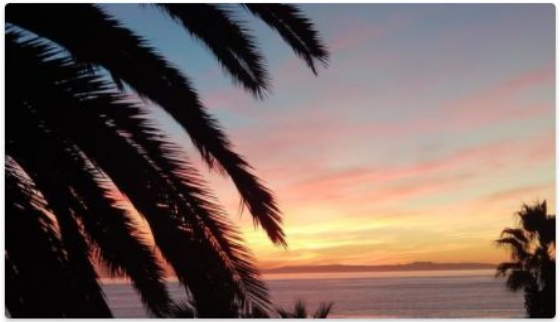
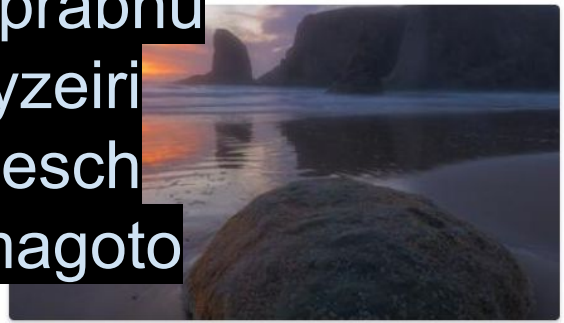






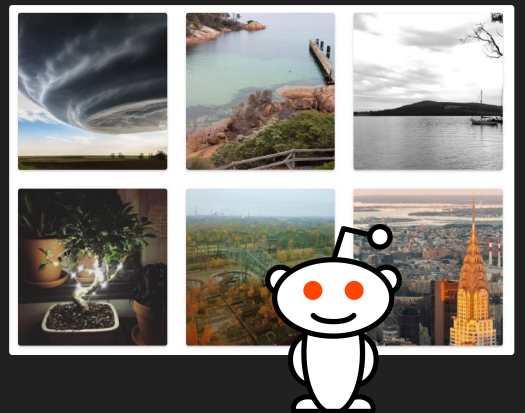
gud wallpapers

team git gud:

stephen wu
nishad prabhu
martin yzeiri
ben janesch
jason magoto

Objective: wallpaper aggregation & sharing service

- source: reddit subs: /r/wallpapers, etc.
- allow for users to favorite, upload, & tag wallpapers
- allow for moderators to manage & delete wallpapers
- allow for search by user-submitted tags, resolutions, etc.
- allow for sorting by views, date created, popularity



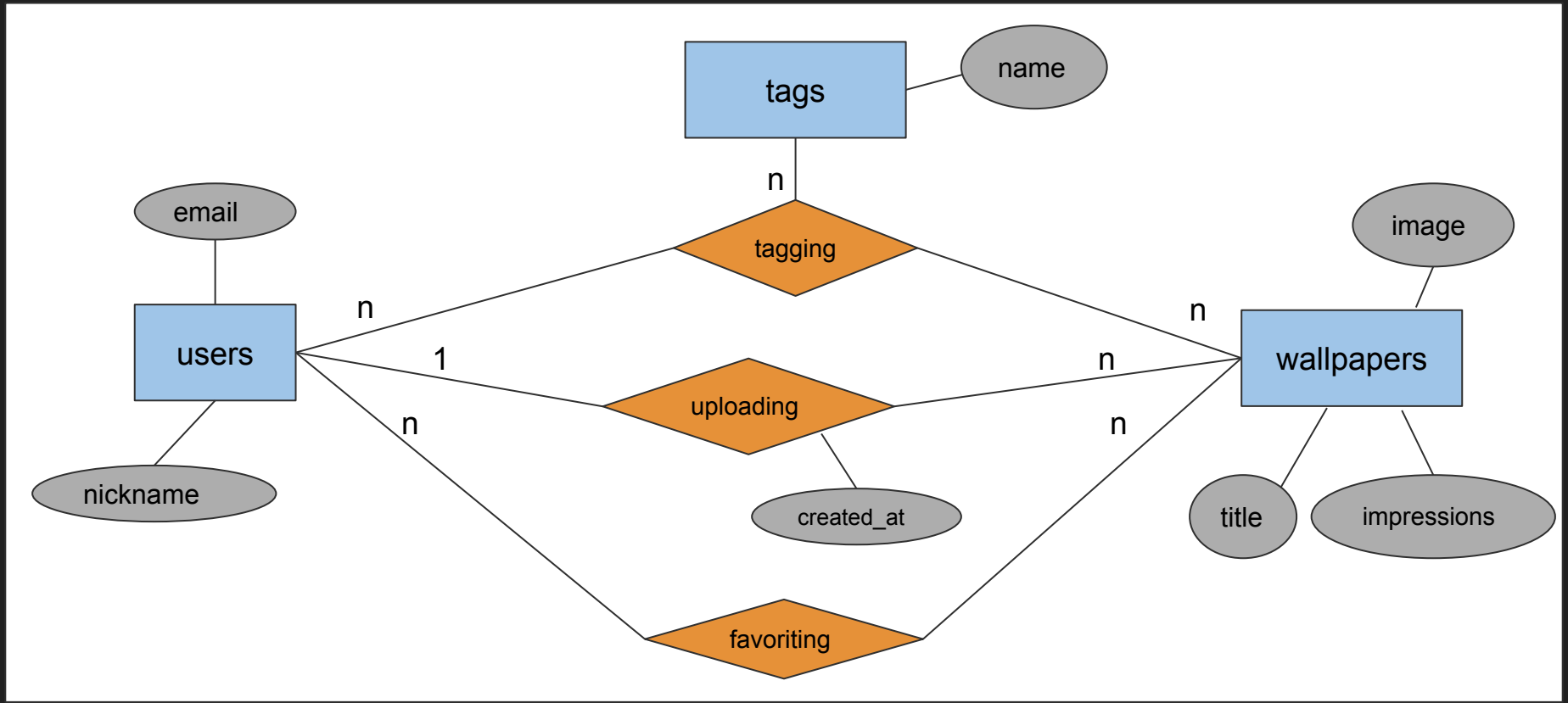
Demo

Stephen

- Redd to scrape images from reddit subs
  - AWS S3 for file storage
  - Paperclip for uploading

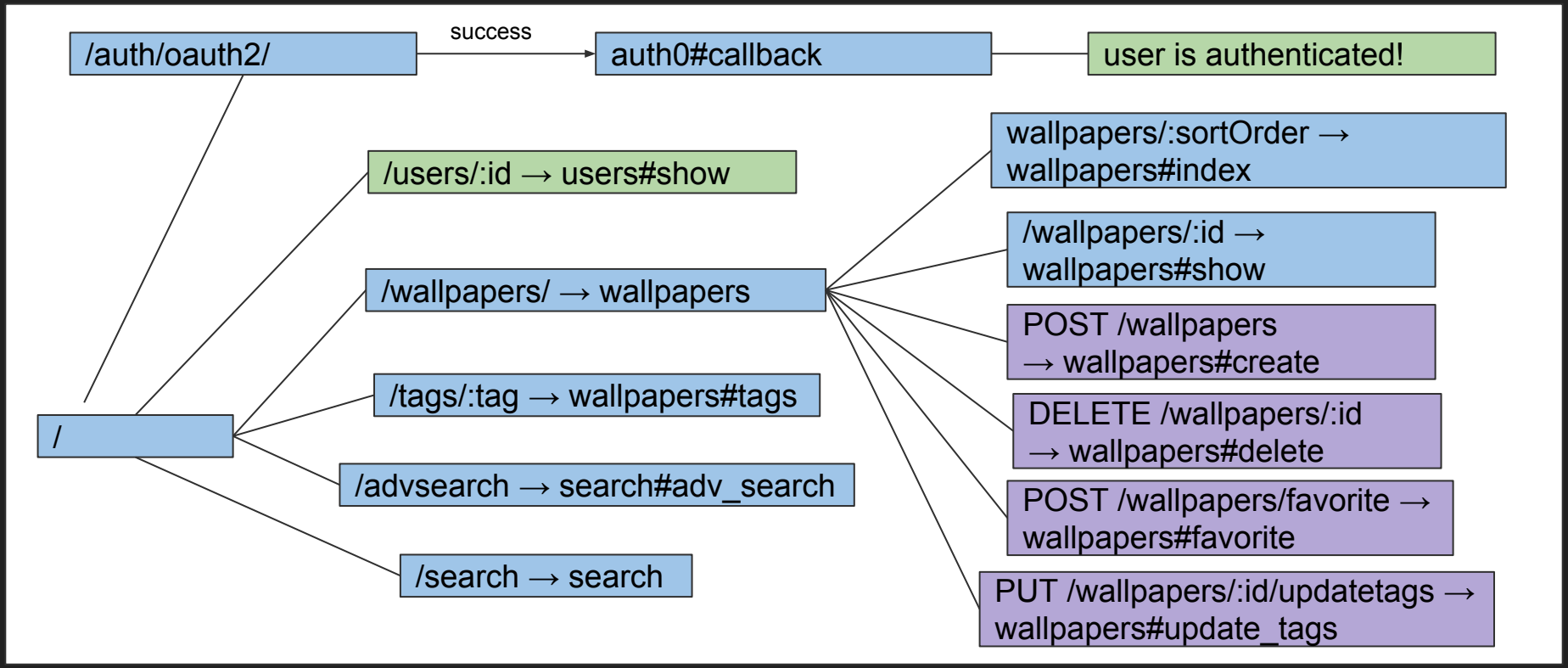
  SQLite for relational db
- Ruby on Rails as full-stack MVC
- OmniAuth/Auth0 as authentication
- Bootstrap+Sass+jQuery for front-end

Tech Stack



ER Diagram (Abridged)

routes


/auth/oauth2/
success
auth0#callback
user is authenticated!
/users/:id → users#show
/wallpapers/ → wallpapers
/tags/:tag → wallpapers#tags
/advsearch → search#adv_search
/search → search
/
wallpapers/:sortOrder → wallpapers#index
/wallpapers/:id → wallpapers#show
POST /wallpapers → wallpapers#create
DELETE /wallpapers/:id → wallpapers#delete
POST /wallpapers/favorite → wallpapers#favorite
PUT /wallpapers/:id/updatetags → wallpapers#update_tags

config

- secrets.yml used for storing S3, reddit, and auth0 credentials
- e.g.
  “Rails.application.secrets.aws_bucket”

```
# Shared variables across all 3 environments
shared: &shared
  # AWS file storage details
  aws_bucket: gudwallpapers
  aws_access_key_id: [redacted]
  aws_secret_access_key: [redacted]
  aws_s3_region: us-east-2
  aws_s3_host_alias: d2cvlttosszbwx.cloudfront.net
  # Auth0 authentication details
  auth0_client_domain: gudwallpapers.auth0.com
  auth0_client_id: [redacted]
  auth0_client_secret: [redacted]
  # Reddit scraper agent details
  reddit_user_agent: Redd:GudWallpapers:v1.0 (by /u/GudWallpapers)
  reddit_client_id: [redacted]
  reddit_client_secret: [redacted]
  reddit_username: GudWallpapers
  reddit_password: [redacted]

development:
  <<: *shared
  secret_key_base: [redacted]

test:
  <<: *shared
  secret_key_base: [redacted]

# Do not keep production secrets in the repository,
# instead read values from the environment.
production:
  <<: *shared
  secret_key_base: <%= ENV["SECRET`_KEY_BASE"] %>
```

Ben

seeds.rb

- Seeds the database with images from Reddit
- One source of images, other is users
- Core components:
  a. Authentication with Reddit using our account
  b. Begin the session
  c. Iterate through top posts on subscribed subreddits
  d. Get image title
  e. Make sure link is to an image
  f. Save image to AWS server using Paperclip

seeds.rb output

```
[Scraping from subscribed subreddits]

1. Blue hour in Cochem [2048x1186]

https://imgur.com/wGz77Vt - added from wallpapers

2. Iron Harvest 1920 × 1080

https://i.redd.it/7xdcdnztx4101.png - added from wallpapers

3. Natural trees in line

https://imgur.com/uwJLQKP - added from wallpapers

4. Galaxy themed wallpaper

https://i.redd.it/gns36z7mh2101.jpg - added from wallpapers

5. A night under the stars

https://i.redd.it/6lcn109534101.jpg - added from wallpapers
```

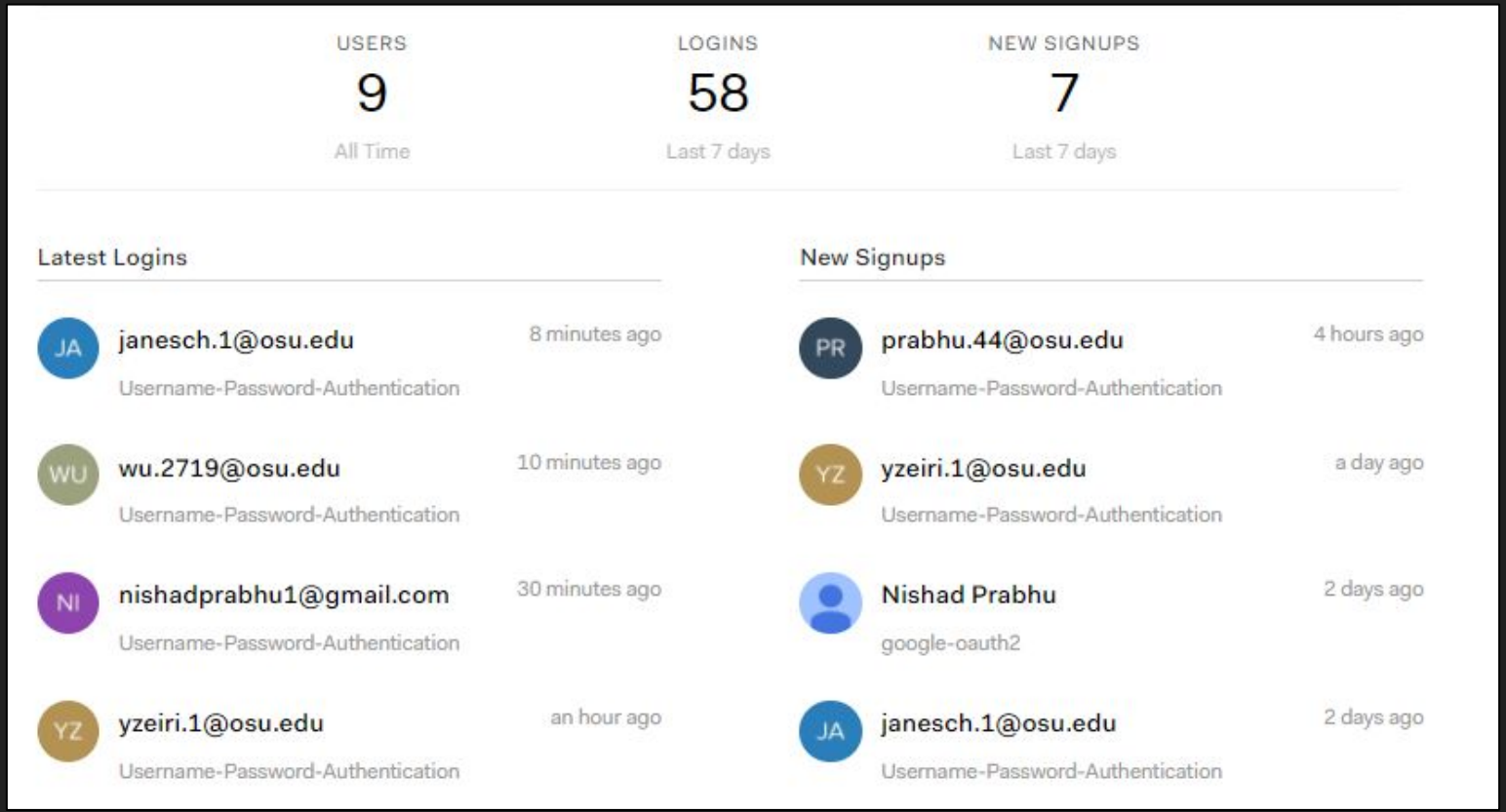
seeds.rb output

```
45. Castle Square, Warsaw [1334x750] [OC]

https://i.redd.it/xqc4ngea11101.jpg - added from CityPorn

46. Bonsai Golden Gate Ficus [OC]

https://i.redd.it/a7v4sxnroz001.jpg - added from BotanicalPorn

47. Purple Hyacinth with texture at Allerton Park, Illinois [OC] 5456 x 3632

https://imgur.com/D82Wx13 - added from BotanicalPorn

48. Orange Aloe Plant [6000x4000]
https://www.flickr.com/photos/152769751@N02/37444209501/in/dateposted-public/ -
error: #<OpenURI::HTTPError: 404 Not Found>

49. Rock Columbine on the Ridge, Bryce Canyon National Park, Utah [OC] [5456 x 3
632]

https://imgur.com/3q0RPA3 - added from BotanicalPorn
```

User (Model)

provider	uid	name	nickname	email	created_at	updated_at	user_rank
string	string	string	string	string	datetime	datetime	integer

- Authentication done through OmniAuth & Auth0
  - We never store passwords
  - Provider: Auth0 (required)
  - Uid: OAuth identifier (required)
- user_rank: 1 for normal, 2 for moderator



Auth0 Dashboard

User (Model)

```ruby
# Association for favorites
has_many :favorite_wallpapers, :through => :favorites, :source => :wallpaper
has_many :favorites

# Association for uploads
has_many :uploads, :class_name => "Wallpaper", :foreign_key => "uploader_id"
# User can tag photos
acts_as_tagger
```

- Has many favorites, uploads, and tags
  - Favorites and uploads are both associated with Wallpapers

User (Authentication)

- Called when OmniAuth succeeds

```
def callback
  # Check if we recieved a hash variable from omniauth. If not, redirect to index.
  if request.env['omniauth.auth']
    # Use the omniauth values to either find or create a user
    @user = User.find_or_create_from_auth(request.env['omniauth.auth'])
    if @user
      set_current_user @user
      redirect_to user_path(@user)
    else
      redirect_to root_path
    end
  else
    redirect_to root_path
  end
end
```

User (Authentication)

```
def self.find_or_create_from_auth(auth)
  user = User.find_or_create_by(provider: auth['provider'], uid: auth['uid'])
  user.nickname = auth['info']['nickname']
  user.email = auth['info']['name']
  user.user_rank = 1; # Default rank is 1
  user.save
  user
end
```

- User is tied to their uid

Authentication Helpers

- Current_user: whoever is logged in currently

```ruby
def current_user
   User.find_by(uid: session[:user_id]) if session[:user_id]
end
```

```ruby
def set_current_user(user)
  @current_user = user
  session[:user_id] = user.nil? ? nil : user.uid
end
```

```ruby
def signed_in?
  !!current_user
end
```

More Authentication Helpers

```
def authorize!
  respond_to do |format|
    format.html {redirect_to "/auth/auth0" unless current_user}
    format.js { (render :js => "window.location = '/auth/auth0'") unless current_user}
  end
end
```

In wallpapers_controller...

```
before_filter :authorize!, :except => [:show, :tags, :index]
```

User (Profile Page)

- GET /user/id
- Can only see your own profile
- Lists both favorites and uploads

```
def show
  @user = User.find_by_id(params[:id])
  #Make sure the user passed in through the parameters matches t
  if !@user.nil? && !current_user.nil? && @user == current_user
    #Find out which gallery should be displayed
    if params[:gallery] == 'favorites'
      @wallpapers = @user.favorite_wallpapers
    #Default should be the uploaded gallery
    else
      @wallpapers = @user.uploads
    end
  #Redirect to index if not
  else
    redirect_to :controller=>"wallpapers", :action =>"index"
  end
end
```

Wallpaper (Model)

title	created_at	updated_at	impressions _count	priority	uploader_id	image...
string	datetime	datetime	integer	float	integer	many

```
# Association for favorites
has_many :favorited_by, :through => :favorites, :source => :user
has_many :favorites
# Association for uploads
belongs_to :uploader, :class_name => "User", :foreign_key => "uploader_id"
# Allow wallpapers to be tagged
acts_as_taggable
# Allow views to be tracked on wallpapers
is_impressionable :counter_cache => true
```

Image Uploading

```
# Wallpapers have images attached to them
has_attached_file :image, styles: {
  index: "600x600"
},
# Path on amazon webserver
:path => "images/:class/:style/:id:title.:extension"
after_post_process :save_image_dimensions
validates_attachment :image, content_type: { content_type: ["image/jpg", "image/jpeg", "image/png", "image/gif"] }
validates_attachment_presence :image
```

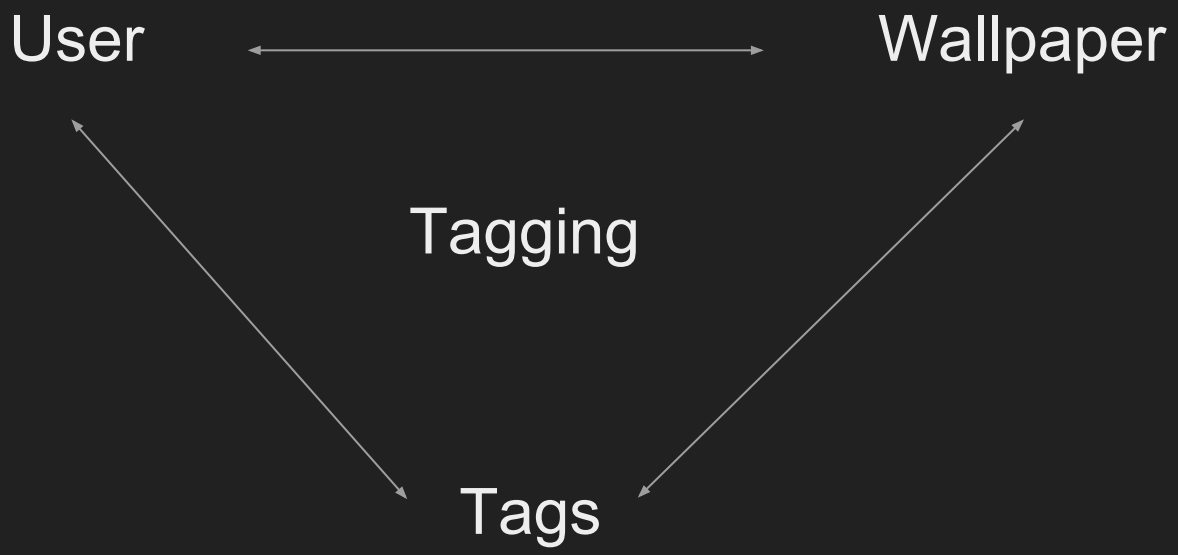
- Paperclip gem automatically sends files to Amazon S3
- Stores image attributes in the wallpapers table

Image Uploading

```ruby
def wallpaper_params
  params.require(:wallpaper).permit(:title, :image, :tag_list)
end
```

```ruby
def create
  @wallpaper = Wallpaper.new(wallpaper_params)
  @wallpaper.priority = @wallpaper.get_priority
  @wallpaper.uploader = current_user
  @wallpaper.set_owner_tag_list_on(current_user, :tags, @wallpaper.tag_list)
  @wallpaper.tag_list.clear
  respond_to do |format|
    if @wallpaper.save
      format.html { redirect_to @wallpaper, notice: 'Wallpaper was successfully created.' }
      format.js { redirect_to @wallpaper}
      format.json { render :show, status: :created, location: @wallpaper }
```

Wallpapers (Tagging)



Wallpapers (Tagging)

- wallpapers_controller actions

GET /tags/:tag

```ruby
def tags
  @wallpapers = Wallpaper.tagged_with(params[:tag])
  @wallpapers = @wallpapers.order(priority: :desc)
  @results_count = @wallpapers.count
  @wallpapers = @wallpapers.page params[:page]
  render :index
end
```

PATCH/PUT /wallpapers/1/updatetags

```ruby
def update_tags
  respond_to do |format|
    current_user.tag(@wallpaper, :with => params[:taglist], :on => :tags)
    if @wallpaper.save
      # Render JS to dynamically updae tags
      format.js { render 'partials/update_tags', status: :ok}
    else
      puts @wallpaper.errors.inspect
      format.js { render json: @wallpaper.errors, status: :unprocessable_entity}
    end
  end
end
```

Favorites

- A way to track images you like
- Show up in your gallery
- Unfavorite removes from gallery
- Total favorites tracked per image

Wallpaper Controller

```
def favorite
  @user = current_user
  @wallpaper = Wallpaper.find(params[:wallpaper_id])
  @user.favorite!(@wallpaper)
  respond_to do |format|
    format.js { render 'favorite'}
  end
end
```

Favorites Model

```
class Favorite < ActiveRecord::Base
  belongs_to :user
  belongs_to :wallpaper
end
```

```
def unfavorite
  @user = current_user
  @favorite = @user.favorites.find_by_wallpaper_id(params[:wallpaper_id])
  @wallpaper = Wallpaper.find(params[:wallpaper_id])
  @favorite.destroy!
  respond_to do |format|
    format.js { render 'unfavorite'}
  end
end
```

View Counts - Tracking with Impressionist

Wallpaper Controller

```
impressionist :actions=>[:show]
```

```
def show
  @user = current_user
  @wallpaper = Wallpaper.find(params[:id])
  @view_count = @wallpaper.impressionist_count
  @favorites = @wallpaper.favorites
  @favorited_by = @wallpaper.favorited_by
  @wallpaper.update_column(:priority, @wallpaper.get_priority)
end
```

Wallpaper model

```
# Allow views to be tracked on wallpapers
is_impressionable :counter_cache => true
```

Index - Sorting

- Displayed wallpapers can be sorted by Top, Popularity, or Date Created.
- Top corresponds to views
- Popular uses Reddit's algorithm to rank Wallpapers based on views + frequency.

```
else
  #If not searching, then choose sorting order
  @wallpapers = Wallpaper.all
  # Get nishad's picks
  @nishad = @wallpapers.sample 4

  @sortOrder = params[:sortOrder]
  if @sortOrder == 'latest'
    @wallpapers = @wallpapers.order(created_at: :desc)
  elsif @sortOrder == 'top'
    @wallpapers = @wallpapers.order(impressions_count: :desc)
  else
    # Default is priority, which factors in view count as well as time created
    @wallpapers = @wallpapers.order(priority: :desc)
  end
  # Paginate the results
  @wallpapers = @wallpapers.page params[:page]
end
respond_to do |format|
  format.html
  # Render javascript to add more wallpapers via the load more button
  format.js { render 'partials/wallpaper_page'}
end
```

Index - Search

- Basic search included in the navbar.
- Looks for wallpapers with keywords in their title or their tag list.
- Logic contained in wallpaper controller.

```
def index

  # Check if we are searching something
  if params[:search]
    @wallpapers = Wallpaper.search(params[:search])
    @results_count = @wallpapers.count
    # We can't call page on an array, which is returned from the tag search method, so special check here
    if @wallpapers.class == Array
      @wallpapers = Kaminari.paginate_array(@wallpapers).page(params[:page]).per(28)
    else
      @wallpapers = @wallpapers.page params[:page]
    end
  else
    #If not searching, then choose sorting order
```

Index - Advanced Search

- Filters based on keywords in title, tag list.
- Filters based on resolution.
- SearchController calls the advanced search method in Wallpaper model.

```ruby
def self.adv_search(title, tags = nil, image_height = nil, image_width = nil)
  results = Wallpaper.all

  if !title.nil? && !title.empty?
    results = results.where("title LIKE ?", "%#{title}%")
  end
  if !tags.nil? && !tags.empty?
    results = results.tagged_with(tags, any:true)
  end
  if !image_height.nil? && !image_height.empty?
    results = results.where(image_height: image_height)
  end
  if !image_width.nil? && !image_width.empty?
    results = results.where(image_width: image_width)
  end
  results
end
```

Index - Advanced Search

Search Controller Code

```
def adv_search
  @wallpapers = Wallpaper.adv_search(params[:title], params[:tags], params[:image_height], params[:image_width])
  # The tag search function returns an array instead of an active record collection
  if @wallpapers.class == Array
    @wallpapers = Kaminari.paginate_array(@wallpapers).page(params[:page]).per(28)
  else
    @wallpapers = @wallpapers.page params[:page]
  end
  # Boolean that is checked when needing to display certain elements.
  @search = true
  render 'wallpapers/index'
end
```

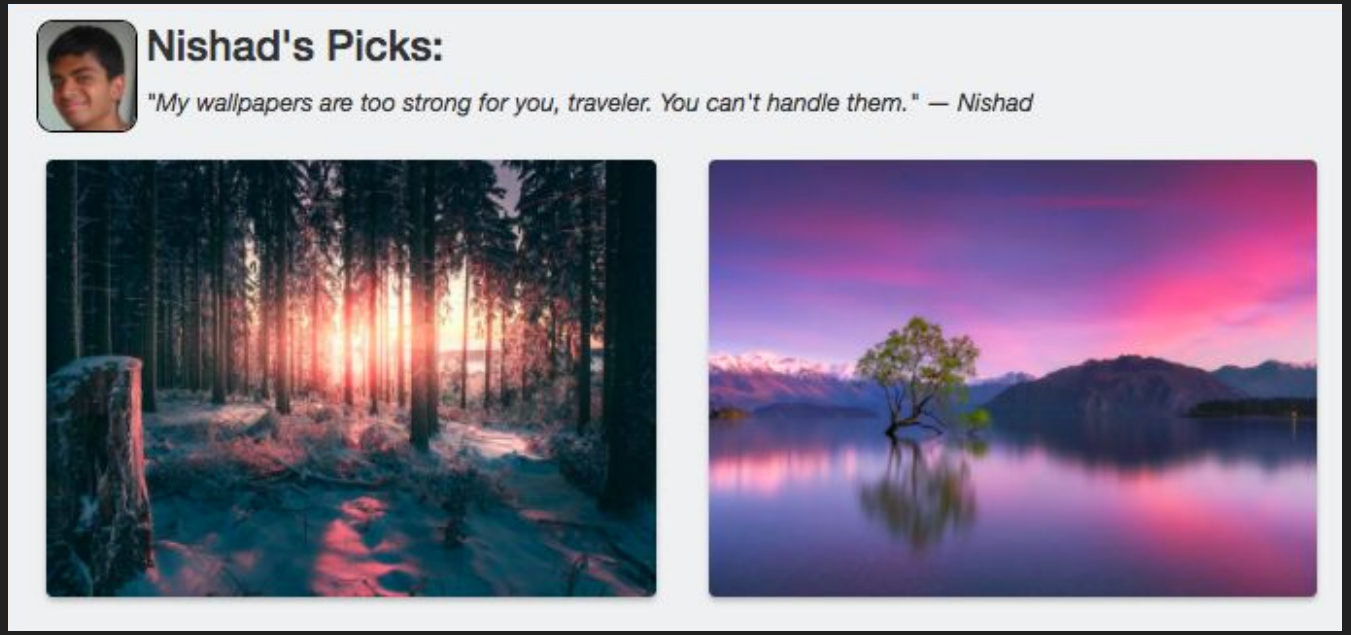
Displaying Wallpapers

- Partial used search, sorting, user gallery, etc.

```
<% @wallpapers.each_slice(4) do |slice| %>
  <div class="image-row">
    <% slice.each do |wallpaper| %>
      <div class="col-lg-auto col-md-3 col-sm-6 col-xs-12">
        <a class="wallpaper img-thumbnail img-responsive" href="<%=
        wallpaper_path(wallpaper) %>" style="width:100%; background-image:
        url(<%=wallpaper.image.url(:index) %>)"></a>
      </div>
    <% end %>
  </div>
<% end %>
```

Nishad’s Picks

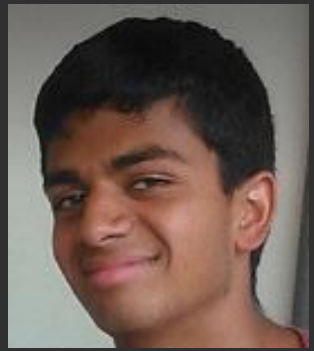
Displays four featured wallpapers and a quote from Nishad



Thanks for listening!

Team: git gud

Questions?



go.osu.edu/gudwallpapers