Requirements

- Add functionality to a Ruby on Rails application to allow users to sign in using at least two external providers, such as Twitter and Mozilla Persona.

- Add new sign in links to your application that, when clicked, redirect the user to an external provider’s sign in page. Upon successful authentication with the provider, the user should be redirected back to your application and see a success message. At this point, the user should be signed in and able to use the application in the same capacity as any other signed-in user (including the ability to sign out).

- If the user signs in using an external provider, and has never before signed in using that provider, your application should automatically create a new user account for the user (without requiring any additional action or input from the user). However, if the user has previously signed in using that provider, your application should establish a new user session using the existing user account.

- Use the open-source library OmniAuth to implement the above functionality.

Steps

1. OmniAuth supports many different authentication providers (the complete list is here). You may choose any two providers for this assignment, but be aware that some providers introduce more complexity than others. Twitter is a safe choice for your first provider, and is used in the steps below. You’ll need a Twitter account, so create one if you don’t already have one.

2. Once you have a Twitter account, go here and create a new application with Twitter. Specify anything you like for the Name, Description, and Website. For the Callback URL, specify http://YOUR-APP.herokuapp.com/auth/twitter/callback (replace YOUR-APP with the name of your Heroku app). Once you create the application, Twitter provides you with a Consumer Key and Consumer Secret. You’ll need these in a minute.

3. Start by adding the Twitter OmniAuth strategy to your project. Add `gem 'omniauth-twitter'` to your Gemfile and run `bundle` in Terminal to install the gem.

4. Configure OmniAuth by creating a new initializer at `config/initializers/omniauth.rb`. Add the following code and insert your Twitter Consumer Key and Consumer Secret.

   ```ruby
   Rails.application.config.middleware.use OmniAuth::Builder do
     provider :twitter, "YOUR_CONSUMER_KEY", "YOUR_CONSUMER_SECRET"
   end
   ```

5. Add a new route to `config/routes.rb` for the OmniAuth callback URL. The route maps to the `create` action in `SessionsController`, which you’ll soon modify to work with OmniAuth.

   ```ruby
   get '/auth/:provider/callback', :to => 'sessions#create'
   ```

6. The following Rails generate command creates a database migration to add string attributes `provider` and `provider_uid` to the User model. These attributes are required to associate a user to an external provider. Generate the migration and apply it.

   ```bash
   $ bin/rails generate migration add_provider_and_provider_uid_to_users provider:string provider_uid:string
   $ bin/rake db:migrate
   ```

7. Open your `SessionsController` and modify the `create` action to look something like the code below. If the request contains authentication data from OmniAuth, the controller will send the data to a new class method on the User model that returns a User instance (you’ll create this method next). The controller will then proceed to create a session for that user. Otherwise, the controller will attempt to validate the email and password provided by the sign in form, just as it did before.
def create
  if omniauth = request.env["omniauth.auth"]
    user = User.find_or_create_with_omniauth(omniauth)
  elsif params[:email]
    user = User.find_by_email(params[:email])
    user = user && user.authenticate(params[:password])
  end

  if user
    session[:user_id] = user.id
    redirect_to root_url, :notice => "Signed in successfully!"
  else
    flash.now.alert = "Invalid email and/or password"
    render :new
  end
end

8. Now create a new class method `find_or_create_with_omniauth` within the User model. The method attempts to find and return a User instance matching the provider and UID given by OmniAuth. If no such user exists, the method creates and returns a new User instance with the email and name given by OmniAuth. And because all users are required to have a password, a random password is specified for the new user.

```ruby
def self.find_or_create_with_omniauth(omniauth)
  where(:provider => omniauth["provider"], :provider_uid => omniauth["uid"])
  .first_or_create!(:email => omniauth["info"]["email"],
                   :first_name => omniauth["info"]["name"],
                   :password => SecureRandom.hex)
end
```

9. Depending on the provider used to authenticate, the authentication data returned by OmniAuth isn’t guaranteed to contain an email address or name for the user. For instance, Twitter never provides an email address. As such, modify the email and name validations on your User model to be optional, but only if the user has a provider specified. Replace your existing email and name validations with something like the following.

```ruby
validates_presence_of :email, :first_name, :last_name, :unless => :provider?
validates_uniqueness_of :email, :allow_nil => true
```

10. If your application sends a welcome email to new users, you’ll want to modify your `UserObserver` to skip users without an email specified.

```ruby
def after_create(user)
  UserMailer.welcome_email(user).deliver if user.email?
end
```

11. Finally, add a new sign in link for Twitter to your sign in page. Click the link, and if everything works, your application should redirect to Twitter. Once you authenticate with Twitter, you should be redirected back to your application and see a success message.

```html
<p>
  Sign in using:
  <%= link_to "Twitter", "/auth/twitter" %>
</p>
```

12. Now add a second provider. Be sure to read the README for the OmniAuth strategy that adds support for the provider, as well as any documentation published by the provider. Like Twitter, many providers require you to register your application with them, at which point they’ll provide you with a key and secret that you’ll configure with OmniAuth. At a minimum, you’ll need to add a gem for the OmniAuth strategy, add the provider to OmniAuth’s configuration in the initializer you created earlier, and add a new sign in link for the provider to your sign in page.

**Resources**

- Use the example from class as a reference when completing the steps above.
OmniAuth

OmniAuth Strategies

Submission

- Show your completed assignment to the instructor during class or office hours to receive credit.