Phase 3 Implementation Plan: Multi-tenancy & White-label

Date: November 15, 2025
Branch: feature/phase-3-multi-tenancy-whitelabel
Total Tasks: 45 tasks (Tasks 81-105 + Tasks 261-280)

Overview

Phase 3 focuses on enhancing multi-tenancy features and adding comprehensive white-label capabilities to the payroll SaaS platform. This phase builds upon the existing tenant infrastructure and adds advanced customization, resource management, and white-labeling features.

Part A: Multi-tenancy Enhancements (25 tasks)

Category 1: Tenant Settings & Preferences (Tasks 81-99)

✅ Already Implemented

- Task 81: ✅ Tenant isolation at database level (via Prisma with `tenantId`)
- Task 82: ✅ Tenant-specific branding customization (basic colors & logo)
- Task 83: ✅ Logo upload and management (URL-based)
- Task 84: ✅ Color scheme customization (primary, accent, background, sidebar, header)

🔧 To Implement

Task 85: Custom Font Selection
- Add `customFont` field to Tenant model
- Support Google Fonts integration
- Create font selector component
- Apply font dynamically across platform

Task 86: Custom Email Domain Configuration
- Add `customEmailDomain` field to Tenant model
- Email domain verification system
- Configure email sending from custom domain
- DNS record verification UI

Task 87: Tenant Onboarding Workflow
- Create multi-step onboarding wizard for new tenants
- Steps: Basic info → Branding → Team setup → Integration
- Progress tracking
- Skip/resume capability

Task 88: Tenant Subscription Management
- Add `subscriptionPlan`, `subscriptionStatus`, `subscriptionStartDate`, `subscriptionEndDate` to Tenant
- Create subscription plans (Free, Starter, Professional, Enterprise)

- Plan features and limits mapping
- Upgrade/downgrade workflows

Task 89: Tenant Billing and Invoicing
- Add billing information fields to Tenant
- Invoice generation for subscription fees
- Payment history tracking
- Billing cycle management

Task 90: Usage Tracking and Limits
- Track: users, contracts, invoices, storage, API calls
- Add `usageMetrics` JSON field to Tenant
- Real-time usage monitoring
- Warning system when approaching limits

Task 91: Feature Flags System
- Create `TenantFeatureFlag` model
- Map features to subscription plans
- Feature toggle UI for super admin
- Runtime feature checking

Task 92: Tenant Settings Management
- Comprehensive settings page with tabs
- General, Branding, Localization, Integrations, Security
- Settings versioning and audit trail

Task 93: Data Export Functionality
- Export all tenant data (users, contracts, invoices, etc.)
- Multiple formats: JSON, CSV, Excel
- Schedule automated exports
- Download center for exports

Task 94: GDPR Data Deletion
- Complete tenant data deletion workflow
- Soft delete with grace period
- Hard delete after confirmation
- Data deletion certificate

Task 95: Tenant Switching for Super Admin
- Switch context between tenants without re-login
- Visual indicator of current tenant
- Recent tenants list
- Quick switch dropdown

Task 96: Tenant Impersonation for Support
- Support staff can view tenant as admin
- Audit trail for impersonation sessions
- Limited duration impersonation
- Clear visual indicator when impersonating

Task 97: Timezone Settings
- Add `timezone` field (default: UTC)
- Timezone selector component

- Auto-detect timezone option
- Apply timezone across all dates

Task 98: Language Preferences
- Add `defaultLanguage` field
- Support: English, French, Spanish, German
- i18n integration
- Language switcher component

Task 99: Currency Settings
- Add `defaultCurrency` field
- Multi-currency support
- Currency formatting
- Currency conversion for reports

Category 2: Resource Management & Quotas (Tasks 100-105)

Task 100: Date Format Preferences
- Add `dateFormat` field (e.g., MM/DD/YYYY, DD/MM/YYYY)
- `timeFormat` field (12h/24h)
- Apply format across platform

Task 101: Resource Quota Management
- Create `TenantQuota` model
- Track: maxUsers, maxContracts, maxStorage, maxInvoices, maxAPICallsPerMonth
- Real-time quota checking
- Quota enforcement

Task 102: Storage Limit Enforcement
- Track file uploads size
- Add `currentStorageUsed` to Tenant
- Block uploads when limit reached
- Storage cleanup tools

Task 103: User Limit Enforcement
- Add `maxUsers` to TenantQuota
- Block new user creation at limit
- Warning before limit
- Upgrade prompts

Task 104: Subdomain Management
- Add `subdomain` field (unique)
- Subdomain validation and availability check
- Automatic subdomain assignment
- Custom subdomain for paid plans

Task 105: Custom Domain Support
- Add `customDomain` field
- Domain verification via DNS
- SSL certificate management
- Domain pointing instructions

Task 105B: Tenant-level Security Settings
- Password policy configuration

- Session timeout settings
- IP whitelist/blacklist
- 2FA enforcement

Part B: White-label Features (20 tasks)

Category 3: Email & Communication Customization (Tasks 261-268)

Task 261: Custom Branding Per Tenant
- Already covered in Part A (Tasks 82-84)
- Enhanced with more branding options

Task 262: Custom Logo Upload
- Already implemented (URL-based)
- Enhance with actual file upload
- Multiple logo variants (main, icon, email)

Task 263: Custom Color Scheme Editor
- Already implemented
- Add more color options (success, warning, error, info)

Task 264: Custom Font Selection
- Covered in Task 85

Task 265: Custom Email Templates
- Create `EmailTemplate` model
- Template variables system
- Preview functionality
- Template for: welcome, invoice, contract, password reset, etc.

Task 266: Custom Email Header/Footer
- Add to EmailTemplate
- Social links in footer
- Contact information
- Unsubscribe link

Task 267: Custom Domain Support
- Covered in Task 105

Task 268: Custom SSL Certificate Management
- Let’s Encrypt integration
- Certificate renewal automation
- Certificate status monitoring

Category 4: UI Customization (Tasks 269-273)

Task 269: Custom Login Page Branding
- Add `loginPageConfig` JSON field to Tenant
- Custom background image
- Custom welcome message
- Custom login form styling

Task 270: Custom Terms of Service
- Add `termsOfService` text field to Tenant
- Version tracking
- User acceptance tracking
- Display on registration

Task 271: Custom Privacy Policy
- Add `privacyPolicy` text field to Tenant
- Version tracking
- Cookie policy section
- GDPR compliance

Task 272: Custom Navigation Menu
- Add `navigationConfig` JSON field to Tenant
- Drag-and-drop menu builder
- Show/hide menu items
- Custom menu ordering

Task 273: Custom Dashboard Widgets
- Widget configuration system
- Add/remove/reorder widgets
- Widget-specific permissions
- Custom widget data sources

Category 5: Document & Template Customization (Tasks 274-280)

Task 274: Custom PDF Templates
- Create `PDFTemplate` model
- Template engine (Handlebars)
- Header/footer customization
- Page numbering and watermarks

Task 275: Custom Contract Templates
- Contract template builder
- Variable placeholders
- Conditional sections
- Multi-language templates

Task 276: Custom Invoice Templates
- Invoice template builder
- Tax calculation display
- Payment terms section
- Logo and branding

Task 277: Custom Payslip Templates
- Payslip template builder
- Earnings/deductions breakdown
- Year-to-date summaries
- Digital signature field

Task 278: Custom Notification Templates
- Notification template system
- In-app notification styling

- Push notification format
- SMS templates

Task 279: White-label Mobile App Configuration
- App name, icon, splash screen
- App store metadata
- Push notification configuration
- Deep linking setup

Task 280: Tenant-specific Feature Toggles
- Covered in Task 91 (Feature Flags)

Implementation Strategy

Phase 3.1: Database Schema Updates

1. Add new fields to Tenant model
2. Create new models: TenantQuota, TenantFeatureFlag, EmailTemplate, PDFTemplate
3. Add indexes for performance
4. Run migrations

Phase 3.2: Backend API Development

1. Extend tenant router with new procedures
2. Create quota checking middleware
3. Create feature flag checking utilities
4. Add validation and error handling

Phase 3.3: Frontend Components

1. Create reusable settings components
2. Build multi-step onboarding wizard
3. Create admin dashboard for super admin
4. Build tenant switching UI

Phase 3.4: Permissions & Security

1. Add Phase 3 permissions to PERMISSION_TREE
2. Implement permission guards
3. Add audit logging for sensitive operations

Phase 3.5: Testing & Documentation

1. Test all new features
2. Verify existing functionality
3. Update documentation
4. Create migration guide

Database Schema Changes

Tenant Model Extensions

```
model Tenant {
  // ... existing fields ...

  // PHASE 3: Multi-tenancy Enhancements
  customFont             String?   @default("Inter")
  customEmailDomain      String?
  emailDomainVerified    Boolean   @default(false)

  // Subscription
  subscriptionPlan       String    @default("free") // free, starter, professional, en-
terprise
  subscriptionStatus     String    @default("active") // active, trial, suspended, can-
celled
  subscriptionStartDate DateTime  @default(now())
  subscriptionEndDate    DateTime?

  // Usage & Limits
  currentStorageUsed     BigInt    @default(0) // in bytes
  usageMetrics           Json?     // { apiCalls: 0, emailsSent: 0, etc. }

  // Localization
  timezone               String    @default("UTC")
  defaultLanguage        String    @default("en")
  defaultCurrency        String    @default("USD")
  dateFormat             String    @default("MM/DD/YYYY")
  timeFormat             String    @default("12h")

  // Domain
  subdomain              String?   @unique
  customDomain           String?   @unique
  customDomainVerified   Boolean   @default(false)
  sslCertificateStatus   String?   // pending, active, expired

  // White-label
  loginPageConfig        Json?     // { backgroundImage, welcomeMessage, etc. }
  navigationConfig       Json?     // Custom menu structure
  termsOfService         String?   @db.Text
  termsVersion           String?
  privacyPolicy          String?   @db.Text
  privacyPolicyVersion   String?

  // Relations
  featureFlags           TenantFeatureFlag[]
  quotas                 TenantQuota?
  emailTemplates         EmailTemplate[]
  pdfTemplates           PDFTemplate[]
}

model TenantQuota {
  id                   String   @id @default(cuid())
  tenantId             String   @unique

  maxUsers             Int      @default(10)
  maxContracts         Int      @default(50)
  maxInvoices          Int      @default(100)
  maxStorage           BigInt   @default(1073741824) // 1GB in bytes
  maxAPICallsPerMonth  Int      @default(10000)
  maxEmailsPerMonth    Int      @default(1000)

  createdAt            DateTime @default(now())
  updatedAt            DateTime @updatedAt
  tenant               Tenant   @relation(fields: [tenantId], references: [id], onDele
te: Cascade)

  @@map("tenant_quotas")
}

model TenantFeatureFlag {
  id          String   @id @default(cuid())
  tenantId    String
  featureKey  String   // e.g., "advanced_analytics", "custom_domain", "api_access"
  enabled     Boolean  @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant   @relation(fields: [tenantId], references: [id], onDelete: Cas-
cade)

  @@unique([tenantId, featureKey])
  @@index([tenantId])
  @@map("tenant_feature_flags")
}

model EmailTemplate {
  id          String   @id @default(cuid())
  tenantId    String
  name        String   // welcome_email, invoice_email, etc.
  subject     String
  htmlBody    String   @db.Text
  textBody    String?  @db.Text
  variables   Json?    // Available template variables
  isActive    Boolean  @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant   @relation(fields: [tenantId], references: [id], onDelete: Cas-
cade)

  @@unique([tenantId, name])
  @@index([tenantId])
  @@map("email_templates")
}

model PDFTemplate {
  id          String   @id @default(cuid())
  tenantId    String
  name        String   // contract_template, invoice_template, payslip_template
  type        String   // contract, invoice, payslip, report
  template    String   @db.Text // Handlebars template
  headerHtml  String?  @db.Text
  footerHtml  String?  @db.Text
  styles      Json?    // CSS styles
  isActive    Boolean  @default(true)
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt

  tenant      Tenant   @relation(fields: [tenantId], references: [id], onDelete: Cas-
cade)

  @@unique([tenantId, name])
  @@index([tenantId])
  @@map("pdf_templates")
}
```

```
model TenantSecuritySettings {
  id                        String   @id @default(cuid())
  tenantId                  String   @unique

  // Password Policy
  minPasswordLength         Int      @default(8)
  requireUppercase          Boolean  @default(true)
  requireLowercase          Boolean  @default(true)
  requireNumbers            Boolean  @default(true)
  requireSpecialChars       Boolean  @default(false)
  passwordExpiryDays        Int?     // null = never expires

  // Session
  sessionTimeoutMinutes     Int      @default(60)
  maxConcurrentSessions     Int      @default(3)

  // IP Restrictions
  ipWhitelist               Json?    // ["IP1", "IP2"]
  ipBlacklist               Json?    // ["IP1", "IP2"]

  // 2FA
  enforce2FA                Boolean  @default(false)

  createdAt                 DateTime @default(now())
  updatedAt                 DateTime @updatedAt

  tenant                    Tenant   @relation(fields: [tenantId], references: [id], onD
elete: Cascade)

  @@map("tenant_security_settings")
}
```

New API Procedures

Tenant Router Extensions

```
// Subscription Management
- getSubscriptionInfo()
- updateSubscriptionPlan({ plan, billingCycle })
- cancelSubscription()
- getBillingHistory()

// Usage & Quotas
- getUsageMetrics()
- getQuotaLimits()
- checkQuotaAvailability({ resourceType })

// Feature Flags
- getEnabledFeatures()
- checkFeatureAccess({ featureKey })

// Localization
- updateLocalizationSettings({ timezone, language, currency, dateFormat })

// Domain Management
- checkSubdomainAvailability({ subdomain })
- updateSubdomain({ subdomain })
- addCustomDomain({ domain })
- verifyCustomDomain({ domain })
- removeCustomDomain()

// Data Export
- requestDataExport({ format, entities })
- getExportHistory()
- downloadExport({ exportId })

// GDPR
- requestDataDeletion()
- confirmDataDeletion({ confirmationToken })

// Security Settings
- getSecuritySettings()
- updateSecuritySettings({ ... })

// Templates
- listEmailTemplates()
- createEmailTemplate({ name, subject, htmlBody })
- updateEmailTemplate({ id, ... })
- deleteEmailTemplate({ id })

- listPDFTemplates()
- createPDFTemplate({ name, type, template })
- updatePDFTemplate({ id, ... })
- deletePDFTemplate({ id })

// Super Admin
- getTenantAnalytics({ tenantId })
- impersonateTenant({ tenantId })
- endImpersonation()
- switchTenant({ tenantId })
```

New Permissions

Add to `PERMISSION_TREE`:

```
tenant: {
  // ... existing permissions ...
  subscription: {
    view: "tenant:subscription:view",
    manage: "tenant:subscription:manage",
  },
  quotas: {
    view: "tenant:quotas:view",
    manage: "tenant:quotas:manage",
  },
  features: {
    view: "tenant:features:view",
    manage: "tenant:features:manage",
  },
  localization: {
    view: "tenant:localization:view",
    manage: "tenant:localization:manage",
  },
  domain: {
    view: "tenant:domain:view",
    manage: "tenant:domain:manage",
  },
  templates: {
    view: "tenant:templates:view",
    manage: "tenant:templates:manage",
  },
  security: {
    view: "tenant:security:view",
    manage: "tenant:security:manage",
  },
  data: {
    export: "tenant:data:export",
    delete: "tenant:data:delete",
  },
}
```

Frontend Pages/Components to Create

1. **Settings → Subscription** (`/settings/subscription`)
   - Current plan details
   - Usage metrics with progress bars
   - Upgrade/downgrade options
   - Billing history

2. **Settings → Localization** (`/settings/localization`)
   - Timezone selector
   - Language selector
   - Currency selector
   - Date/time format selector

3. **Settings → Domain** (`/settings/domain`)
   - Subdomain configuration
   - Custom domain setup
   - DNS verification instructions
   - SSL certificate status

4. **Settings → Templates** (`/settings/templates`)
   - Email templates list
   - PDF templates list
   - Template editor with preview
   - Variable reference

5. **Settings → Security** (`/settings/security`)
   - Password policy
   - Session settings
   - IP restrictions
   - 2FA enforcement

6. **Settings → Features** (`/settings/features`)
   - Available features list
   - Enabled/disabled status
   - Feature descriptions
   - Upgrade prompts for premium features

7. **SuperAdmin → Tenant Management** (enhance existing)
   - Tenant analytics dashboard
   - Impersonation controls
   - Quota management
   - Feature flag management

8. **Onboarding Wizard** (`/onboarding`)
   - Step 1: Welcome & Basic Info
   - Step 2: Branding & Logo
   - Step 3: Team Setup
   - Step 4: Integrations
   - Progress indicator

Testing Checklist

Database & Schema

- [ ] All migrations run successfully
- [ ] All new fields have proper defaults
- [ ] Indexes are created correctly
- [ ] Relations work as expected

Backend API

- [ ] All new procedures work correctly
- [ ] Proper error handling
- [ ] Permissions are enforced

- [ ] Audit logs are created

Frontend

- [ ] All new pages render correctly
- [ ] Forms validate properly
- [ ] Success/error messages display
- [ ] Loading states work
- [ ] Responsive design

Integration

- [ ] Existing features still work
- [ ] No breaking changes
- [ ] Proper data isolation between tenants
- [ ] Performance is acceptable

Security

- [ ] Permissions are enforced at all levels
- [ ] No data leakage between tenants
- [ ] Sensitive operations are logged
- [ ] Input validation prevents injection

Success Criteria

✅ All 45 Phase 3 tasks implemented
✅ Database schema updated with migrations
✅ Backend API fully functional and tested
✅ Frontend pages/components created
✅ Permissions properly implemented
✅ No breaking changes to existing functionality
✅ Code follows existing patterns and conventions
✅ Audit logging for sensitive operations
✅ Documentation updated

Next Steps

1. ✅ Create feature branch
2. ✅ Create implementation plan (this document)
3. ⏭️ Update Prisma schema
4. ⏭️ Run migrations
5. ⏭️ Implement backend API
6. ⏭️ Create frontend components
7. ⏭️ Add permissions
8. ⏭️ Test implementation
9. ⏭️ Push to GitHub dev branch

Implementation Progress: 0% → Target: 100%
Estimated Time: 3 weeks
Priority: HIGH (Phase 3)