# Codebase Analysis: Hakhel - Multi-Tenant Yahrzeit Memorial Management System

## 🏗️ Architecture Overview

This is a **Rails 8 application** built on **Jumpstart Pro** that manages yahrzeit (Jewish memorial anniversary) notifications for multiple communities. It combines:

1. **Jumpstart Pro's SaaS foundation** - Multi-tenant account management, billing, authentication
2. **HKE (Hakhel) Engine** - Custom memorial/yahrzeit management system integrated into the main app

## 🎯 Core Business Domain: HKE (Hakhel)

The application manages **memorial notifications** for Jewish communities with these key entities:

### Primary Models ( `app/models/hke/` )

#### 1. Community

- Synagogues or schools that use the system
- Each has an associated `Account` (Jumpstart's multi-tenancy)
- Types: `synagogue` or `school`
- Has addresses and preferences

#### 2. DeceasedPerson

- People being memorialized
- Hebrew death dates (day, month, year)
- Cemetery information
- Parental names (for Hebrew naming)
- Relations to contact people

### 3. ContactPerson

- Family members who receive notifications
- Phone, email for notifications
- Gender (for Hebrew message customization)
- Relations to deceased people

### 4. Relation

- Join model connecting deceased ↔ contact
- Relationship type (father, mother, spouse, etc.)
- Generates future messages
- Has secure token for tracking

### 5. FutureMessage

- Scheduled yahrzeit notifications
- **Approval workflow** (pending/approved/rejected)
- Delivery methods: SMS, WhatsApp, Email
- Rendered message content
- Send date calculation

### 6. SentMessage

- Historical record of sent notifications
- Twilio message SID for tracking
- Delivery status
- Copy of message content

# 🔐 Multi-Tenancy Architecture

The app uses **TWO PARALLEL** tenancy systems:

## 1. Jumpstart's Account-Based Tenancy

- **Account** model - Teams or personal accounts
- **AccountUser** - Join table with roles (admin, member)
- Users can belong to multiple accounts
- Switching between accounts via cookies/subdomains

## 2. HKE's Community-Based Tenancy

- **Community** model - Each community has one Account
- All HKE models inherit from `CommunityRecord`
- `CommunityRecord` uses `acts_as_tenant :community`
- Automatic scoping: `Hke::DeceasedPerson.all` → only current community's records

```ruby
# app/models/hke/community_record.rb
class Hke::CommunityRecord < Hke::ApplicationRecord
  self.abstract_class = true
  acts_as_tenant :community
  belongs_to :community
end
```

# 👥 User Roles & Authorization

## User Model has custom HKE roles (JSONB column):

- `system_admin` - Full system access, manages all communities
- `community_admin` - Manages their community
- `community_user` - Limited community access

```ruby
# app/models/user.rb
ROLES = [:system_admin, :community_admin, :community_user]
belongs_to :community, class_name: "Hke::Community", optional: true
```

## Authorization Strategy:

- **Jumpstart controllers** → Use `Pundit` with `AccountUser` (account-level roles)
- **HKE controllers** → Use `Pundit` with `User` directly (community-level roles)

```ruby
# app/policies/hke/application_policy.rb
def initialize(user, record)
  raise Pundit::NotAuthorizedError, "must be logged in" unless user
  @user = user
  @record = record
end
```

# 📨 Message Processing System

## Workflow:

1. **Generation** - `Relation` model generates `FutureMessage` records
2. **Approval** - Community admins approve/reject messages
3. **Sending** - `MessageProcessor` service sends via Twilio/SendGrid
4. **Recording** - Creates `SentMessage` with Twilio SID

## Key Services:

- `Hke::MessageProcessor` - Orchestrates sending
- `Hke::TwilioSend` - Handles SMS/WhatsApp/Email delivery
- `Hke::LiquidRenderer` - Renders message templates
- `Hke::MessageGenerator` - Generates messages from relations

```
# app/services/hke/message_processor.rb
def call
  @future_message.full_message = @future_message.rendered_full_message(reference_dat
  message_sids = send_message(methods: delivery_methods, future_message: @future_mes
  Hke::SentMessage.create!(...)
end
```

# 📊 Data Import System

- **CsvImport** - Tracks bulk import jobs
- **CsvImportLog** - Detailed row-level logging
- API endpoints for async import processing
- Deduplication logic via `Hke::Deduplicatable` concern

# 🗄️ Database Schema Highlights

## HKE Tables (all prefixed `hke_`)

- `hke_communities` - Community records
- `hke_deceased_people` - Memorial records
- `hke_contact_people` - Recipients
- `hke_relations` - Deceased ↔ Contact links
- `hke_future_messages` - Scheduled notifications (with approval_status)
- `hke_sent_messages` - Historical sends
- `hke_preferences` - Delivery preferences (polymorphic)
- `hke_csv_imports` - Import tracking
- `hke_logs` - Audit trail

## Jumpstart Tables

- `accounts`, `account_users` - Multi-tenancy
- `users` - Authentication (Devise)
- `pay_*` - Payment processing (Pay gem)
- `noticed_*` - Notifications

## 🛣️ Routing Structure

```ruby
# config/routes.rb
namespace :hke do
  resources :contact_people
  resources :deceased_people
  resources :future_messages

  namespace :api do
    namespace :v1 do
      resources :csv_imports
      resources :twilio_callback
    end
  end
end
```

## 🔧 Technology Stack

- **Rails 8** with Hotwire (Turbo + Stimulus)
- **PostgreSQL** with JSONB for roles/preferences
- **Devise** for authentication
- **Pundit** for authorization
- **Pay gem** for payments (currently disabled)
- **acts_as_tenant** for multi-tenancy scoping
- **Twilio** for SMS/WhatsApp
- **SendGrid** for email
- **Liquid** for message templates
- **Sidekiq** for background jobs
- **TailwindCSS v4** for styling

# 🎨 Key Patterns & Concerns

## 1. Modular Models - Concerns for shared behavior:

- `Hke::Deduplicatable` - Prevents duplicate records
- `Hke::LogModelEvents` - Audit trail
- `Hke::Addressable` - Polymorphic addresses
- `Hke::Preferring` - Polymorphic preferences

## 2. Service Objects - Complex operations:

- `MessageProcessor`, `TwilioSend`, `LiquidRenderer`

## 3. Tenant Scoping

- Automatic via `acts_as_tenant`

## 4. Approval Workflow

- Enum-based state machine on `FutureMessage`

# 📁 Directory Structure

```
app/
├── models/hke/          # HKE domain models
├── controllers/hke/     # HKE controllers
├── policies/hke/        # HKE authorization
├── services/hke/        # Business logic services
├── views/hke/           # HKE views
└── jobs/hke/            # Background jobs

lib/jumpstart/           # Jumpstart Pro engine code
├── app/models/          # Account, User concerns
├── app/controllers/     # Billing, auth controllers
└── app/policies/        # Account-level policies
```

# 🔑 Critical Integration Points

1. **Community ↔ Account** - Each Community creates an Account
2. **User ↔ Community** - Users belong to one community
3. **Session Management** - `session[:selected_community_id]` for system admins
4. **Current Scoping** - `Current.community` set per request

# ⚠️ Notable Design Decisions

1. **Dual Authorization** - Jumpstart uses AccountUser, HKE uses User directly
2. **Approval System** - Messages require approval before sending
3. **Hebrew Date Handling** - Stored as strings (day/month/year)
4. **Message Immutability** - SentMessage preserves exact content sent
5. **Deduplication** - Prevents duplicate deceased/contact/relation records
6. **Liquid Templates** - Flexible message customization

# 🚀 Key Workflows

## User Login Flow

1. User signs in via Devise
2. `after_sign_in_path_for` checks user role
3. System admins → `hke_admin_root_path`
4. Community admins/users → `hke_root_path`

## Message Generation Flow

1. Relation created/updated
2. `process_future_messages` callback triggered
3. `MessageGenerator` calculates yahrzeit dates
4. Creates `FutureMessage` records for upcoming dates
5. Messages enter "pending" approval status

## Message Approval & Sending Flow

1. Community admin reviews pending messages
2. Approves/rejects via dashboard
3. Approved messages picked up by scheduled job
4. `MessageProcessor` sends via Twilio/SendGrid
5. `SentMessage` created with delivery tracking

## CSV Import Flow

1. User uploads CSV file
2. `CsvImport` record created
3. Background job processes rows
4. Deduplication checks prevent duplicates
5. `CsvImportLog` records success/errors
6. Statistics updated on `CsvImport`

# 🔍 Data Model Relationships

```
Community (1) ──┬── (N) DeceasedPerson
                ├── (N) ContactPerson
                ├── (N) Relation
                ├── (N) FutureMessage
                ├── (N) SentMessage
                ├── (N) Cemetery
                ├── (N) Selection
                └── (1) Account

DeceasedPerson (N) ─── (N) ContactPerson
                        (via Relation)

Relation (1) ─── (N) FutureMessage (polymorphic messageable)
         (1) ─── (N) SentMessage (polymorphic messageable)

User (N) ─── (N) Account (via AccountUser)
     (N) ─── (1) Community
```

# 🛡️ Security & Authorization

## Pundit Policies

- **HKE Policies** ( `app/policies/hke/` )
  - Use `User` directly
  - Check `system_admin?` , `community_admin?` , `community_user?`
  - Scope queries by community
- **Jumpstart Policies** ( `lib/jumpstart/app/policies/` )
  - Use `AccountUser` (contains account-specific roles)
  - Check `admin?` role on AccountUser
  - Scope queries by account

## Tenant Isolation

- `acts_as_tenant :community` on all HKE models
- Automatic scoping prevents cross-community data access
- System admins can switch communities via session

# 📝 Configuration

## Jumpstart Config ( `config/jumpstart.rb` )

```
Jumpstart.config = Jumpstart::Configuration.new({
  "application_name" => "hakhel",
  "business_name" => "Hakhel, LLC",
  "domain" => "hakhel.me",
  "support_email" => "david@odeca.net",
  "background_job_processor" => "sidekiq",
  "email_provider" => "sendgrid",
  "account_types" => "both",
  "payment_processors" => [],  # Currently disabled
  "multitenancy" => [],
  "gems" => ["acts_as_tenant"]
})
```

# 🧪 Testing Strategy

- **Minitest** with fixtures
- **System tests** with Capybara + Selenium
- **Parallel execution** enabled
- **WebMock** disables external HTTP
- Test fixtures in `test/fixtures/`

# 📦 Key Dependencies

## Core Rails

- `rails (~> 8.0)`
- `pg` - PostgreSQL
- `puma` - Web server

## Authentication & Authorization

- `devise` - User authentication
- `pundit` - Authorization policies

## Multi-tenancy

- `acts_as_tenant` - Automatic tenant scoping

## Background Jobs

- `sidekiq` - Job processing
- `solid_queue` - Alternative job backend

## Messaging

- `twilio-ruby` - SMS/WhatsApp
- `sendgrid-ruby` - Email delivery

## Frontend

- `hotwire-rails` - Turbo + Stimulus
- `tailwindcss-rails` - Styling
- `importmap-rails` - JavaScript management

## Utilities

- `liquid` - Template rendering
- `noticed` - Notifications
- `pay` - Payment processing (configured but not active)

# 🎯 Summary

This is a **sophisticated multi-tenant SaaS application** for managing memorial notifications with:

- **Dual tenancy** (Jumpstart Accounts + HKE Communities)
- **Role-based access control** (system/community/user levels)
- **Approval workflows** for message sending
- **Multiple delivery channels** (SMS, WhatsApp, Email)
- **Comprehensive audit trails** and logging

- **CSV import** with deduplication
- **Hebrew date handling** for yahrzeit calculations
- **Liquid templates** for message customization

The architecture cleanly separates Jumpstart's account management from HKE's domain-specific community tenancy, allowing for flexible scaling and feature development.

**Generated:** 2026-01-14
**Rails Version:** 8.1
**Jumpstart Pro:** Integrated
**Primary Domain:** Yahrzeit Memorial Management