Phase 2 — Project & Stand Management (Complete Implementation)

This document provides **copy-paste ready** migrations, models, controllers, views, routes, and seed data for Projects and Stands. It assumes Phase-1 foundation (Users, Agents, Clients, etc.) is already in place.

> Files provided below should be created exactly at the specified paths in your Rails 7 app. After creating them run:
>
> ```
> bin/rails db:migrate
> bin/rails db:seed
> bin/rails server
> ```

---

1) Migrations

Create migration files (timestamps used as example; your filenames will differ).

`db/migrate/20251008_create_projects.rb`

```ruby
class CreateProjects < ActiveRecord::Migration[7.0]
  def change
    create_table :projects, id: :uuid do |t|
      t.string :core_project_id
      t.string :name, null: false
      t.string :location
      t.text :description
      t.string :developer_name
      t.integer :total_stands, default: 0, null: false
      t.integer :status, default: 0, null: false
      t.string :map_image_url
      t.timestamps
    end

    add_index :projects, :core_project_id
    add_index :projects, :name
  end
end
```

`db/migrate/20251008_create_stands.rb`

```ruby
class CreateStands < ActiveRecord::Migration[7.0]
  def change
    create_table :stands, id: :uuid do |t|
      t.string :core_stand_id
      t.uuid :project_id, null: false
      t.string :stand_number, null: false
      t.decimal :area_sqm, precision: 12, scale: 2
      t.decimal :price, precision: 14, scale: 2
      t.integer :status, default: 0, null: false
      t.datetime :reserved_at
      t.datetime :sold_at
      t.decimal :latitude, precision: 10, scale: 6
      t.decimal :longitude, precision: 10, scale: 6
      t.datetime :updated_from_core_at
      t.jsonb :attributes, default: {}
      t.timestamps
    end

    add_index :stands, :project_id
    add_index :stands, :core_stand_id
    add_index :stands, [:project_id, :stand_number], unique: true
    add_foreign_key :stands, :projects
  end
end
```

---

2) Models

`app/models/project.rb`

```ruby
class Project < ApplicationRecord
  has_many :stands, dependent: :destroy
  has_many :reservations, through: :stands
  has_many :accounts, through: :stands

  enum status: { active: 0, completed: 1, archived: 2 }

  validates :name, presence: true
  validates :total_stands, numericality: { greater_than_or_equal_to: 0 }

  # helper to compute counts
  def stands_status_counts
    stands.group(:status).count
```

```
  end
end
```

`app/models/stand.rb`

```
class Stand < ApplicationRecord
  belongs_to :project
  has_many :reservations
  has_one :account

  enum status: { available: 0, reserved: 1, sold: 2 }

  validates :stand_number, presence: true
  validates :price, numericality: { greater_than_or_equal_to: 0 }, allow_nil:
true
  validates :area_sqm, numericality: { greater_than_or_equal_to: 0 },
allow_nil: true

  scope :available, -> { where(status: statuses[:available]) }

  def reserve!(attrs = {})
    transaction do
      update!(status: 'reserved', reserved_at: Time.current)
      reservations.create!(attrs)
    end
  end
end
```

---

3) Controllers (namespaced)

`app/controllers/admin_portal/projects_controller.rb`

```
module AdminPortal
  class ProjectsController < ApplicationController
    before_action :authenticate_admin!
    before_action :set_project, only: [:show, :edit, :update, :destroy]

    def index
      @projects = Project.order(:name).page(params[:page])
    end

    def show
      @stands = @project.stands.order(:stand_number).page(params[:page])
```

```
    end

    def new
      @project = Project.new
    end

    def create
      @project = Project.new(project_params)
      if @project.save
        redirect_to admin_portal_project_path(@project), notice: 'Project
created.'
      else
        render :new, status: :unprocessable_entity
      end
    end

    def edit; end

    def update
      if @project.update(project_params)
        redirect_to admin_portal_project_path(@project), notice: 'Project
updated.'
      else
        render :edit, status: :unprocessable_entity
      end
    end

    def destroy
      @project.destroy
      redirect_to admin_portal_projects_path, notice: 'Project removed.'
    end

    private

    def set_project
      @project = Project.find(params[:id])
    end

    def project_params

params.require(:project).permit(:name, :location, :description, :developer_name, :total_stands, :
    end
  end
end
```

app/controllers/admin_portal/stands_controller.rb

```
module AdminPortal
  class StandsController < ApplicationController
    before_action :authenticate_admin!
    before_action :set_project, only: [:new, :create]
    before_action :set_stand, only: [:show, :edit, :update, :destroy]

    def index
      @stands = Stand.includes(:project).order('projects.name,
stands.stand_number').references(:project).page(params[:page])
    end

    def show; end

    def new
      @stand = @project.stands.build
    end

    def create
      @stand = @project.stands.build(stand_params)
      if @stand.save
        redirect_to admin_portal_project_path(@project), notice: 'Stand
created.'
      else
        render :new, status: :unprocessable_entity
      end
    end

    def edit; end

    def update
      if @stand.update(stand_params)
        redirect_to admin_portal_stand_path(@stand), notice: 'Stand updated.'
      else
        render :edit, status: :unprocessable_entity
      end
    end

    def destroy
      project = @stand.project
      @stand.destroy
      redirect_to admin_portal_project_path(project), notice: 'Stand removed.'
    end

    private
```

```
    def set_project
      @project = Project.find(params[:project_id]) if params[:project_id]
    end

    def set_stand
      @stand = Stand.find(params[:id])
    end

    def stand_params

params.require(:stand).permit(:stand_number, :area_sqm, :price, :status, :latitude, :longitude, :
    end
  end
end
```

---

4) Views

All views are basic Bootstrap-powered ERB templates. Create the following files.

`app/views/admin_portal/projects/index.html.erb`

```
<div class="container mt-4">
  <h1>Projects</h1>
  <%= link_to 'New Project', new_admin_portal_project_path, class: 'btn btn-
primary mb-2' %>
  <table class="table table-striped">
    <thead>
      <tr><th>Name</th><th>Location</th><th>Stands</th><th></th></tr>
    </thead>
    <tbody>
      <% @projects.each do |p| %>
        <tr>
          <td><%= link_to p.name, admin_portal_project_path(p) %></td>
          <td><%= p.location %></td>
          <td><%= p.total_stands %></td>
          <td><%= link_to 'Edit', edit_admin_portal_project_path(p), class:
'btn btn-sm btn-outline-secondary' %></td>
        </tr>
      <% end %>
    </tbody>
  </table>
  <%= paginate @projects %>
</div>
```

app/views/admin_portal/projects/show.html.erb

```
<div class="container mt-4">
  <h1><%= @project.name %></h1>
  <p class="text-muted"><%= @project.location %> — <%= @project.developer_name
%></p>
  <p><%= @project.description %></p>

  <h3>Stands</h3>
  <%= link_to 'Add Stand', new_admin_portal_project_stand_path(@project),
class: 'btn btn-sm btn-primary mb-2' %>
  <div class="row">
    <% @project.stands.each do |s| %>
      <div class="col-md-3 mb-2">
        <div class="card">
          <div class="card-body">
            <h5><%= s.stand_number %></h5>
            <p><%= number_to_currency(s.price || 0) %> • <%= s.area_sqm %> sqm</
p>
            <span class="badge <%= case s.status when 'available' then 'bg-
success' when 'reserved' then 'bg-warning text-dark' else 'bg-danger' end
%>"><%= s.status %></span>
          </div>
          <div class="card-footer">
            <%= link_to 'View', admin_portal_stand_path(s), class: 'btn btn-sm
btn-outline-primary' %>
          </div>
        </div>
      </div>
    <% end %>
  </div>
</div>
```

app/views/admin_portal/stands/show.html.erb

```
<div class="container mt-4">
  <h2>Stand <%= @stand.stand_number %></h2>
  <p>Project: <%= link_to @stand.project.name,
admin_portal_project_path(@stand.project) %></p>
  <p>Price: <%= number_to_currency(@stand.price || 0) %></p>
  <p>Area: <%= @stand.area_sqm %> sqm</p>
  <p>Status: <%= @stand.status %></p>
</div>
```

app/views/admin_portal/stands/_form.html.erb

```
<%= form_with model: [(@stand.project if @stand.project), @stand].compact,
local: true do |f| %>
  <div class="mb-3">
    <%= f.label :stand_number %>
    <%= f.text_field :stand_number, class: 'form-control' %>
  </div>
  <div class="mb-3 row">
    <div class="col">
      <%= f.label :area_sqm %>
      <%= f.number_field :area_sqm, step: 0.01, class: 'form-control' %>
    </div>
    <div class="col">
      <%= f.label :price %>
      <%= f.number_field :price, step: 0.01, class: 'form-control' %>
    </div>
  </div>
  <div class="mb-3">
    <%= f.label :status %>
    <%= f.select :status, Stand.statuses.keys.map { |k| [k.humanize, k] }, {},
class: 'form-select' %>
  </div>
  <%= f.submit class: 'btn btn-primary' %>
<% end %>
```

app/views/admin_portal/stands/new.html.erb

```
<div class="container mt-4">
  <h2>New Stand for <%= @project.name %></h2>
  <%= render 'form', stand: @stand %>
</div>
```

app/views/admin_portal/stands/edit.html.erb

```
<div class="container mt-4">
  <h2>Edit Stand <%= @stand.stand_number %></h2>
  <%= render 'form', stand: @stand %>
</div>
```

---

5) Routes

Add nested routes for stands inside projects and standalone index/show for stands.

Update `config/routes.rb` with the following inside the `admin_portal` namespace:

```
namespace :admin_portal do
  resources :projects do
    resources :stands, only: [:new, :create]
  end
  resources :stands, only: [:index, :show, :edit, :update, :destroy]
end
```

---

6) Seed Data

Append to `db/seeds.rb` (or replace) to create projects and stands:

```
puts 'Seeding projects and stands...'

project1 = Project.find_or_create_by!(name: 'Sunridge Estate Phase 1') do |p|
  p.location = 'Harare - Highlands'
  p.developer_name = 'Sunridge Developers'
  p.total_stands = 50
  p.status = 'active'
end

project2 = Project.find_or_create_by!(name: 'Maple Gardens Phase 2') do |p|
  p.location = 'Harare - Borrowdale'
  p.developer_name = 'Maple Homes'
  p.total_stands = 30
  p.status = 'active'
end

(1..50).each do |i|
  Stand.find_or_create_by!(project: project1, stand_number: "S-#{i}") do |s|
    s.area_sqm = 400 + i
    s.price = 20000 + (i * 100)
    s.status = ['available', 'available', 'reserved', 'sold'].sample
  end
end

(1..30).each do |i|
  Stand.find_or_create_by!(project: project2, stand_number: "M-#{i}") do |s|
```

```
    s.area_sqm = 350 + i
    s.price = 18000 + (i * 90)
    s.status = ['available', 'reserved', 'sold'].sample
  end
end

puts 'Projects & stands seeded.'
```

---

7) Pagination & Helpers

I used `page(params[:page])` in controllers — install a pagination gem like `kaminari` or `pagy`.

Add to `Gemfile`:

```
gem 'kaminari'
```

Run `bundle install` and optionally generate views: `rails g kaminari:views bootstrap5`.

---

8) Quick Manual Tests

1. `bin/rails db:migrate` (if not yet ran)
2. `bin/rails db:seed`
3. Start server `bin/rails s`
4. Visit `/admin_portal/projects` (ensure you are logged in as an admin user seeded earlier)

---

9) Next Steps (after Phase 2)

- Build agent and client sitemap views using project.stands coordinates (map or grid)
- Add Stand search & filters in Agent portal
- Add concurrency-safe reservation logic (DB locks or optimistic locking)
- Prepare CoreSystemSyncService integration points for importing projects/stands

---

If you want, I can now: - Add the **agent sitemap** view + Stimulus controller for interactive filtering, or - Implement **concurrency-safe reservation** logic (db-level locking plus tests), or - Add **kaminari** pagination views into the canvas.

Tell me which of the above to do next and I will add it to the canvas.