Software Requirements Specification (SRS)

Flow Management System (FMS)

---

1. Executive Summary

1.1 Purpose

We are seeking an experienced Full-Stack Developer to build a **Flow Management System (FMS)** - a web-based workflow automation platform designed for managing email marketing operations, task assignments, and team performance tracking.

1.2 Project Overview

The FMS is a role-based workflow management application that enables organizations to:

- Create reusable workflow templates with sequential tasks
- Spawn process instances from templates for specific clients/campaigns
- Assign tasks to team members with SLA tracking
- Monitor real-time progress and team performance
- Manage approvals and quality control

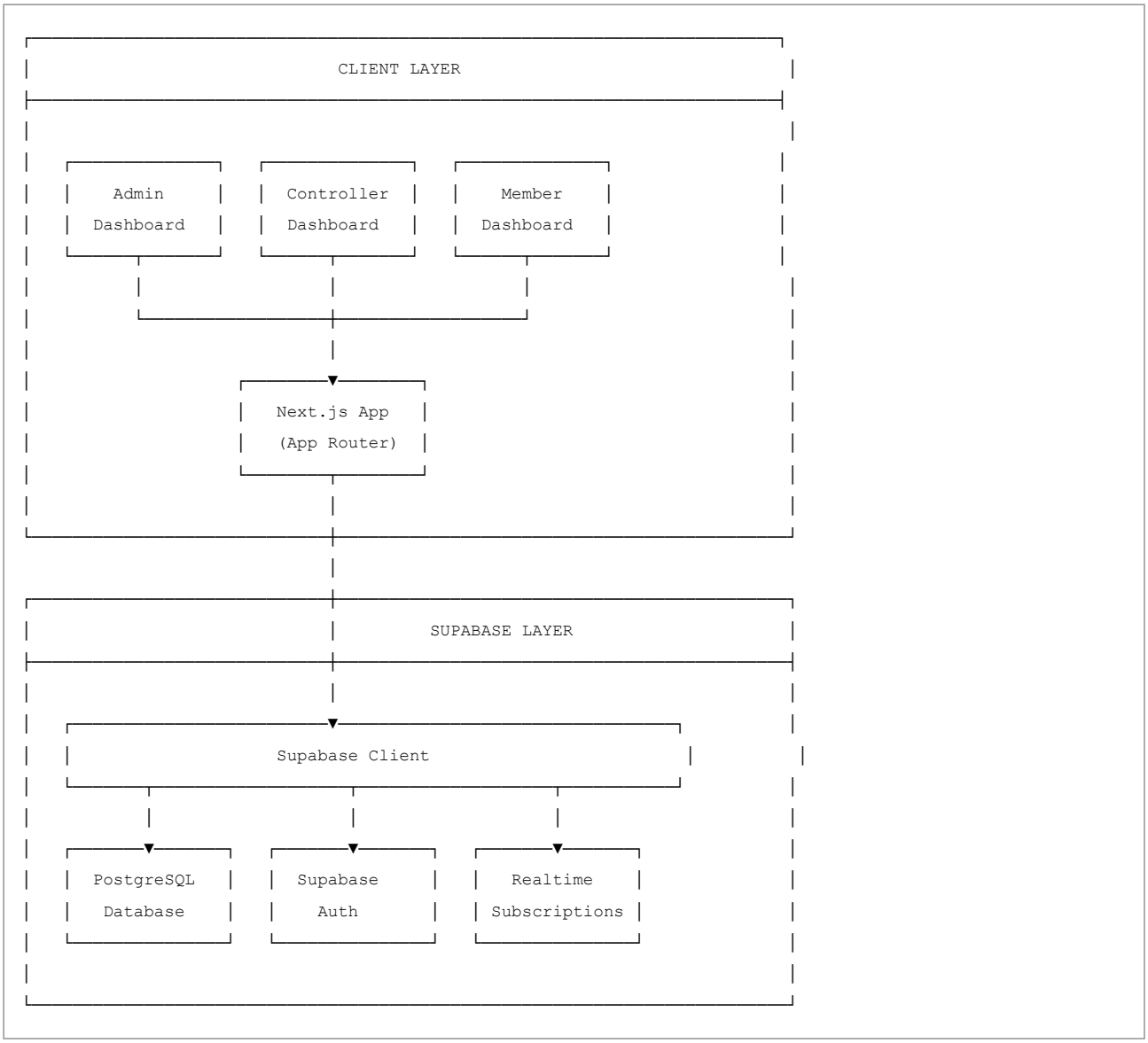
1.3 Technology Stack (Required)

Layer	Technology
Frontend	Next.js 14+ (App Router), React 18+, TypeScript
UI Framework	Tailwind CSS, shadcn/ui components
State Management	React Query (TanStack Query)
Backend/Database	Supabase (PostgreSQL + Auth + Real-time)
Real-time	Supabase Realtime subscriptions
Deployment	Vercel (preferred)

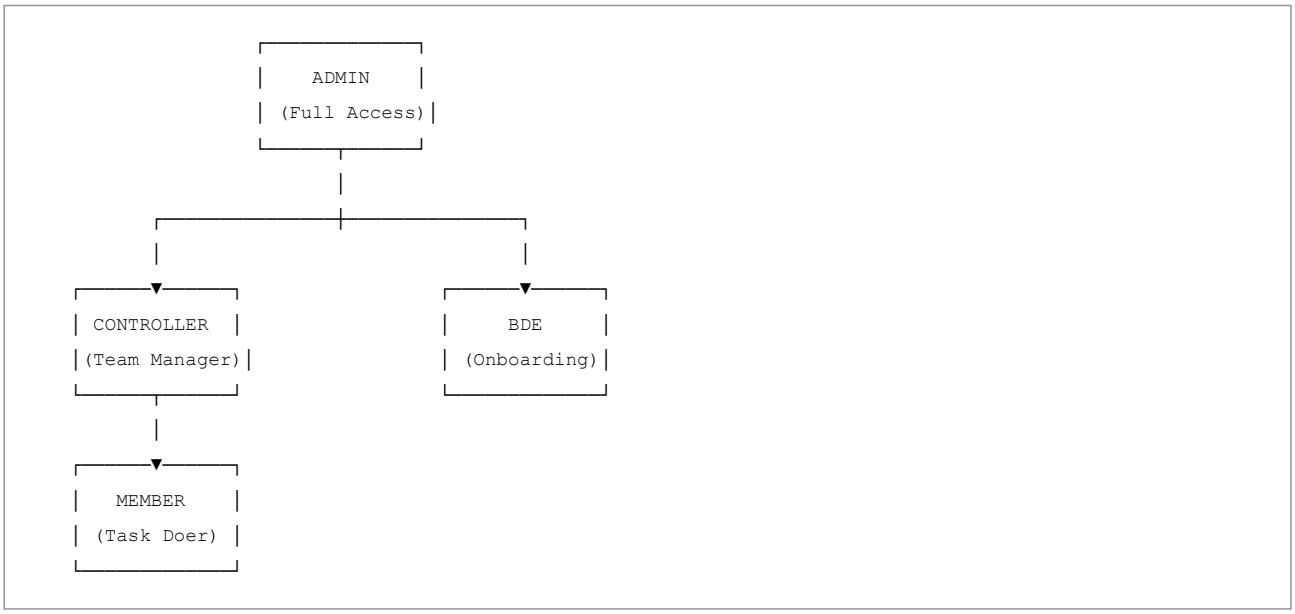
---

2. System Architecture

2.1 High-Level Architecture Diagram



2.2 User Role Hierarchy



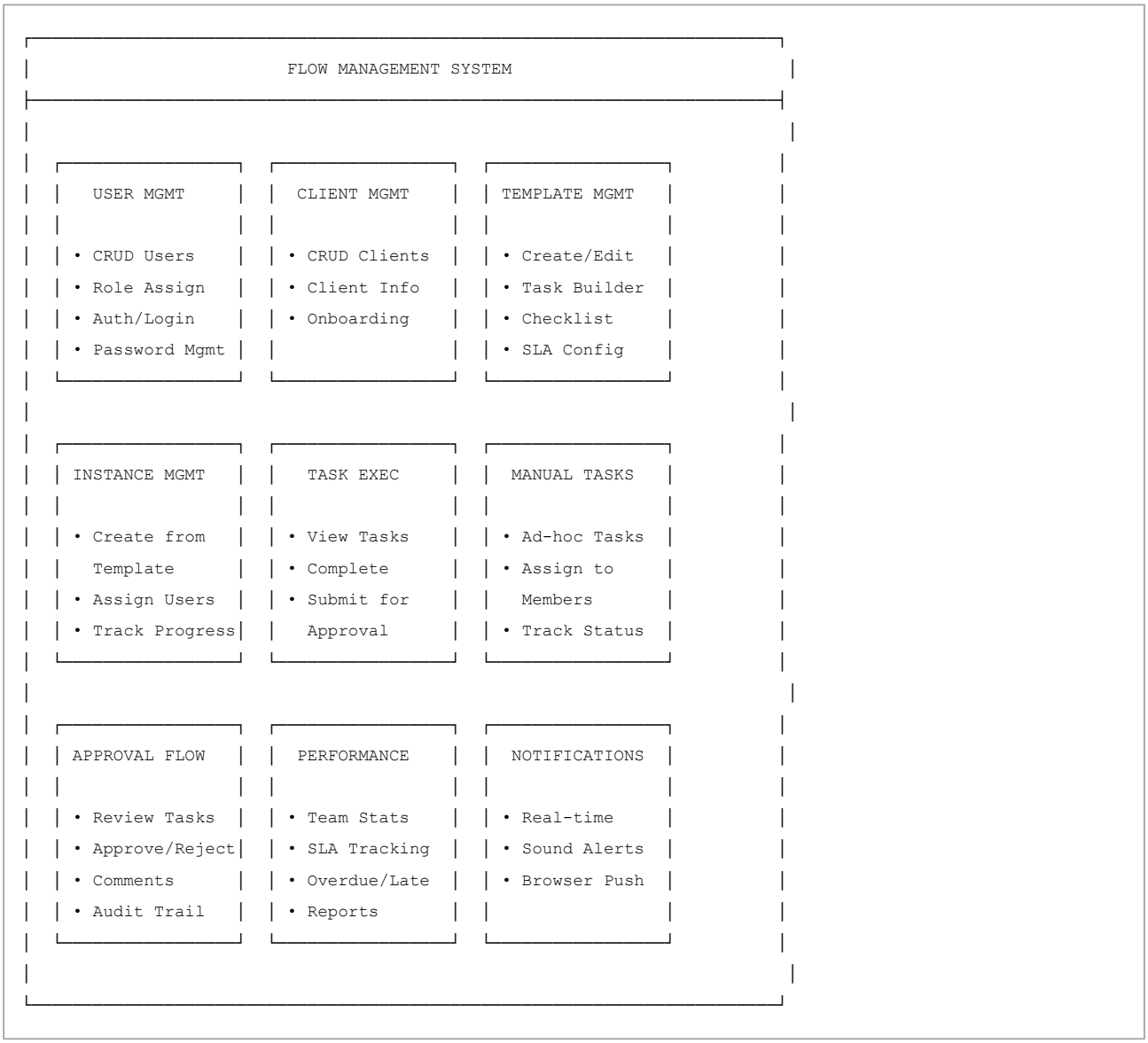
2.3 Role Permissions Matrix

Feature	Admin	Controller	Member	BDE
User Management	□ Full	□ View/Limited Edit	□	□
Client Management	□ Full	□ View	□	□ Onboarding
Template Management	□ Full	□ View	□	□
Create Instances	□	□	□	□
Assign Tasks	□	□	□	□
Execute Tasks	□	□ Own	□ Own	□
Approve Tasks	□	□	□	□
View Team Performance	□	□	□	□
Manual Tasks	□	□ Create/Assign	□ View Own	□
Settings	□	□	□	□

---

3. Core Modules

3.1 Module Overview Diagram



---

4. Detailed Module Requirements

4.1 Authentication & User Management

4.1.1 Requirements

- Email/password authentication via Supabase Auth
- Role-based access control (Admin, Controller, Member, BDE)
- User CRUD operations (Admin only)
- Password reset functionality
- Session management with auto-refresh
- Two-table sync: Supabase Auth ↔ Users table

4.1.2 User Interface

- Login page with email/password form
- User list with pagination and sorting
- Add/Edit user modal
- Password update functionality
- Active/Inactive toggle

4.1.3 Database Schema

```
CREATE TABLE users (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  auth_id UUID REFERENCES auth.users(id),
  email VARCHAR(255) UNIQUE NOT NULL,
  name VARCHAR(255) NOT NULL,
  role VARCHAR(50) NOT NULL CHECK (role IN ('admin', 'controller', 'member', 'bde')),
  is_active BOOLEAN DEFAULT true,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

4.2 Client Management

4.2.1 Requirements

- CRUD operations for clients
- Client information fields:
    - Client name, company name
    - Contact person, email, phone
    - Address, location, timezone
    - Website, notes
- Active/Inactive status
- Client onboarding workflow (BDE role)

4.2.2 Database Schema

```
CREATE TABLE clients (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  client_name VARCHAR(255) NOT NULL,
  company_name VARCHAR(255),
  contact_person VARCHAR(255),
  email VARCHAR(255),
  phone_number VARCHAR(50),
  address TEXT,
  location VARCHAR(255),
  timezone VARCHAR(100),
  website VARCHAR(255),
  notes TEXT,
  is_active BOOLEAN DEFAULT true,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

4.3 Template Management

4.3.1 Requirements

- Create workflow templates with multiple sequential tasks
- Each template includes:
  - Name, category, description
  - Active/Inactive status
  - List of ordered tasks
- Each task includes:
  - Task name and description
  - Duration (in minutes)
  - Turnaround time/SLA (configurable in hours or days)
  - Requires approval flag
  - Checklist items (optional, with input fields)
- Reorder tasks via up/down arrows
- Activate/deactivate templates
- Templates are reusable for multiple instances

4.3.2 Template Task Flow Diagram

4.3.3 Database Schema

```
CREATE TABLE process_templates (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  category VARCHAR(100),
  description TEXT,
  type VARCHAR(50) DEFAULT 'one-time',
  is_active BOOLEAN DEFAULT true,
  owner_id UUID REFERENCES users(id),
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE template_tasks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  template_id UUID REFERENCES process_templates(id) ON DELETE CASCADE,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  "order" INTEGER NOT NULL,
  task_duration_minutes INTEGER DEFAULT 30,
  sla_hours INTEGER DEFAULT 24,
  requires_approval BOOLEAN DEFAULT false,
  default_role VARCHAR(50),
  checklist JSONB DEFAULT '[]',
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

4.3.4 Checklist JSON Structure

```
[
  {
    "id": "uuid",
    "text": "Grammar and spelling check",
    "required": true
  },
  {
    "id": "uuid",
    "text": "Design link",
    "required": true,
    "hasInput": true,
    "inputLabel": "Enter Figma/Design URL"
  }
]
```

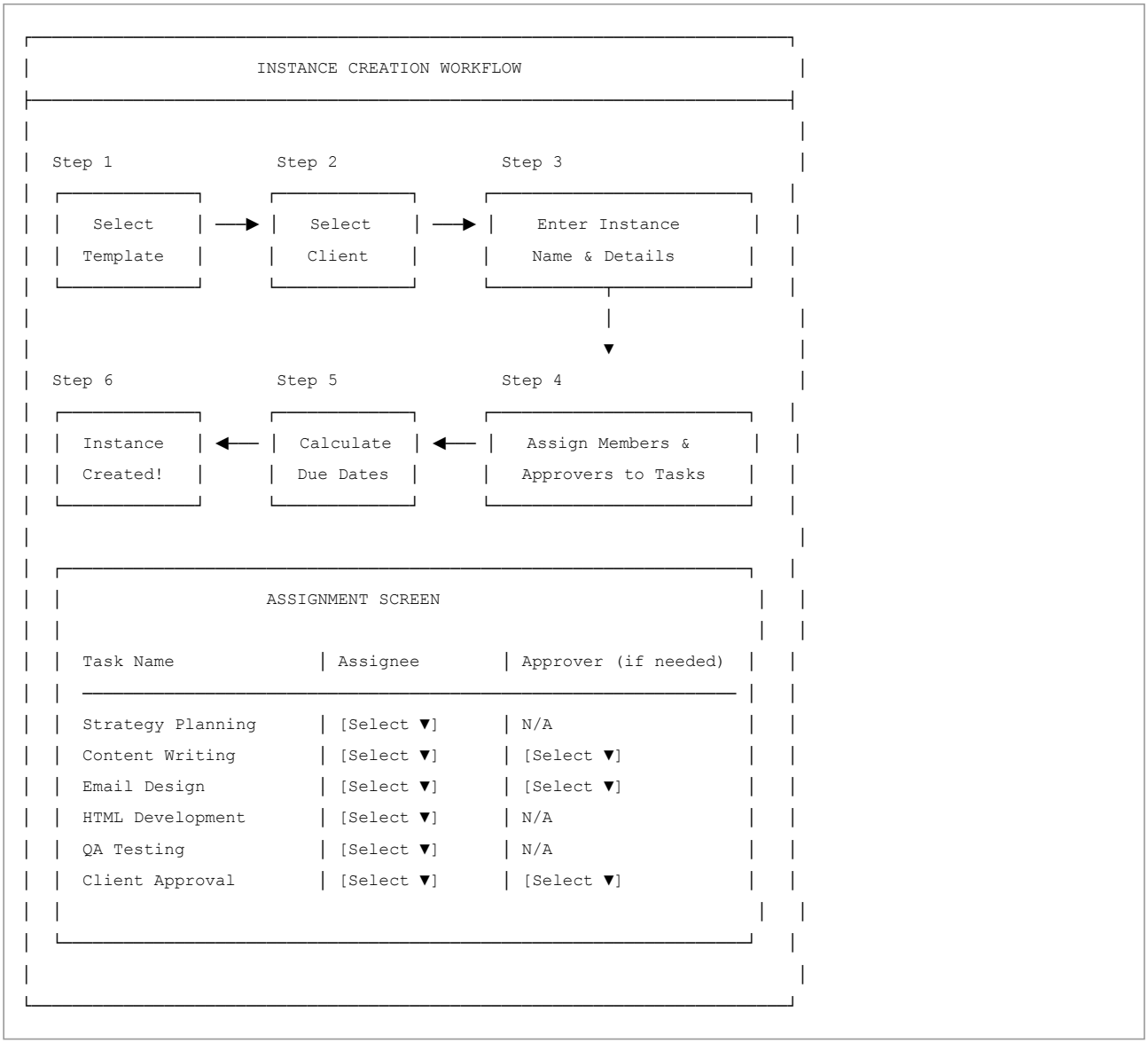
4.4 Process Instance Management

4.4.1 Requirements

- Create instances from templates
- Select client for the instance
- Assign specific team members to each task
- Assign approvers for tasks requiring approval
- Calculate due dates based on:

    - Working days configuration
    - Holiday calendar
    - Task duration and SLA
- Track instance progress (percentage complete)
- Instance statuses: `active`, `completed`, `archived`

4.4.2 Instance Creation Flow



4.4.3 Database Schema

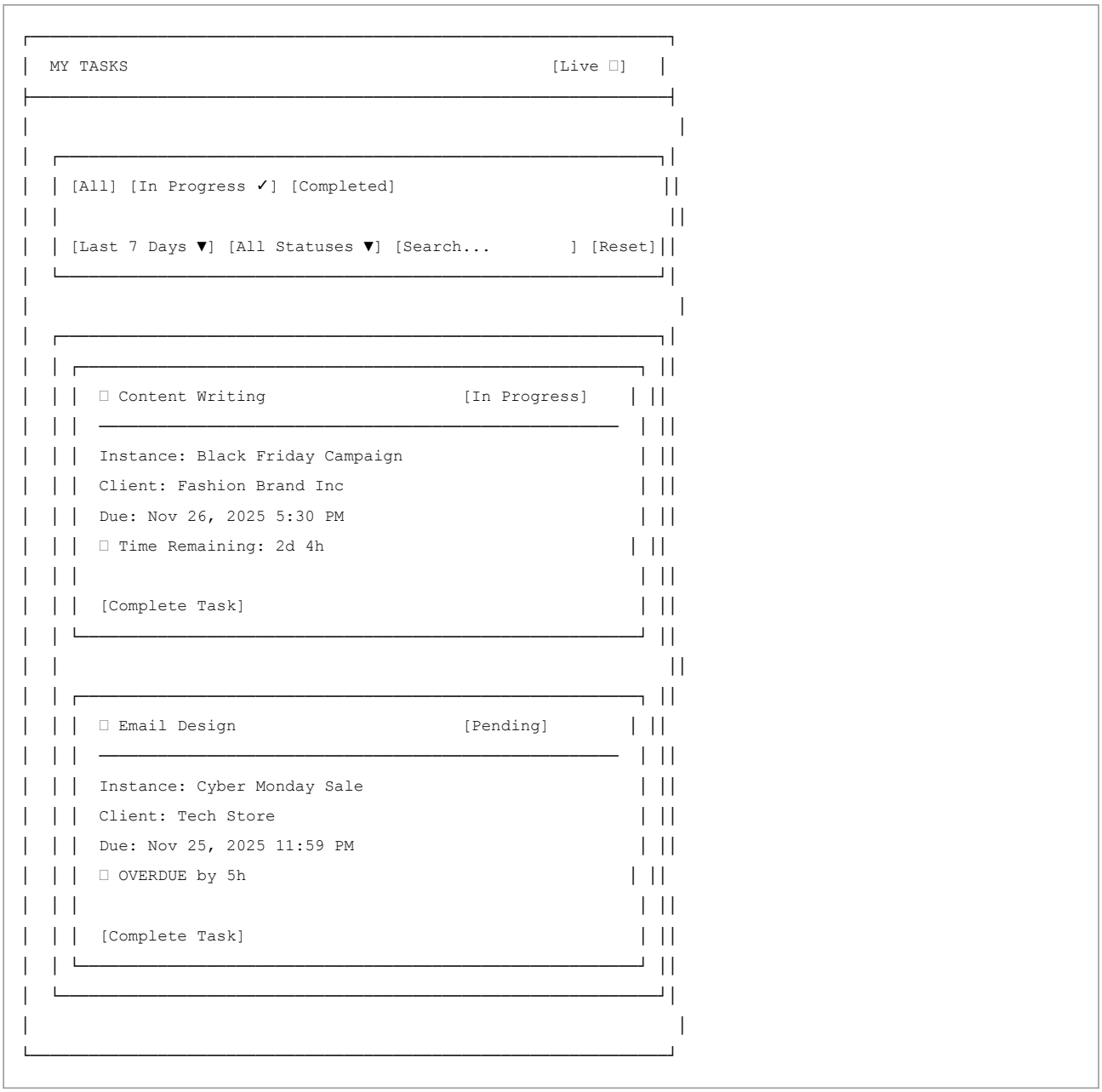
```
CREATE TABLE process_instances (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  template_id UUID REFERENCES process_templates(id),
  client_id UUID REFERENCES clients(id),
  name VARCHAR(255) NOT NULL,
  status VARCHAR(50) DEFAULT 'active' CHECK (status IN ('active', 'completed', 'archived')),
  current_task_index INTEGER DEFAULT 0,
  progress DECIMAL(5,2) DEFAULT 0,
  started_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  completed_at TIMESTAMP WITH TIME ZONE,
  created_by UUID REFERENCES users(id),
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE instance_task_statuses (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  instance_id UUID REFERENCES process_instances(id) ON DELETE CASCADE,
  template_task_id UUID REFERENCES template_tasks(id),
  assigned_to_user_id UUID REFERENCES users(id),
  approver_id UUID REFERENCES users(id),
  status VARCHAR(50) DEFAULT 'not_started'
    CHECK (status IN ('not_started', 'pending', 'in_progress', 'pending_approval', 'completed', 'rejected')),
  due_date TIMESTAMP WITH TIME ZONE,
  estimated_hours DECIMAL(5,2),
  started_at TIMESTAMP WITH TIME ZONE,
  completed_at TIMESTAMP WITH TIME ZONE,
  checklist_values JSONB DEFAULT '[]',
  comments JSONB DEFAULT '[]',
  deliverable_link TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

4.5 Task Execution (Member View)

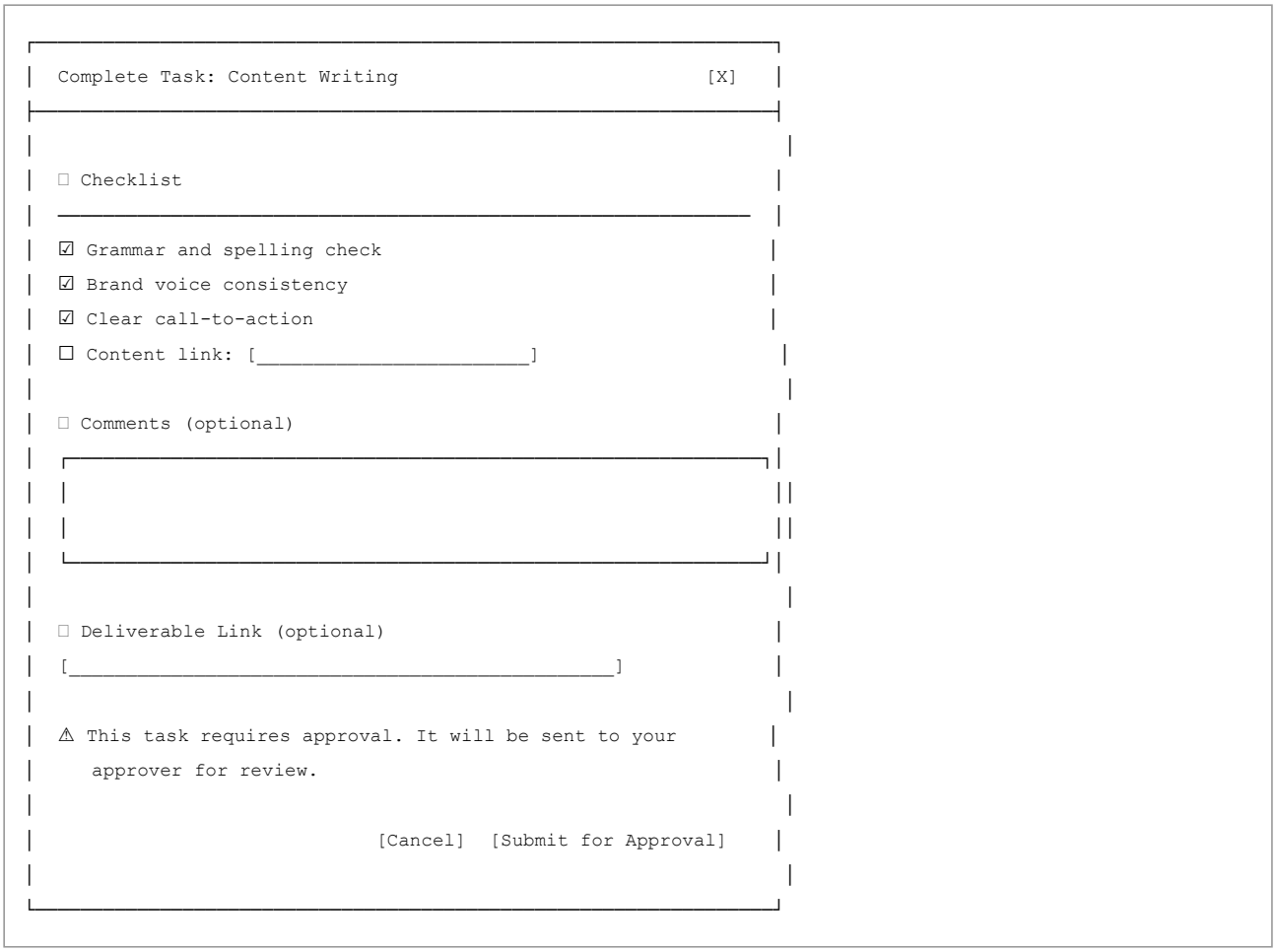
4.5.1 Requirements

- View assigned tasks with filters:
  - Status tabs: All, In Progress, Completed (default: In Progress)
  - Date range presets: Today, Yesterday, Last 7 Days, Last 30 Days, Custom
  - Status filter dropdown
  - Search by task name, instance, client
- Task card displays:
  - Task name, instance name, client name
  - Due date and time remaining
  - Overdue indicator (red) if past due
  - Late badge if completed after due date
- Complete task workflow:
  - Fill checklist items (if required)
  - Add optional comments/deliverable links
  - Submit for approval (if required) or mark complete
- Real-time updates when new tasks are assigned

4.5.2 Task Card UI



4.5.3 Task Completion Dialog

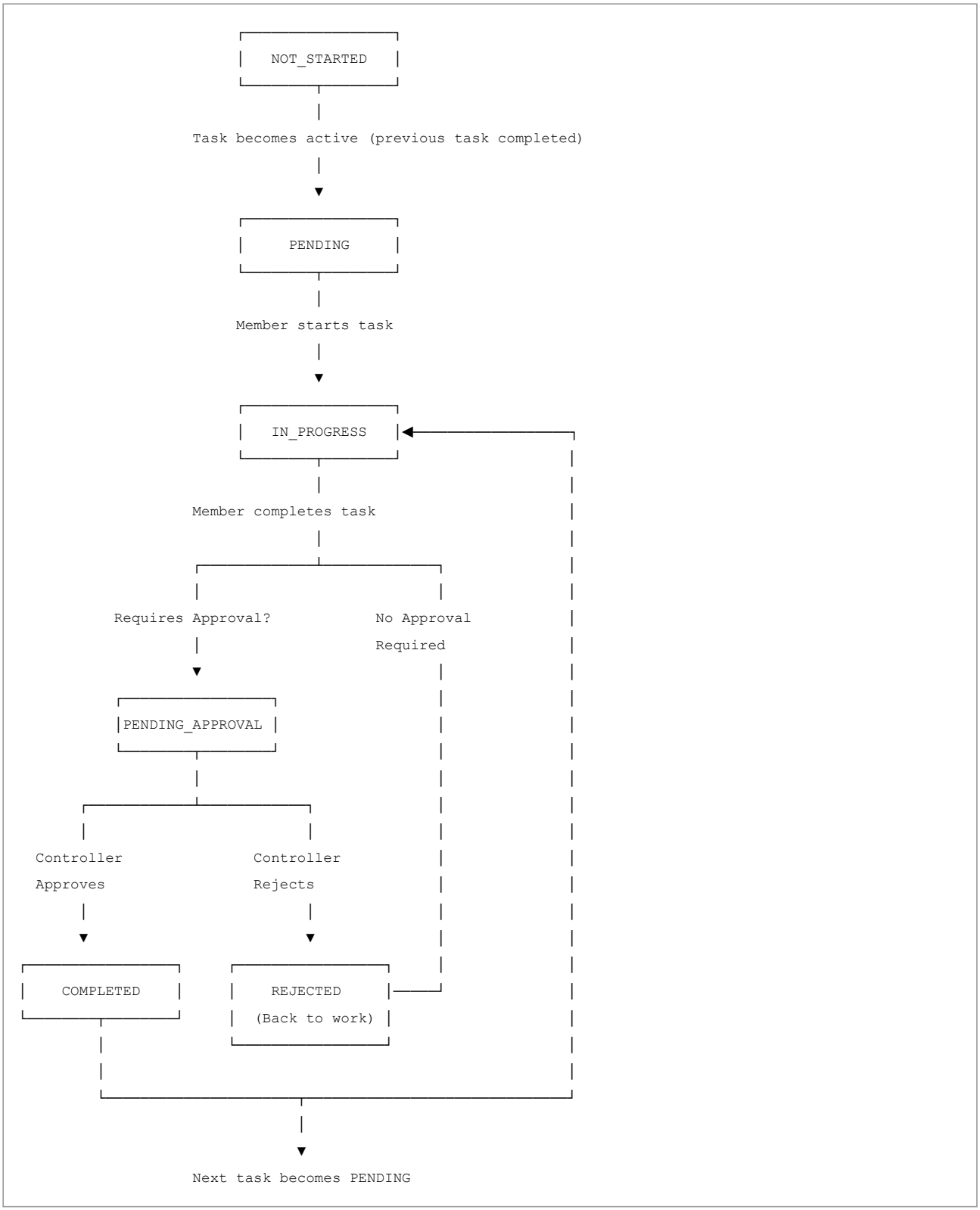


4.6 Approval Workflow

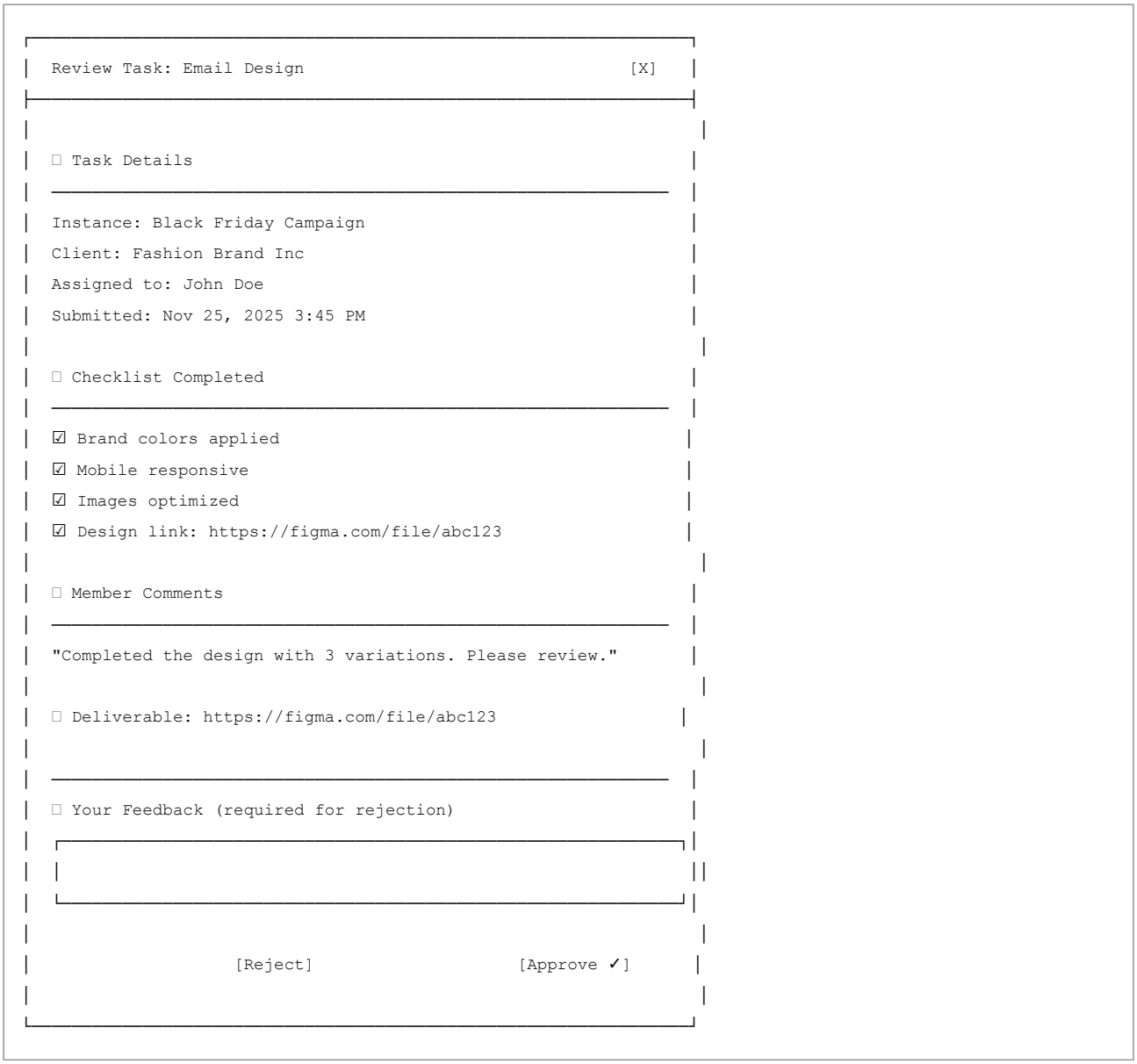
4.6.1 Requirements

- Controllers see tasks pending their approval
- Review completed work:
  - View filled checklist items
  - View comments/deliverables
  - See task history
- Approve or Reject with comments
- On approval: Task marked complete, next task in sequence activates
- On rejection: Task returns to assignee with feedback

4.6.2 Approval State Diagram



4.6.3 Approval Review Dialog



4.7 Manual Tasks

4.7.1 Requirements

- Create ad-hoc tasks not tied to templates
- Assign to any team member
- Set due date, priority, duration
- Track status independently
- Visible in member's task list

4.7.2 Database Schema

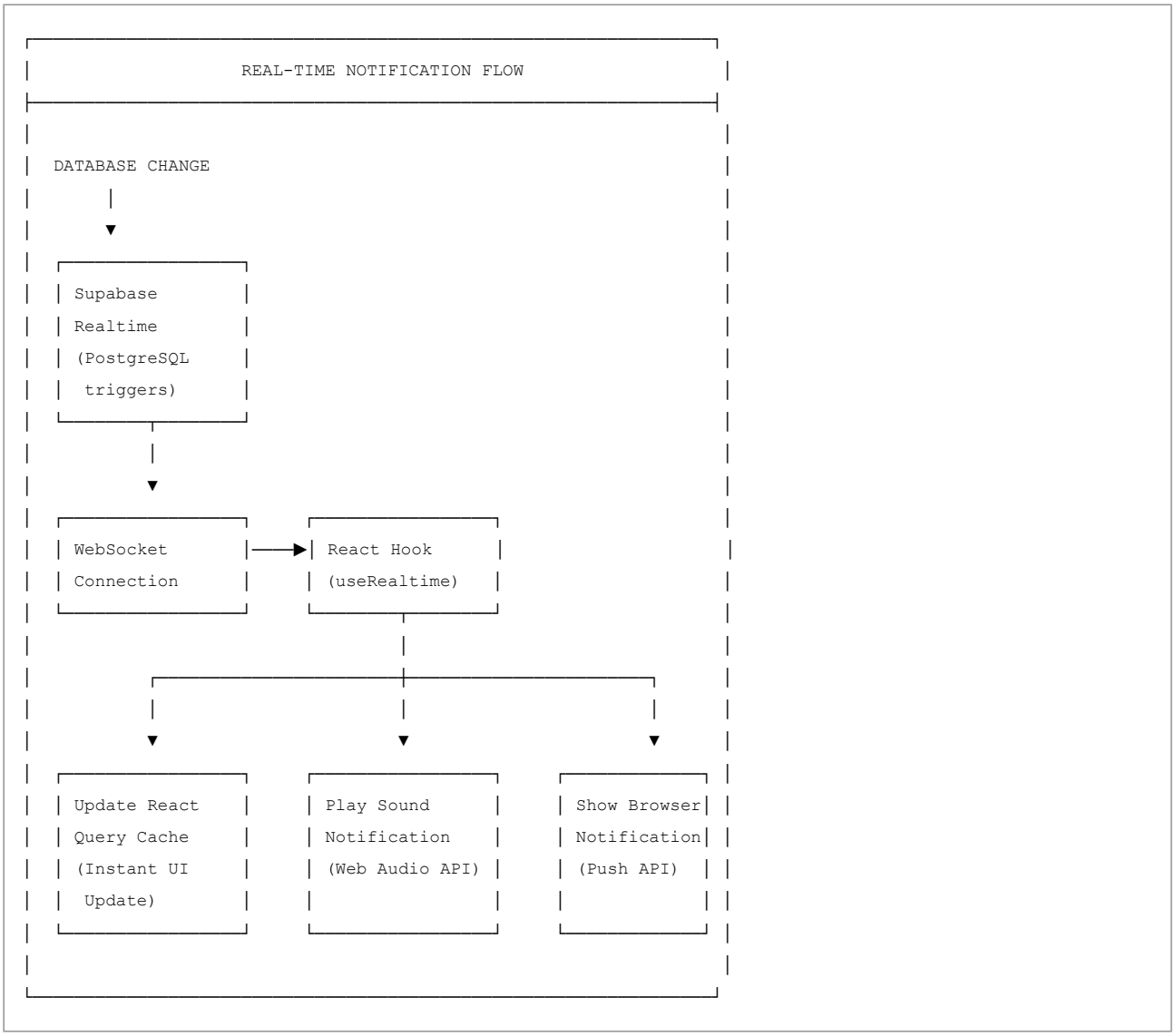
```
CREATE TABLE manual_tasks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  title VARCHAR(255) NOT NULL,
  description TEXT,
  assigned_to UUID REFERENCES users(id),
  created_by UUID REFERENCES users(id),
  due_date TIMESTAMP WITH TIME ZONE,
  priority VARCHAR(20) DEFAULT 'medium' CHECK (priority IN ('low', 'medium', 'high', 'urgent')),
  status VARCHAR(50) DEFAULT 'pending' CHECK (status IN ('pending', 'in_progress', 'completed', 'cancelled')),
  estimated_minutes INTEGER,
  turnaround_hours INTEGER,
  started_at TIMESTAMP WITH TIME ZONE,
  completed_at TIMESTAMP WITH TIME ZONE,
  archived BOOLEAN DEFAULT false,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

4.8 Real-Time Notifications

4.8.1 Requirements

- Supabase Realtime subscriptions for:
  - New task assignments
  - Task status changes
  - Approval/rejection notifications
  - Instance progress updates
- Browser notifications (with permission request)
- Sound notifications:
  - New task assigned: Double beep
  - Task approved: Rising tone
  - Task rejected: Alert/urgent tone
- Visual indicators:
  - "Live" badge when connected (green)
  - Pulse animation on updates
  - Toast notifications

4.8.2 Notification Flow

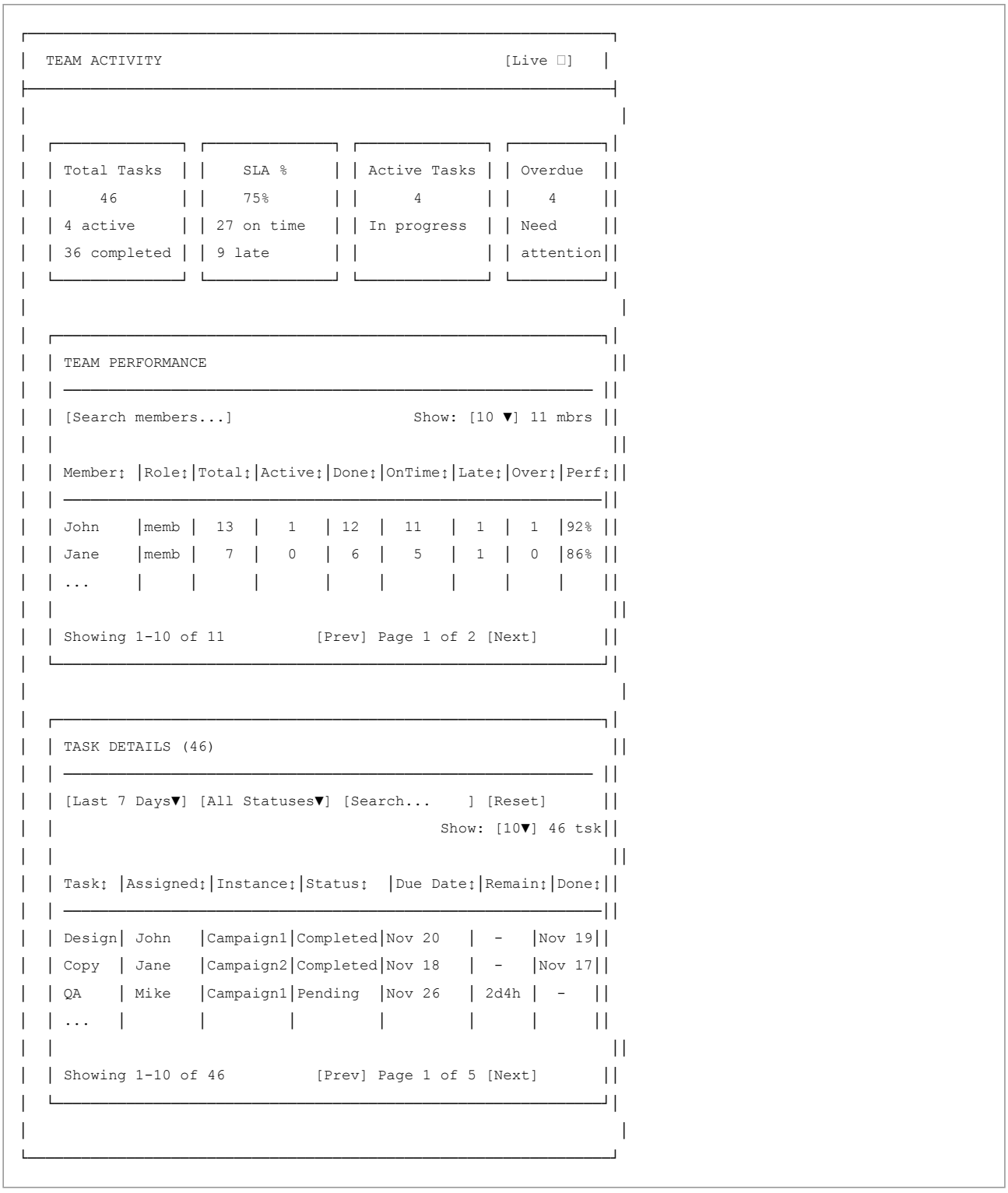


4.9 Performance Metrics & Reporting

4.9.1 Requirements

- Team Performance table with sortable columns:
  - Member name, role
  - Total tasks, active, completed
  - On-time, late, overdue counts
  - Completion rate percentage (progress bar)
- Task Details table with:
  - Date range filter presets (default: Last 7 Days)
  - Status filter
  - Search
  - All columns sortable
  - Pagination
  - Late badge for tasks completed after due date
- Summary cards:
  - Total tasks
  - SLA performance percentage
  - Active tasks count
  - Overdue tasks count

4.9.2 Performance Dashboard Layout



4.10 Business Calendar Configuration

4.10.1 Requirements

- Configure working days (checkboxes for each day)
- Add/manage holidays with date picker
- Due date calculations must respect:
    - Non-working days (skip weekends if configured)
    - Holidays (skip holiday dates)

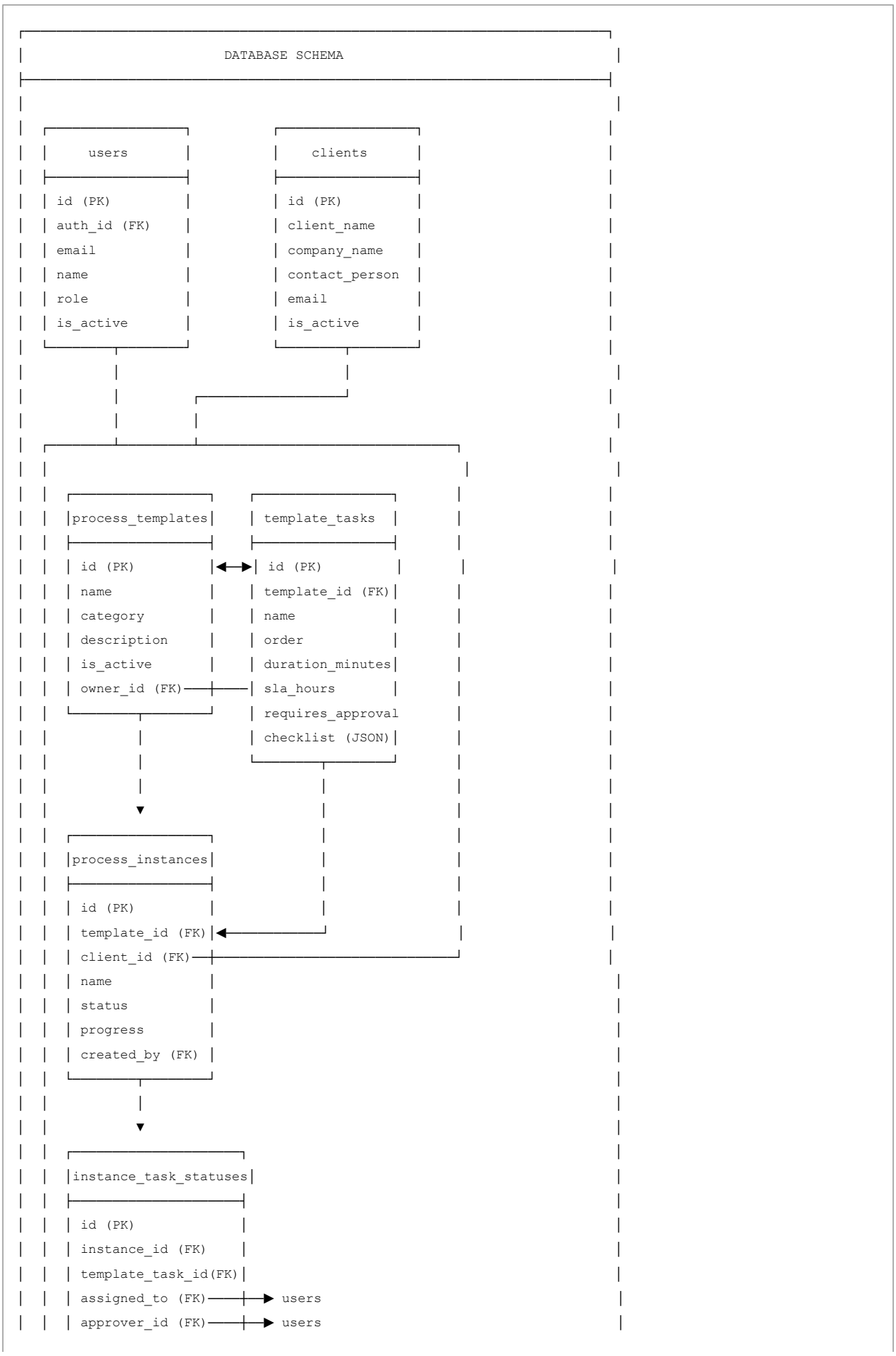
- All times in IST (GMT+5:30)

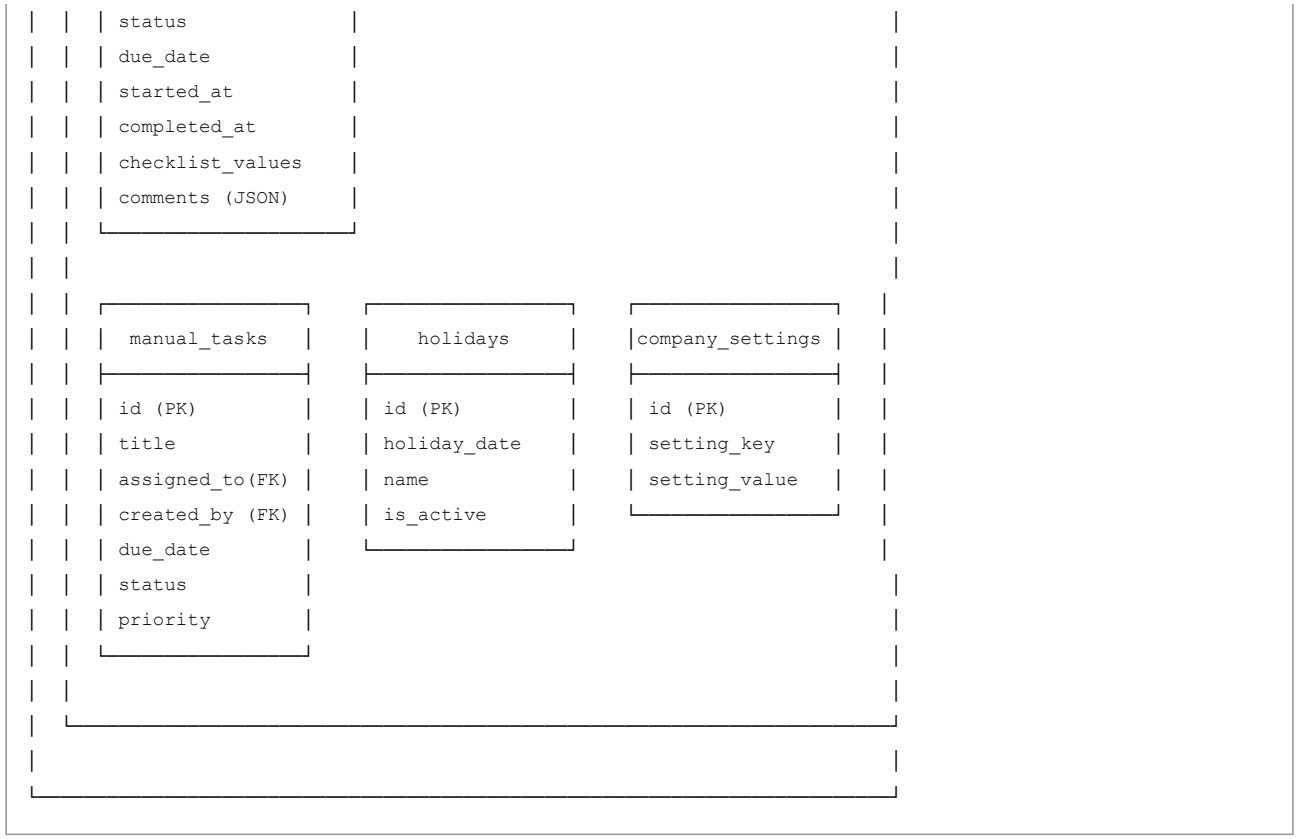
4.10.2 Database Schema

```
CREATE TABLE company_settings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  setting_key VARCHAR(100) UNIQUE NOT NULL,
  setting_value JSONB NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Example: working_days setting
-- { "monday": true, "tuesday": true, "wednesday": true,
--   "thursday": true, "friday": true, "saturday": false, "sunday": false }

CREATE TABLE holidays (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  holiday_date DATE NOT NULL,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  is_active BOOLEAN DEFAULT true,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

---

5. Complete Database Schema Diagram


DATABASE SCHEMA
users
id (PK)
auth_id (FK)
email
name
role
is_active
clients
id (PK)
client_name
company_name
contact_person
email
is_active
process_templates
id (PK)
name
category
description
is_active
owner_id (FK)
template_tasks
id (PK)
template_id (FK)
name
order
duration_minutes
sla_hours
requires_approval
checklist (JSON)
process_instances
id (PK)
template_id (FK)
client_id (FK)
name
status
progress
created_by (FK)
instance_task_statuses
id (PK)
instance_id (FK)
template_task_id(FK)
assigned_to (FK)
approver_id (FK)
users
users



6. UI/UX Requirements

6.1 Design Principles

- Clean, modern interface using shadcn/ui components
- Responsive design (desktop-first, mobile-friendly)
- Consistent color scheme and typography
- Clear visual hierarchy
- Intuitive navigation with sidebar

6.2 Required UI Components

- Dashboard stat cards with icons
- Data tables with sorting, filtering, pagination
- Modal dialogs for forms (create/edit)
- Toast notifications (success, error, info)
- Dropdown menus and select components
- Progress bars (linear and circular)
- Status badges (color-coded)
- Date pickers with preset options
- Search inputs with clear button
- Tabs for navigation within pages
- Collapsible sidebar

6.3 Color Scheme (Suggested)

Element	Color
Primary	Blue (#3B82F6)
Success/Completed	Green (#22C55E)

Element	Color
Warning/Late	Orange (#F97316)
Error/Overdue	Red (#EF4444)
Pending	Gray (#6B7280)
In Progress	Blue (#3B82F6)
Background	White/Light Gray
Text	Dark Gray (#1F2937)

---

7. Deliverables

7.1 Required Deliverables

#	Deliverable	Description
1	**Source Code**	Complete Next.js application with TypeScript
2	**Database Schema**	Supabase migrations and seed data
3	**Environment Setup**	`.env.example` with required variables
4	**Documentation**	README with setup instructions
5	**Deployment**	Working deployment on Vercel
6	**Testing**	Basic test coverage for critical flows

7.2 Project Milestones

Phase	Deliverables	Duration
Phase 1: Foundation	Auth, User Management, Client Management, Basic Layout	1 week
Phase 2: Templates	Template Builder, Task Configuration, Checklist Builder	1 week
Phase 3: Instances	Instance Creation, Task Assignment, Due Date Calculation	1 week
Phase 4: Execution	Task Execution, Approval Workflow, Status Updates	1 week
Phase 5: Real-time	Real-time Updates, Notifications, Sound Alerts	1 week
Phase 6: Polish	Performance Metrics, Testing, Bug Fixes, Deployment	1 week

Total Estimated Duration: 6 weeks

---

8. Technical Requirements

8.1 Performance Requirements

- Initial page load: < 3 seconds
- Subsequent navigation: < 1 second
- Real-time updates: < 1 second latency
- Support 50+ concurrent users
- Handle 1000+ tasks without performance degradation

8.2 Security Requirements

- Row Level Security (RLS) on all Supabase tables
- Secure authentication via Supabase Auth
- Role-based access control enforced at API level
- Input validation and sanitization
- HTTPS only
- Secure session management

8.3 Browser Support

- Chrome (latest 2 versions)
- Firefox (latest 2 versions)
- Safari (latest 2 versions)
- Edge (latest 2 versions)

8.4 Code Quality

- TypeScript strict mode
- ESLint configuration
- Prettier formatting
- Component-based architecture
- Reusable hooks for data fetching
- Proper error handling

---

9. Evaluation Criteria

Candidates will be evaluated on:

Criteria	Weight	Description
Next.js/React Expertise	25%	Proficiency with App Router, Server Components, React Query
Supabase Experience	20%	Database design, Auth, Realtime, RLS policies
UI/UX Quality	20%	Clean design, responsive, intuitive navigation
Code Architecture	15%	Clean code, proper abstractions, maintainability
Real-time Implementation	10%	Efficient subscriptions, proper cleanup
Communication	10%	Clear updates, timeline adherence, documentation

---

10. How to Apply

10.1 Required Materials

1. **Portfolio** - 2-3 similar projects demonstrating:
    - Next.js/React applications
    - Database-driven applications
    - Real-time features (preferred)
2. **Technical Assessment** - Build a mini prototype with:
    - Login page with Supabase Auth
    - Simple template builder (add/edit/delete tasks)
    - Display templates in a sortable table
    - Basic responsive design
3. **Proposal** - Include:
    - Your timeline estimate
    - Hourly/fixed rate
    - Availability
    - Any questions or clarifications

10.2 Submission

Send your application to: [YOUR EMAIL HERE]

Include:

- Portfolio links
- GitHub profile
- Assessment project (GitHub repo or deployed link)
- Brief cover letter

---

11. Appendix

11.1 Glossary

Term	Definition
Template	A reusable workflow definition with ordered tasks
Instance	A specific execution of a template for a client
Task	A single unit of work within a workflow
SLA	Service Level Agreement - the turnaround time for a task
Checklist	A list of items to complete within a task
Approver	The person who reviews and approves completed work

11.2 Sample Data

Sample Template:

```
{
  "name": "Email Campaign Workflow",
  "category": "Email Marketing",
  "tasks": [
    { "name": "Strategy", "duration": 120, "sla_hours": 24, "approval": false },
    { "name": "Content", "duration": 240, "sla_hours": 48, "approval": true },
    { "name": "Design", "duration": 180, "sla_hours": 48, "approval": true },
    { "name": "Development", "duration": 120, "sla_hours": 24, "approval": false },
    { "name": "QA", "duration": 60, "sla_hours": 8, "approval": false },
    { "name": "Client Approval", "duration": 30, "sla_hours": 24, "approval": true }
  ]
}
```

Document Version: 1.0

Created: November 2025

Last Updated: November 25, 2025

End of Document