Ex 05: To integrate Devise for authentication and authorization

Aim:

Integrating Devise or another authentication gem into a Rails application is a common practice to handle user authentication and authorization.

Do with Ruby

Step-1: *Install Devise*: Add Devise to your Gemfile and install it by running bundle install.

```
/> gem 'devise'

Then run:
/> bundle install
```

Step-02: *Generate Devise Configuration*: Run the Devise generator to set up its configuration files.

```
/> rails generate devise:install
```

Do with sql:

Step-03: *Generate User Model*: Create a User model if you haven't already. Devise will handle the necessary configurations.

```
/> rails generate devise User
```

Step-04: *Database Migration*: Run the migration to apply the changes to the database.

```
/> rails db:migrate
```

Do with ruby:

Step-05: *Configure Routes*: Devise automatically sets up routes for authentication. You can customize these routes if needed.

```
# config/routes.rb

/> devise_for :users
```

Step-06: *Controllers and Views*: Devise provides controllers and views for authentication out of the box. If you need to customize them, you can generate them and customize as needed.

```
rails generate devise:controllers [scope]

rails generate devise:views [scope]
```

Replace [scope] with the appropriate scope (usually 'users') if needed.

Step-07: Restrict Access: Now, you can restrict access to certain parts of your application by using Devise's authentication helpers in controllers or views.

```
before_action :authenticate_user!
```

This line in a controller ensures that the user must be signed in to access any action in that controller.

Step-08: *Authorization:* For authorization (controlling what users can do within the application), you might want to use additional gems like CanCanCan or Pundit. These gems work well with Devise to provide fine-grained authorization controls.

CanCanCan: A popular authorization library. You define abilities for each user role and check those abilities in your controllers or views.

```
# app/models/ability.rb
class Ability
  include CanCan::Ability

  def initialize(user)
    user ||= User.new # guest user (not logged in)
    if user.admin?
      can :manage, :all
    else
      can :read, :all
    end
  end
end
```

Pundit: Another authorization library. It relies on policy objects to define authorization rules.

Step-09: *Testing*: Don't forget to thoroughly test your authentication and authorization logic using tools like RSpec or MiniTest.

Step-10: *Secure Configuration*: Ensure that you're following security best practices, such as using secure cookies, enabling HTTPS, and protecting sensitive routes.
