RosterBhai

System Documentation

Version 2.0.0
November 08, 2025

Table of Contents

1. Executive Summary

RosterBhai is a modern, multi-tenant SaaS application designed for roster and shift management. Built using Next.js 14 with TypeScript, it provides a comprehensive solution for organizations managing employee shifts across different teams.

Key Features:

• Multi-tenant Architecture with subdomain-based tenant isolation

• Employee Portal for self-service shift viewing and swap requests

• Admin Panel with comprehensive roster management and RBAC

• Developer Panel for super-admin controls and tenant management

• Google Sheets Integration for CSV import/export functionality

• Real-time Notifications via email and in-app system

2. System Architecture Overview

RosterBhai follows a 3-tier architecture with clear separation of concerns:

Layer	Technology	Purpose
Presentation	Next.js App Router + React	User Interface
Business Logic	Next.js API Routes	Server-side Processing
Data	JSON File Storage	Tenant-isolated Data

3. Multi-Tenant Architecture

RosterBhai implements subdomain-based multi-tenancy for complete tenant isolation:

```
Main Domain: rosterbhai.me
Tenant Subdomains: [tenant-slug].rosterbhai.me
```

Each tenant has completely isolated data storage in separate directories, ensuring data privacy and security. The middleware enforces routing rules to prevent cross-tenant data access.

4. System Components

Landing Page

Public-facing marketing website with company registration

Developer Portal

Super-admin panel for SaaS management and tenant approval

Admin Panel

Tenant-level roster management with RBAC support

Employee Portal

Self-service portal for schedule viewing and requests

5. Data Models

The system uses a file-based JSON storage with the following core entities:

• Tenant - Organization using the service

• Employee - Individual workers with schedules

• Admin - Tenant-level administrators

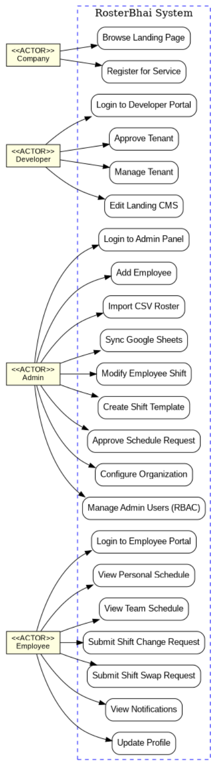
• Developer - System super-administrators

• Schedule Request - Shift change/swap requests

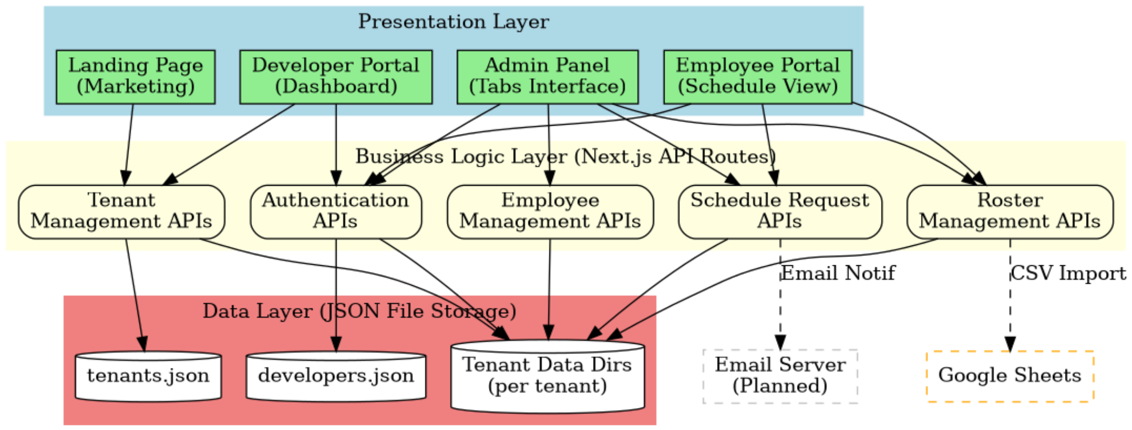
• Roster Data - Shift schedules per team

9. System Diagrams

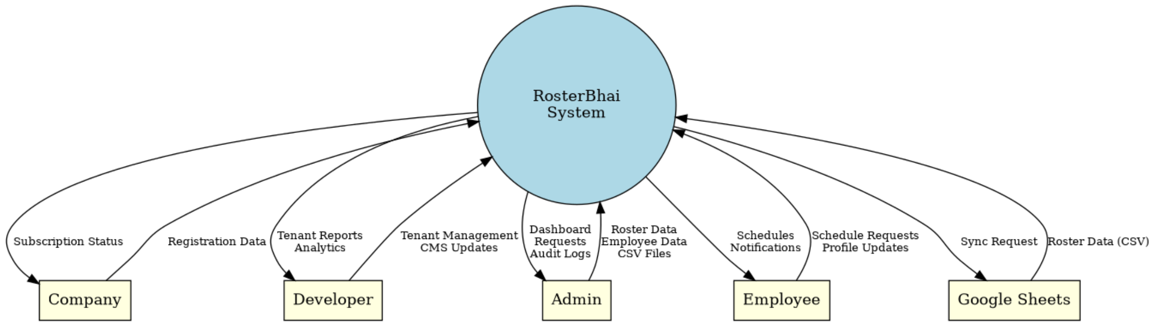
Use Case Diagram



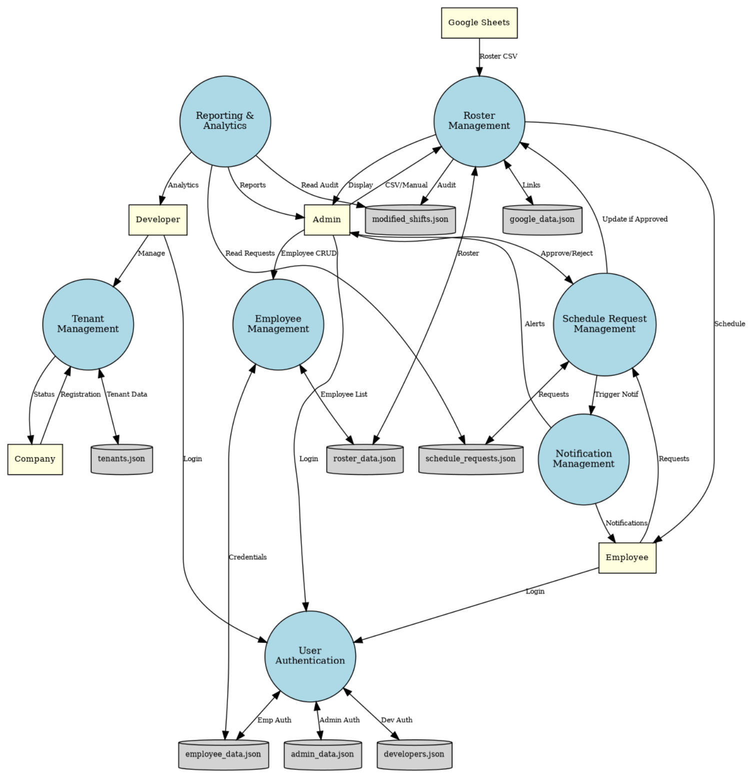
System Architecture



Context DFD

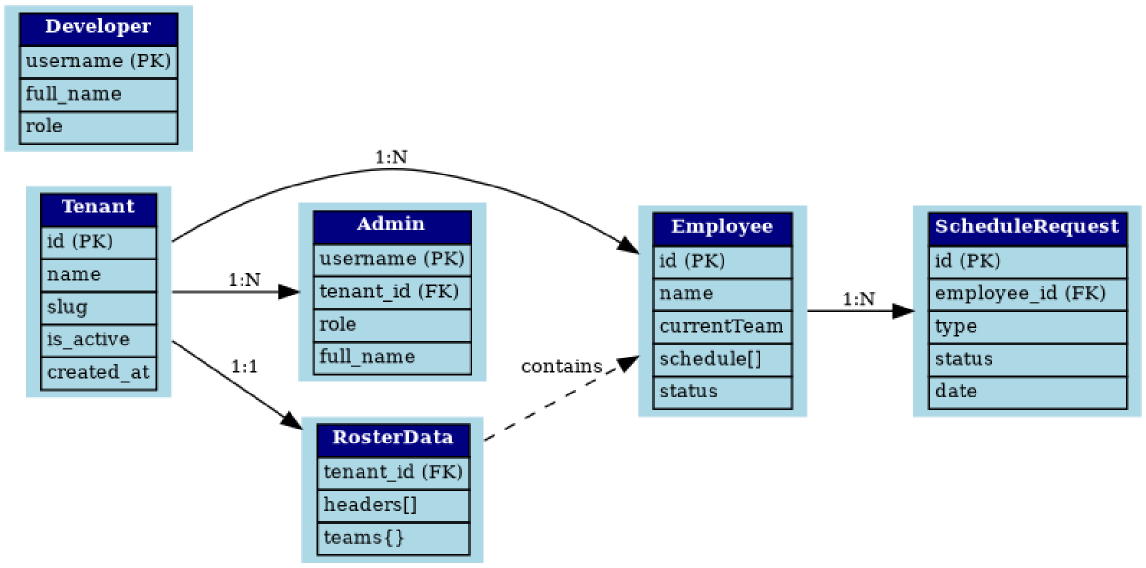

RosterBhai System
Subscription Status
Registration Data
Tenant Reports Analytics
Tenant Management CMS Updates
Dashboard Requests Audit Logs
Roster Data Employee Data CSV Files
Schedules Notifications
Schedule Requests Profile Updates
Sync Request
Roster Data (CSV)
Company
Developer
Admin
Employee
Google Sheets

Level 1 DFD



Google Sheets
Roster CSV
Reporting & Analytics
Roster Management
Analytics
Reports
Read Audit
Display
CSV/Manual
Audit
Links
Developer
Admin
modified_shifts.json
google_data.json
Update if Approved
Manage
Read Requests
Employee CRUD
Roster
Approve/Reject
Tenant Management
Employee Management
Alerts
Schedule Request Management
Schedule
Status
Registration
Tenant Data
Employee List
Requests
Trigger Notif
Company
tenants.json
Login
Login
roster_data.json
schedule_requests.json
Notification Management
Requests
Notifications
Credentials
Employee
Login
User Authentication
Emp Auth
Admin Auth
Dev Auth
employee_data.json
admin_data.json
developers.json

Entity Relationship Diagram


Developer
username (PK)
full_name
role
Tenant
id (PK)
name
slug
is_active
created_at
Admin
username (PK)
tenant_id (FK)
role
full_name
RosterData
tenant_id (FK)
headers[]
teams{}
Employee
id (PK)
name
currentTeam
schedule[]
status
ScheduleRequest
id (PK)
employee_id (FK)
type
status
date
1:N
1:N
1:1
contains
1:N