review backup completion

#### 16.1.2 Weekly Tasks

* Analyze performance metrics
* Review security logs
* Update system documentation
* Test backup restore procedures

#### 16.1.3 Monthly Tasks

* System performance review
* Capacity planning assessment
* Security audit
* Documentation updates

### 16.2 Support Procedures

#### 16.2.1 Issue Classification

* **Critical**: System unavailable, data loss
* **High**: Major functionality impaired
* **Medium**: Minor functionality issues
* **Low**: Enhancement requests

#### 16.2.2 Escalation Process

1. First Level: Application logs and health checks
2. Second Level: Database and infrastructure analysis
3. Third Level: Development team involvement
4. Fourth Level: Vendor support if required

### 16.3 Change Management

* Version control for all changes
* Testing procedures for updates
* Rollback procedures
* Communication protocols

## 17. Appendices

### Appendix A: API Reference

[Complete list of all API endpoints with request/response examples]

### Appendix B: Database Schema

[Complete database schema with relationships and constraints]

### Appendix C: Configuration Reference

[All configuration options and their descriptions]

### Appendix D: Error Codes

[Complete list of error codes and their meanings]

### Appendix E: Glossary

[Definitions of technical terms and acronyms]

### Appendix F: Contact Information

[Support contacts and escalation procedures]

**Document Control** - **Version**: 1.0 - **Last Updated**: October 1, 2025 - **Next Review**: November 1, 2025 - **Document Owner**: Development Team - **Approved By**: Project Manager

*This document contains proprietary and confidential information. Distribution is restricted to authorized personnel only.*