Controllers:

**Account\_activations.rb:**

Controller to handle when a user clicks on the ‘activate’ link in the activation email. It will find the user by email, confirm is not already activated, confirms the users session is authenticated, and will then activate. Upon successful activation, the user will be redirected to the user’s account page. If activation is unsuccessful, the user will be redirected to the home landing screen.

**Albums\_controller:**

Controller to all album actions. Functions include:

index: will order all album\_items (images) by position

campground: gather all the images with the campground topic id

river\_run: gather all images with the river run topic id

estate\_park: gather all images with the vllage topid id

new: initialize a new album\_item

create: create the new album\_item and store in the database with the required parameters

edit: find the album\_item based on the album\_item ID

update: after finding the album item based on the album ID, update the parameters based on the input in the edit form

show: find the album item and display the show view

destroy: find the album item and destroy it from the database

post\_params: require the correct parameters

find\_album\_item: find ablum item by ID

admin\_user: only allow admins to access the new, create, update, destroy, and edit functions

**Application\_controller:**

Includes SessionHelper, protects from forgery, and does ‘before\_actions’ before any other action. Function:

detect\_device\_format: requests user agent (client browser information) and uses a rails request variant to set specific views for only on mobile. The ‘About’ page is displayed differently on desktop than it is on mobile.

**Messages\_controller:**

Handles the messages sent from the Contact Us form. Functions:

new: initialize a new messages

create: build a new message with the parameters given in the form. This function uses AJAX to deliver the message without refreshing the page. When the email is sent successfully, the create.js.erb file in the views/messages directory will render a flash on the page and confirming its success, all without refreshing the entire page.

**Password\_resets\_controller:**

Handles when a user needs to reset their password. Before\_actions include: get\_user, valid\_user, and check\_expiration. These functions run before anything else. Functions include:

create: find the User by email, if successful, create a reset digest and send an email to the user.

Update: attempt to update user password attribute, if successful, log the user in and redirect to user account page

user\_params: verifies that password and password confirm parameters are present

get\_user: before action that finds the user in the database by email

valid\_user: before action that verifies the user is logged in and activated

check\_expiration: verify the password reset email is younger than 2 hours old

**Posts\_controller:**

Handles the blog post actions. Before\_include: find\_post, and admin\_user. Functions include:

index: order and display all posts in the database by time created (newest first). If the user clicks one of the category filters, only posts regarding that category will be displayed.

new: initialize a new Post object

create: attempt to create the Post given the form parameters and save it to the database. Redirect to the post itself after successful save.

Update: Will attempt to update the post’s content based on the parameters entered in the edit post view.

Destroy: will remove the post from the database.

post\_params: verifies the post contains a valid title, content, category ID, and image.

find\_post: before action that will retrieve the post in the database based on the post ID

admin\_user: before action that only allows an admin to use the new, create, update, destroy, and edit functions.

**Sessions\_controller:**

Controller to handle user sessions when logging in on the website. Functions include:

create: will find the User in the database by email, authenticate given the session and password parameters, and check to see if the user is activated in the system. It will check if the ‘remember me’ checkbox is checked, and attempt to log the user in. If the user is not activated, a warning will be flashed and the user will be redirected to the root landing page.

Destroy: will destroy the user session by logging the user out if they are logged in. Redirected to root landing page if logout is successful.

**Static\_pages\_controller:**

Controller to simply route url paths to the various static view pages. Functions include:

root: ‘home’ is the root, so this routes to the home landing page view

home: routes to the home landing page view

help: routes to the help page view

about: will determine whether the user is on a desktop or mobile device based on the helper function in the Application\_controller (detect\_device\_format) and display the correct ‘About’ page. The desktop about page uses a javascript scrolling paralax effect, which does not render well on small screens.

Contact: will route to the contact view page

campground: will route to the campground view page

mobilehome: will route to the mobile home park view page

estatepark: will route to the estate park view page

**Users\_controller:**

Handles all the User operations. Before\_actions include: set\_user, logged\_in\_user, correct\_user, and admin\_user. This controller also uses the ‘DashboardLayout’ view in the layouts directory. This layout is different from the regular application layout that the rest of the site uses. Functions include:

index: display all the users and paginate them on the screen

show: find the User by ID and display the account page

new: initialize a new User object

edit: get the User ID and show the edit user page

create: Attempt to create and save the User to the database with the inputed user parameters. If the user is saved to the DB successfully, send their email an activation email.

Update: updates the user information in the database if the parameters are valid

destroy: find the user based on ID and remove from the DB

user\_params: allow only whitelist parameters

set\_user: before\_action to set the current user based on ID

logged\_in\_user: before\_action to check if the user is currently logged in

correct\_user: before\_action to confirm the correct user

admin\_user: before\_action to confirm an admin user. Only admins may destroy a user.