Adding Users, Authentication and Limiting Access

Adding Users

Create a User model:

```
rails generate model User name:string password:digest
rails db:migrate
```

This creates a users table without any controller and views. Users will be added through the console.

Edit User model for validation:

```
class User < ApplicationRecord
  validates :name, presence: true, uniqueness: true
  has_secure_password
end
```

Before you test your user model in the console, uncomment the gem bcrypt statement in the file Gemfile in your main application folder. Then run bundle install at the command line.

Example code for adding a user in the console:

```
u = User.new
u.name = “craig”
u.password = “secret”
u.save
```

To test authenticate method:

```
u.authenticate(“secret”)
u.authenticate(“guess”)
```

Creating a session control for login

Generate sessions controller with views:

```
rails generate controller Sessions new create destroy
```

The new.html.erb file needs a form. It also includes a possible message to handle returns to it for incorrect login:

```
<% if flash[:alert] %>
  <p><%= flash[:alert] %></p>
<% end %>

<%= form_tag controller: "sessions", action: "create" do %>
  <h2>Please log in</h2>

  <div class="field">
    <%= label_tag :name, "Name:" %>
    <%= text_field_tag :name, params[:name] %>
  </div>

  <div class="field">
    <%= label_tag :password, "Password:" %>
    <%= password_field_tag :password, params[:password] %>
  </div>

  <div class="field">
    <%= submit_tag "Login" %>
  </div>
<% end %>
```

Modify routes.rb (in config folder) so that create takes a post request:

```
post 'sessions/create'
```

Add code to handle controller actions:

```
class SessionsController < ApplicationController
  def new
  end

  def create
    user = User.find_by_name(params[:name])
    if user and user.authenticate(params[:password])
      session[:user_id] = user.id
      redirect_to airports_url
    else
      redirect_to "/sessions/new", alert: "Invalid user/password combination"
    end
  end

  def destroy
    session[:user_id] = nil
    redirect_to airports_url, notice: "Logged out"
  end
end
```

Limiting Access

Adding this statement and method definition in the application_controller.rb file requires that the user logs in before any controller action is executed.

```
class ApplicationController < ActionController::Base
  before_action :authorize
  protect_from_forgery with: :exception

  protected
    def authorize
      if not User.find_by_id(session[:user_id])
        redirect_to "/sessions/new", notice: "Please log in"
      end
    end
end
```

The authorize method is applied to all controller actions, which causes a problem for the sessions controller. You shouldn't be required to be logged in to go to the login form! Add this statement to the sessions controller:

```
class SessionsController < ApplicationController
  skip_before_action :authorize
```

You can also allow "whitelist" access to specific controller actions. For example, this addition to the airports controller allows access to index and show without being logged in:

```
class AirportsController < ApplicationController
  skip_before_action :authorize, only: [:show, :index]
```