Slugs

ah yes — the “ensure slugs” bit. here’s a tight recap + copy-paste helpers so you can lock this down fast.

**1) Model concern (already in your bundle)**

# app/models/concerns/sluggable.rb

module Sluggable

extend ActiveSupport::Concern

included do

before\_validation :ensure\_slug

validates :slug, presence: true, uniqueness: true

def to\_param = slug

end

private

def ensure\_slug

return if slug.present?

base = respond\_to?(:name) ? name : (respond\_to?(:title) ? title : nil)

base = full\_name if base.blank? && respond\_to?(:full\_name)

self.slug = base.to\_s.parameterize if base

end

end

**2) Add to each public model**

class Soldier < ApplicationRecord

include Sluggable

# (optionally include SoldierSlug for better candidates)

end

# Same for Battle, War, Medal, Category, Source, Article, etc.

**3) DB constraints (idempotent migration)**

class AddSlugsEverywhere < ActiveRecord::Migration[7.1]

TABLES = %i[soldiers battles wars medals categories sources]

def change

TABLES.each do |t|

add\_column t, :slug, :string unless column\_exists?(t, :slug)

add\_index t, :slug, unique: true unless index\_exists?(t, :slug, unique: true)

end

end

end

**4) Backfill in bulk (already in your rake; quick one-off too)**

# one-liner to force slug generation (uses before\_validation)

[Soldier, Battle, War, Medal, Category, Source].each do |k|

k.find\_each { |r| r.slug = nil if r.slug.blank?; r.save!(validate: false) }

end

Or run the provided task:

bin/rails data:backfill\_slugs

**5) Handle dupes/collisions (find & fix)**

# find duplicate slugs on a model

dupes = Soldier.group(:slug).having("COUNT(\*) > 1").count

puts dupes

# resolve by regenerating for the later records

dupes.keys.each do |slug|

Soldier.where(slug: slug).order(:id).offset(1).find\_each do |rec|

rec.slug = "#{slug}-#{rec.id}" # or custom logic (year, unit, etc.)

rec.save!(validate: false)

end

end

If you prefer smarter soldier slugs (e.g., “john-doe-1880”):

# app/models/concerns/soldier\_slug.rb

module SoldierSlug

extend ActiveSupport::Concern

included { before\_validation :build\_soldier\_slug }

private

def build\_soldier\_slug

return if slug.present?

yr = birthday&.year || try(:birth\_year)

base = [first\_name, last\_name, yr].compact.join(" ")

self.slug = base.parameterize.presence || SecureRandom.hex(4)

end

end

…and include it:

class Soldier < ApplicationRecord

include Sluggable

include SoldierSlug

end

**6) Use slugs in controllers**

def show

@soldier = Soldier.find\_by!(slug: params[:id])

end

**7) Quick sanity script**

[Soldier, Battle, War, Medal, Category, Source].each do |k|

puts [k.name, k.where(slug: nil).count, k.group(:slug).having("COUNT(\*)>1").count.size].inspect

end

# => ["Soldier", 0, 0] means good: no nils, no dupes

**8) Optional: FriendlyId (only if you want more)**

If later you want automatic collision handling, history slugs, etc., you can layer in friendly\_id. For now, your lightweight concern + unique index is perfectly fine.