Models:

**Album.rb**

The album model is how we define the three albums which will hold album items for each of the three businesses. Each album belongs to a topic, which determines which of the three businesses the album is for.

Campground album is topic 1, Estate Park album is topic 2, and the mobile home park is topic 3. When an administrator uploads an image, they select which album (topic) the image will be categorized in.

When uploading an image, the album model validates that a title and topic ID are present.

To handle the image upload, we’ve used a gem called ‘Carrierwave’ which is configured with another gem called ‘fog’ to host the images in an Amazon S3 bucket. The S3 bucket only gets used in the production environment. When running in the development environment, the images are stored in the public/uploads directory.

**Topic.rb**

The topic model simply has many albums.

**Post.rb**

The post model defines how blog posts are stored. It belongs to the category model, which determines which of the three businesses the blog post is about. When uploading a blog post, the title, content, and category ID must be present. We also used the same image uploader (carrierwave) for the posts as we did for the album images.

**Category.rb**

The category simply has many posts.

**Message.rb**

The message model is how we define the message form on the contact page. To make this work, we’ve used a gem called ‘mail\_form’. The message class extends MailForm::Base, and allows email contents and email headers to be defined. Message model will verify that a valid name, email, and message is present in the form. Regular expressions verify that a valid email is entered. Our headers define the subject, the email address we want the message delivered to, and the name and email that it’s from, based on the name and email entered in the form.

**User.rb**

The User model is the largest model in the application, and is used to define User information entered when registering an account. This verifies that the name is under 50 characters, the email is valid (using regular expressions), and the password is greater than 6 characters. To handle passwords, we’ve used a gem called ‘bcrypt’, which hashes passwords and stores them safely.

*Functions is the User model:*

User digest: returns the hash digest of the given string

User new token: returns a random token

Remember: remembers a user in the database for use in persistent sessions

Authenticated?: returns true if the given token matches the digest

Forget: forgets a user

Activate: activates an account

Send\_activation\_email: sends an activation email

create\_reset\_digest: sets the password reset attributes

send\_password\_reset\_email: sends an email to reset a password

password\_reset\_expired?: returns true if a password reset has expired (<2hrs)

downcase\_email: converts email to all lower-case

create\_activation\_digest: create an assign the activation token and digest