TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

BOOKSTORE WEBSITE

Student Name: Cristea Tudor

Group: 30433

CONTENT

1) ABSTRACT

My “Bookstore” website represents a small demo of what an actual book e-shop might look and feel like. It only presents the functionalities that an administrator might have over the website, but the main focus of this project is the security aspect of the entire application, namely protection against various exploits such as “Eavesdropping” (only for production), “Cross-site request forgery (CSRF)”, “Cross-site scripting (XSS)”, “SQL injection” or “Executing protected actions”.

Firstly, “Eavesdropping” is easily avoided if the website uses a secure connection, meaning that all HTTP traffic is encrypted by transforming it using cryptographic techniques driven by a *secret* (such as a password) known only by the two communication parties. Running HTTP over such a secured connection is called HTTPS. Of course, this is already implemented in the background, but it needs to be explicitly enabled in Ruby on Rails in order to activate it (in development it cannot be activated, but in production it is).

Secondly, “CSRF” attacks involve tricking the user’s browser into visiting a different website for which the user has a valid cookie (such as a recently accessed online bank account, in which the user is still logged in), and performing an illicit action on that site as the user. Luckily, with Ruby on Rails it is relatively easy to prevent this kind of attacks through the use of *csrf meta tags* inside the views of the application together with *protection from forgery* inside the controllers of the application.

Thirdly, “XSS” and “SQL injection” attacks are very similar in the sense that a malicious user either runs a script (usually a JavaScript or CSS script) in the case of cross-site scripting or an SQL query in the case of SQL injections in order to determine the website to either perform an action or return results from the website’s database (which is largely represented by sensitive data such as passwords, email addresses and so on, that can be sold on the black market for a lot of money or they can be used by the attacker themselves in order to gain access to various accounts of unsuspecting victims). Fortunately, there are solutions against these types of attacks as well, the main being related to the so-called “sanitization” of user input. This basically implies that every input that is inserted in a form text field or in a search bar (so everything that the user can type somewhere and hit “enter”), needs to be verified for certain keywords such as “SELECT” or “<script></script>” that need to be removed from the input in order to prevent the website from running these sort of commands/actions. However, a better solution than “restricted lists” which filter out these “bad” keywords come in the form of “permitted lists” because a “restricted list” can never be exhaustive enough in order to not be tricked by a crafty, malicious user. A “permitted list”, on the other hand, can only allow certain “whitelisted” words or phrases, but still accept the “bad” ones, with the difference that they will not trigger an action from the website’s server.

Moreover, protecting the website from “executing protected actions” by a malicious user is done through several steps, the first and most important of which is to ensure that certain “sensitive” methods from controllers are declared using the *private* or *protected* keyword. Another step that can be taken in this direction is to restrict the use of certain methods only after other actions have been taken (for example, do not let users access a page that should not be accessible to them unless they are logged in).

Furthermore, other precautionary measures can be taken in order to increase the security of a web application including: forcing the use of SHA256 encoding on all cookies and converting any, old cookies that used the SHA1 encoding to the newer, more secure one; hiding sensitive information such as password details, login credentials or SQL queries inside the log files which can easily be accessed by an attacker; and using an encryption, secret key in order to store this sensitive data in the database in the case that the attacker is able to gain access to it.

2) DIAGRAMS

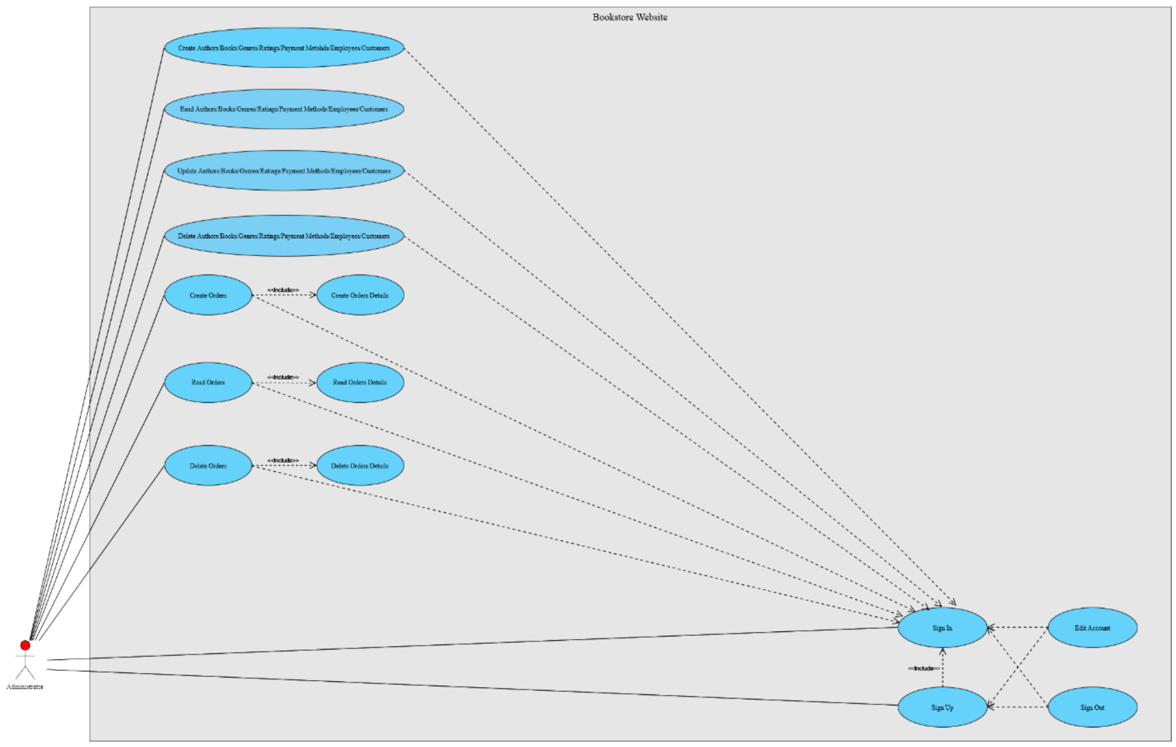
Bookstore Website
Create Authors/Books/Genres/Ratings/Payment Methods/Employees/Customers
Read Authors/Books/Genres/Ratings/Payment Methods/Employees/Customers
Update Authors/Books/Genres/Ratings/Payment Methods/Employees/Customers
Delete Authors/Books/Genres/Ratings/Payment Methods/Employees/Customers
Create Orders
<<Include>>
Create Orders Details
Read Orders
<<Include>>
Read Orders Details
Delete Orders
<<Include>>
Delete Orders Details
Sign In
Edit Account
<<Include>>
Sign Up
Sign Out
Administrator

Fig. 1 - Use Case Diagram

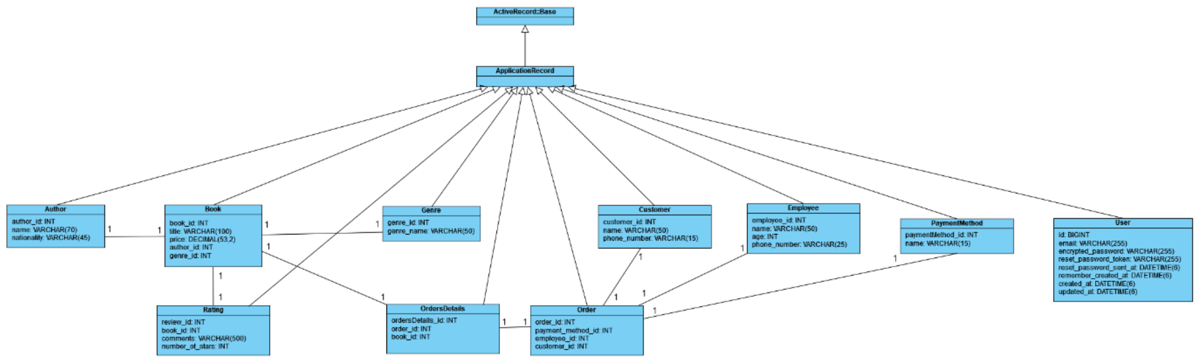

Fig. 2 - Models Class Diagram

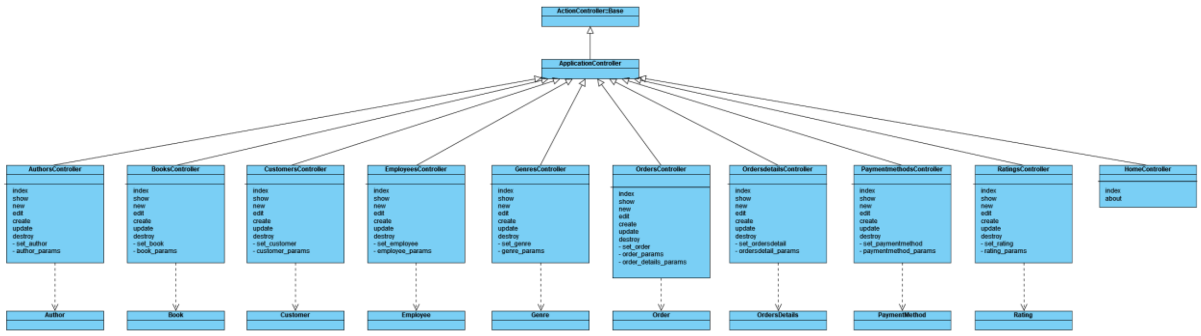

Fig. 3 - Controllers Class Diagram

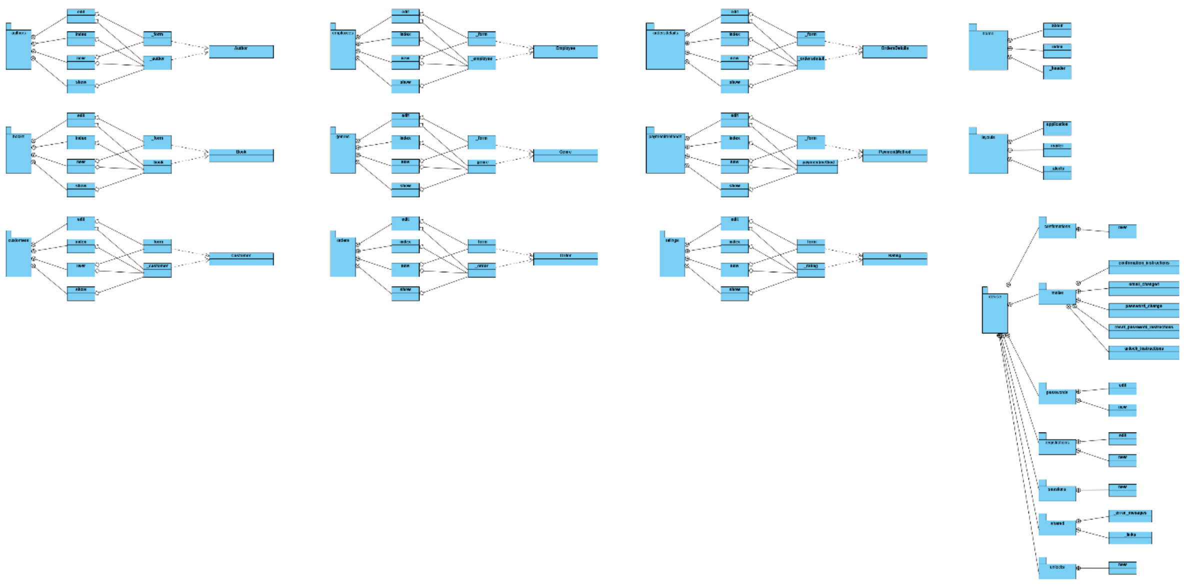

Fig. 4 - Views Class Diagram

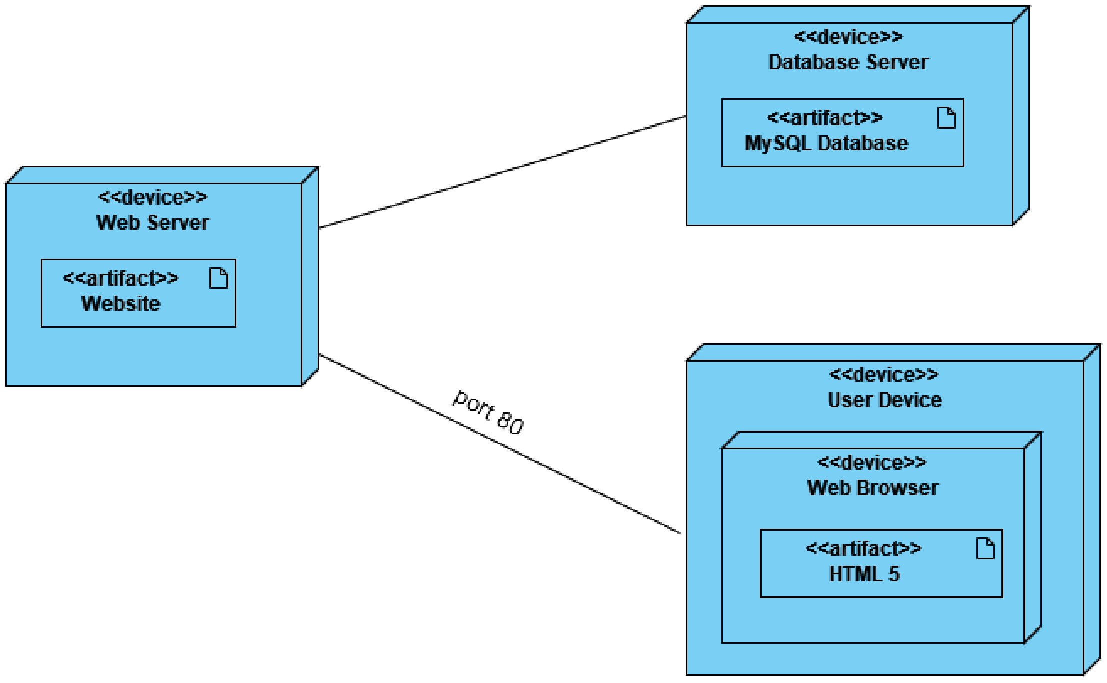

Fig. 5 - Deployment Diagram

3) SOURCE CODE

app/controllers/application_controller.rb

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

  rescue_from ActionController::InvalidAuthenticityToken do |exception|
    sign_out_user # Example method that will destroy the user cookies
  end

end
```

app/controllers/home_controller.rb

```
class HomeController < ApplicationController
  def index

  end

  def about

  end
end
```

app/controllers/authors_controller.rb

```
class AuthorsController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_author, only: %i[ show edit update destroy ]

  # GET /authors or /authors.json
  def index
    @authors = Author.order(:name)
  end

  # GET /authors/1 or /authors/1.json
  def show
  end

  # GET /authors/new
  def new
    @author = Author.new
  end

  # GET /authors/1/edit
  def edit
  end

  # POST /authors or /authors.json
  def create
    @author = Author.new(author_params)

    respond_to do |format|
      if @author.save
        format.html { redirect_to author_url(@author), notice: "Author was successfully 
created." }
        format.json { render :show, status: :created, location: @author }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @author.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /authors/1 or /authors/1.json
  def update
    respond_to do |format|
      if @author.update(author_params)
        format.html { redirect_to author_url(@author), notice: "Author was successfully 
updated." }
        format.json { render :show, status: :ok, location: @author }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @author.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /authors/1 or /authors/1.json
  def destroy
    @author.destroy!

    respond_to do |format|
      format.html { redirect_to authors_url, notice: "Author was successfully destroyed." 
}
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_author
      @author = Author.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def author_params
      params.require(:author).permit(:name, :nationality)
    end
end
```

app/controllers/books_controller.rb

```ruby
class BooksController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_book, only: %i[ show edit update destroy ]

  # GET /books or /books.json
  def index
    @books = Book.order(:title)
  end

  # GET /books/1 or /books/1.json
  def show
  end

  # GET /books/new
  def new
    @book = Book.new
  end

  # GET /books/1/edit
  def edit
  end

  # POST /books or /books.json
  def create
    @book = Book.new(book_params)

    respond_to do |format|
      if @book.save
        format.html { redirect_to book_url(@book), notice: "Book was successfully 
created." }
        format.json { render :show, status: :created, location: @book }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @book.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /books/1 or /books/1.json
  def update
    respond_to do |format|
      if @book.update(book_params)
        format.html { redirect_to book_url(@book), notice: "Book was successfully 
updated." }
        format.json { render :show, status: :ok, location: @book }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @book.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /books/1 or /books/1.json
  def destroy
    @book.destroy!

    respond_to do |format|
      format.html { redirect_to books_url, notice: "Book was successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_book
      @book = Book.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def book_params
      params.require(:book).permit(:title, :price, :author_id, :genre_id)
    end
end
```

app/controllers/customers_controller.rb

```
class CustomersController < ApplicationController
  before_action :authenticate_user!
  before_action :set_customer, only: %i[ show edit update destroy ]

  # GET /customers or /customers.json
  def index
    @customers = Customer.order(:name)
  end

  # GET /customers/1 or /customers/1.json
  def show
  end

  # GET /customers/new
  def new
    @customer = Customer.new
  end

  # GET /customers/1/edit
  def edit
  end

  # POST /customers or /customers.json
  def create
    @customer = Customer.new(customer_params)

    respond_to do |format|
      if @customer.save
        format.html { redirect_to customer_url(@customer), notice: "Customer was 
successfully created." }
        format.json { render :show, status: :created, location: @customer }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @customer.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /customers/1 or /customers/1.json
  def update
    respond_to do |format|
      if @customer.update(customer_params)
        format.html { redirect_to customer_url(@customer), notice: "Customer was 
successfully updated." }
        format.json { render :show, status: :ok, location: @customer }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @customer.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /customers/1 or /customers/1.json
  def destroy
    @customer.destroy!

    respond_to do |format|
      format.html { redirect_to customers_url, notice: "Customer was successfully 
destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_customer
      @customer = Customer.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def customer_params
      params.require(:customer).permit(:name, :phone_number)
    end
end
```

app/controllers/employees_controller.rb

```
class EmployeesController < ApplicationController
  before_action :authenticate_user!
  before_action :set_employee, only: %i[ show edit update destroy ]

  # GET /employees or /employees.json
  def index
    @employees = Employee.order(:name)
  end

  # GET /employees/1 or /employees/1.json
  def show
  end

  # GET /employees/new
  def new
    @employee = Employee.new
  end

  # GET /employees/1/edit
  def edit
  end

  # POST /employees or /employees.json
  def create
    @employee = Employee.new(employee_params)

    respond_to do |format|
      if @employee.save
        format.html { redirect_to employee_url(@employee), notice: "Employee was
successfully created." }
        format.json { render :show, status: :created, location: @employee }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @employee.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /employees/1 or /employees/1.json
  def update
    respond_to do |format|
      if @employee.update(employee_params)
        format.html { redirect_to employee_url(@employee), notice: "Employee was
successfully updated." }
        format.json { render :show, status: :ok, location: @employee }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @employee.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /employees/1 or /employees/1.json
  def destroy
    @employee.destroy!

    respond_to do |format|
      format.html { redirect_to employees_url, notice: "Employee was successfully
destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_employee
      @employee = Employee.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def employee_params
      params.require(:employee).permit(:name, :age, :phone_number)
    end
end
```

app/controllers/genres_controller.rb

```ruby
class GenresController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_genre, only: %i[ show edit update destroy ]

  # GET /genres or /genres.json
  def index
    @genres = Genre.order(:genre_name)
  end

  # GET /genres/1 or /genres/1.json
  def show
  end

  # GET /genres/new
  def new
    @genre = Genre.new
  end

  # GET /genres/1/edit
  def edit
  end

  # POST /genres or /genres.json
  def create
    @genre = Genre.new(genre_params)

    respond_to do |format|
      if @genre.save
        format.html { redirect_to genre_url(@genre), notice: "Genre was successfully
created." }
        format.json { render :show, status: :created, location: @genre }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @genre.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /genres/1 or /genres/1.json
  def update
    respond_to do |format|
      if @genre.update(genre_params)
        format.html { redirect_to genre_url(@genre), notice: "Genre was successfully
updated." }
        format.json { render :show, status: :ok, location: @genre }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @genre.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /genres/1 or /genres/1.json
  def destroy
    @genre.destroy!

    respond_to do |format|
      format.html { redirect_to genres_url, notice: "Genre was successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_genre
      @genre = Genre.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def genre_params
      params.require(:genre).permit(:genre_name)
    end
end
```

app/controllers/genres_controller.rb

```ruby
class GenresController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_genre, only: %i[ show edit update destroy ]

  # GET /genres or /genres.json
  def index
    @genres = Genre.order(:genre_name)
  end

  # GET /genres/1 or /genres/1.json
  def show
  end

  # GET /genres/new
  def new
    @genre = Genre.new
  end

  # GET /genres/1/edit
  def edit
  end

  # POST /genres or /genres.json
  def create
    @genre = Genre.new(genre_params)

    respond_to do |format|
      if @genre.save
        format.html { redirect_to genre_url(@genre), notice: "Genre was successfully
created." }
        format.json { render :show, status: :created, location: @genre }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @genre.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /genres/1 or /genres/1.json
  def update
    respond_to do |format|
      if @genre.update(genre_params)
        format.html { redirect_to genre_url(@genre), notice: "Genre was successfully
updated." }
        format.json { render :show, status: :ok, location: @genre }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @genre.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /genres/1 or /genres/1.json
  def destroy
    @genre.destroy!

    respond_to do |format|
      format.html { redirect_to genres_url, notice: "Genre was successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_genre
      @genre = Genre.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def genre_params
      params.require(:genre).permit(:genre_name)
    end
end
```

app/controllers/orders_controller.rb

```ruby
class OrdersController < ApplicationController
  before_action :authenticate_user!
  before_action :set_order, only: %i[ show edit update destroy ]

  # GET /orders or /orders.json
  def index
    @orders = Order.all
  end

  # GET /orders/1 or /orders/1.json
  def show
  end

  # GET /orders/new
  def new
    @order = Order.new
  end

  # GET /orders/1/edit
  def edit
    return
  end

  # POST /orders or /orders.json
  def create
    @order = Order.new(order_params)

    order_created = true

    respond_to do |format|
      if @order.save
        book_ids = order_details_params
        book_ids.each do |book_id|

          actual_book_id = book_id.last
          next if actual_book_id.blank?

          logger.error(actual_book_id)
          ordersdetail = Ordersdetail.new(order_id: @order.order_id, book_id: actual_book_id)
          unless ordersdetail.save
            order_created = false
          end
        end

        if order_created
          format.html { redirect_to order_url(@order), notice: "Order was successfully created." }
          format.json { render :show, status: :created, location: @order }
        else
          format.html { render :new, status: :unprocessable_entity }
          format.json { render json: @order.errors, status: :unprocessable_entity }
        end
      end
    end
  end

  # PATCH/PUT /orders/1 or /orders/1.json
  def update
    return
    respond_to do |format|
      if @order.update(order_params)
        format.html { redirect_to order_url(@order), notice: "Order was successfully updated." }
        format.json { render :show, status: :ok, location: @order }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @order.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /orders/1 or /orders/1.json
  def destroy
    orders_details = Ordersdetail.where(order_id: @order.order_id)
    orders_details.each do |orders_detail|
      orders_detail.destroy!
    end
    @order.destroy!

    respond_to do |format|
      format.html { redirect_to orders_url, notice: "Order was successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_order
      @order = Order.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def order_params
      params.require(:order).permit(:payment_method_id, :employee_id, :customer_id)
    end

    def order_details_params
      params.require(:order).permit(:book_id_1, :book_id_2, :book_id_3, :book_id_4, :book_id_5)
    end
end
```

app/controllers/ordersdetails_controller.rb

```ruby
class OrdersdetailsController < ApplicationController
  before_action :authenticate_user!
  before_action :set_ordersdetail, only: %i[ show edit update destroy ]

  # GET /ordersdetails or /ordersdetails.json
  def index
    return
    @ordersdetails = Ordersdetail.all
  end

  # GET /ordersdetails/1 or /ordersdetails/1.json
  def show
    return
  end

  # GET /ordersdetails/new
  def new
    return
    @ordersdetail = Ordersdetail.new
  end

  # GET /ordersdetails/1/edit
  def edit
    return
  end

  # POST /ordersdetails or /ordersdetails.json
  def create
    return
    @ordersdetail = Ordersdetail.new(ordersdetail_params)

    respond_to do |format|
      if @ordersdetail.save
        format.html { redirect_to ordersdetail_url(@ordersdetail), notice: "Ordersdetail was
successfully created." }
        format.json { render :show, status: :created, location: @ordersdetail }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @ordersdetail.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /ordersdetails/1 or /ordersdetails/1.json
  def update
    return
    respond_to do |format|
      if @ordersdetail.update(ordersdetail_params)
        format.html { redirect_to ordersdetail_url(@ordersdetail), notice: "Ordersdetail was
successfully updated." }
        format.json { render :show, status: :ok, location: @ordersdetail }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @ordersdetail.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /ordersdetails/1 or /ordersdetails/1.json
  def destroy
    return
    @ordersdetail.destroy!

    respond_to do |format|
      format.html { redirect_to ordersdetails_url, notice: "Ordersdetail was successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_ordersdetail
      @ordersdetail = Ordersdetail.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def ordersdetail_params
      params.require(:ordersdetail).permit(:order_id, :book_id)
    end
end
```

app/controllers/paymentmethods_controller.rb

```ruby
class PaymentmethodsController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_paymentmethod, only: %i[ show edit update destroy ]

  # GET /paymentmethods or /paymentmethods.json
  def index
    @paymentmethods = Paymentmethod.order(:name)
  end

  # GET /paymentmethods/1 or /paymentmethods/1.json
  def show
  end

  # GET /paymentmethods/new
  def new
    @paymentmethod = Paymentmethod.new
  end

  # GET /paymentmethods/1/edit
  def edit
  end

  # POST /paymentmethods or /paymentmethods.json
  def create
    @paymentmethod = Paymentmethod.new(paymentmethod_params)

    respond_to do |format|
      if @paymentmethod.save
        format.html { redirect_to paymentmethod_url(@paymentmethod), notice: 
"Paymentmethod was successfully created." }
        format.json { render :show, status: :created, location: @paymentmethod }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @paymentmethod.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /paymentmethods/1 or /paymentmethods/1.json
  def update
    respond_to do |format|
      if @paymentmethod.update(paymentmethod_params)
        format.html { redirect_to paymentmethod_url(@paymentmethod), notice: 
"Paymentmethod was successfully updated." }
        format.json { render :show, status: :ok, location: @paymentmethod }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @paymentmethod.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /paymentmethods/1 or /paymentmethods/1.json
  def destroy
    @paymentmethod.destroy!

    respond_to do |format|
      format.html { redirect_to paymentmethods_url, notice: "Paymentmethod was 
successfully destroyed." }
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_paymentmethod
      @paymentmethod = Paymentmethod.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def paymentmethod_params
      params.require(:paymentmethod).permit(:name)
    end
end
```

app/controllers/ratings_controller.rb

```ruby
class RatingsController < ApplicationController
  before_action :authenticate_user!, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_rating, only: %i[ show edit update destroy ]

  # GET /ratings or /ratings.json
  def index
    @ratings = Rating.all
  end

  # GET /ratings/1 or /ratings/1.json
  def show
  end

  # GET /ratings/new
  def new
    @rating = Rating.new
  end

  # GET /ratings/1/edit
  def edit
  end

  # POST /ratings or /ratings.json
  def create
    @rating = Rating.new(rating_params)

    respond_to do |format|
      if @rating.save
        format.html { redirect_to rating_url(@rating), notice: "Rating was successfully
created." }
        format.json { render :show, status: :created, location: @rating }
      else
        format.html { render :new, status: :unprocessable_entity }
        format.json { render json: @rating.errors, status: :unprocessable_entity }
      end
    end
  end

  # PATCH/PUT /ratings/1 or /ratings/1.json
  def update
    respond_to do |format|
      if @rating.update(rating_params)
        format.html { redirect_to rating_url(@rating), notice: "Rating was successfully
updated." }
        format.json { render :show, status: :ok, location: @rating }
      else
        format.html { render :edit, status: :unprocessable_entity }
        format.json { render json: @rating.errors, status: :unprocessable_entity }
      end
    end
  end

  # DELETE /ratings/1 or /ratings/1.json
  def destroy
    @rating.destroy!

    respond_to do |format|
      format.html { redirect_to ratings_url, notice: "Rating was successfully destroyed."
}
      format.json { head :no_content }
    end
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_rating
      @rating = Rating.find(params[:id])
    end

    # Only allow a list of trusted parameters through.
    def rating_params
      params.require(:rating).permit(:book_id, :comments, :number_of_stars)
    end
end
```

app/models/application_record.rb

```ruby
class ApplicationRecord < ActiveRecord::Base
  primary_abstract_class
end
```

app/models/author.rb

```ruby
class Author < ApplicationRecord
end
```

app/models/book.rb

```ruby
class Book < ApplicationRecord
  belongs_to :author
  belongs_to :genre
end
```

app/models/customer.rb

```ruby
class Customer < ApplicationRecord
end
```

app/models/employee.rb

```ruby
class Employee < ApplicationRecord
end
```

app/models/genre.rb

```ruby
class Genre < ApplicationRecord
end
```

app/models/order.rb

```ruby
class Order < ApplicationRecord
  belongs_to :paymentmethod, :optional => true
  belongs_to :employee
  belongs_to :customer
end
```

app/models/ordersdetail.rb

```ruby
class Ordersdetail < ApplicationRecord
  belongs_to :order
  belongs_to :book
end
```

app/models/paymentmethod.rb

```ruby
class Paymentmethod < ApplicationRecord
end
```

app/models/rating.rb

```ruby
class Rating < ApplicationRecord
  belongs_to :book
end
```

app/models/user.rb

```ruby
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable
end
```

db/schema.rb

```ruby
ActiveRecord::Schema[7.1].define(version: 2023_12_22_191946) do
  create_table "authors", primary_key: "author_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci",
force: :cascade do |t|
    t.string "name", limit: 70, null: false
    t.string "nationality", limit: 45, null: false
    t.index ["author_id"], name: "author_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
  end

  create_table "books", primary_key: "book_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force:
:cascade do |t|
    t.string "title", limit: 100, null: false
    t.decimal "price", precision: 53, scale: 2, null: false
    t.integer "author_id", null: false
    t.integer "genre_id", null: false
    t.index ["author_id"], name: "author_id"
    t.index ["book_id"], name: "book_id_UNIQUE", unique: true
    t.index ["genre_id"], name: "genre_id_idx"
  end

  create_table "customers", primary_key: "customer_id", id: :integer, charset: "utf8mb4", collation:
"utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 50, null: false
    t.string "phone_number", limit: 15
    t.index ["customer_id"], name: "customer_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
  end

  create_table "database_structures", charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "employees", primary_key: "employee_id", id: :integer, charset: "utf8mb4", collation:
"utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 50, null: false
    t.integer "age", null: false
    t.string "phone_number", limit: 25, null: false
    t.index ["employee_id"], name: "employee_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
    t.check_constraint "(`age` > 15) and (`age` < 71)", name: "employees_chk_1"
  end

  create_table "genres", primary_key: "genre_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci",
force: :cascade do |t|
    t.string "genre_name", limit: 50, null: false
    t.index ["genre_id"], name: "genre_id_UNIQUE", unique: true
    t.index ["genre_name"], name: "genre_name_UNIQUE", unique: true
  end

  create_table "orders", primary_key: "order_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci",
force: :cascade do |t|
    t.integer "payment_method_id", null: false
    t.integer "employee_id", null: false
    t.integer "customer_id", null: false
    t.index ["customer_id"], name: "customer_id_idx"
    t.index ["employee_id"], name: "employee_id_idx"
    t.index ["order_id"], name: "order_id_UNIQUE", unique: true
    t.index ["payment_method_id"], name: "payment_method_id_idx"
  end

  create_table "ordersdetails", primary_key: "ordersDetails_id", id: :integer, charset: "utf8mb4", collation:
"utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.integer "order_id", null: false
    t.integer "book_id", null: false
    t.index ["book_id"], name: "book_id_idx"
    t.index ["order_id"], name: "order_id_idx"
    t.index ["ordersDetails_id"], name: "ordersDetails_id_UNIQUE", unique: true
  end

  create_table "paymentmethods", primary_key: "paymentMethod_id", id: :integer, charset: "utf8mb4", collation:
"utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 15, null: false
    t.index ["name"], name: "name_UNIQUE", unique: true
    t.index ["paymentMethod_id"], name: "paymentMethod_id_UNIQUE", unique: true
  end

  create_table "ratings", primary_key: "review_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci",
force: :cascade do |t|
    t.integer "book_id", null: false
    t.string "comments", limit: 500
    t.integer "number_of_stars", null: false
    t.index ["book_id"], name: "book_id_idx"
    t.index ["review_id"], name: "review_id_UNIQUE", unique: true
    t.check_constraint "(`number_of_stars` > 0) and (`number_of_stars` < 6)", name: "ratings_chk_1"
  end

  create_table "users", charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "email", default: "", null: false
    t.string "encrypted_password", default: "", null: false
    t.string "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.index ["email"], name: "index_users_on_email", unique: true
    t.index ["reset_password_token"], name: "index_users_on_reset_password_token", unique: true
  end

end
```

app/views/authors/_author.html.erb

```
<div id="<%= dom_id author %>">
  <p>
    <strong>Name:</strong>
    <%= author.name %>
  </p>

  <p>
    <strong>Nationality:</strong>
    <%= author.nationality %>
  </p>

</div>
```

app/views/authors/_form.html.erb

```
<%= form_with(model: author) do |form| %>
  <% if author.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(author.errors.count, "error") %> prohibited this
author from being saved:</h2>

      <ul>
        <% author.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-group">
    <%= form.label :name, style: "display: block" %>
    <%= form.text_field :name, class: "form-control", style: "width:
200px;" %>
  </div>

  <div class="form-group">
    <%= form.label :nationality, style: "display: block" %>
    <%= form.text_field :nationality, class: "form-control", style:
"width: 200px;" %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/authors/edit.html.erb

```
<h1>Edit Author</h1>
<br>
<%= render "form", author: @author %>

<br>

<div>
  <%= link_to "Back", authors_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/authors/new.html.erb

```
<h1>New Author</h1>
<br>
<%= render "form", author: @author %>

<br>

<div>
  <%= link_to "Back", authors_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/authors/show.html.erb

```
<%= render @author %>

<div>
  <%= link_to "Back", authors_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/books/_book.html.erb

```
<div id="<%= dom_id book %>">
  <p>
    <strong>Title:</strong>
    <%= book.title %>
  </p>

  <p>
    <strong>Price:</strong>
    <%= book.price %>
  </p>

  <p>
    <strong>Author:</strong>
    <% author = Author.find_by(author_id: book.author_id) %>
    <%= author.name if author %>
  </p>

  <p>
    <strong>Genre:</strong>
    <% genre = Genre.find_by(genre_id: book.genre_id) %>
    <%= genre.genre_name if genre %>
  </p>

</div>
```

app/views/books/edit.html.erb

```
<h1>Edit Book</h1>
<br>
<%= render "form", book: @book %>

<br>

<div>
  <%= link_to "Back", books_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/books/_form.html.erb

```
<%= form_with(model: book) do |form| %>
  <% if book.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(book.errors.count, "error") %> prohibited this
book from being saved:</h2>

      <ul>
        <% book.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div>
    <%= form.label :title, style: "display: block" %>
    <%= form.text_field :title, class: "form-control", style: "width:
200px;" %>
  </div>

  <div>
    <%= form.label :price, style: "display: block" %>
    <%= form.text_field :price, class: "form-control", style: "width:
200px;" %>
  </div>

  <div>
    <%= form.label :author_id, style: "display: block" %>
    <%= form.select :author_id, Author.all.map { |author| [author.name,
author.author_id] }, class: "form-control" %>
  </div>

  <div>
    <%= form.label :genre_id, style: "display: block" %>
    <%= form.select :genre_id, Genre.all.map { |genre|
[genre.genre_name, genre.genre_id] }, class: "form-control" %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/books/new.html.erb

```
<h1>New Book</h1>
<br>
<%= render "form", book: @book %>

<br>

<div>
  <%= link_to "Back", books_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/books/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Books</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Book Name</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @books.each do |book| %>
      <tr>
        <td><%= book.title %></td>
        <td>
          <%= link_to "Show", book, class: "btn btn-primary" %>
        </td>
        <% if user_signed_in? %>
          <td>
            <%= link_to "Edit", edit_book_path(book), class: "btn btn-
warning", style: "color: white;"%>
          </td>
          <td>
            <%= button_to "Destroy", book, method: :delete, class: "btn
btn-danger"%>
          </td>
        <% end %>
      </tr>
    <% end %>
    </tbody>
  </table>
  <% if user_signed_in? %>
    <%= link_to "Create New Book", new_book_path, class: "btn btn-
success" %>
  <% end %>
</div>
<br><br>
```

app/views/books/show.html.erb

```
<%= render @book %>

<div>
  <%= link_to "Back", books_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/customers/edit.html.erb

```
<h1>Edit Customer</h1>

<%= render "form", customer: @customer %>

<br>

<div>
  <%= link_to "Back", customers_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/customers/_customer.html.erb

```
<div id="<%= dom_id customer %>">
  <p>
    <strong>Name:</strong>
    <%= customer.name %>
  </p>

  <p>
    <strong>Phone number:</strong>
    <%= customer.phone_number %>
  </p>

</div>
```

app/views/customers/_form.html.erb

```
<%= form_with(model: customer) do |form| %>
  <% if customer.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(customer.errors.count, "error") %> prohibited
this customer from being saved:</h2>

      <ul>
        <% customer.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-group">
    <%= form.label :name, style: "display: block" %>
    <%= form.text_field :name, class: "form-control", style: "width:
200px;" %>
  </div>

  <div class="form-group">
    <%= form.label :phone_number, style: "display: block" %>
    <%= form.text_field :phone_number, class: "form-control", style:
"width: 200px;"  %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/customers/show.html.erb

```
<%= render @customer %>

<div>
  <%= link_to "Back", customers_path, class: "btn btn-info btn-sm",
style: "color: white;"%>
</div>
```

app/views/customers/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Customers</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Customer Name</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @customers.each do |customer| %>
      <tr>
        <td><%= customer.name %></td>
        <td>
          <%= link_to "Show", customer, class: "btn btn-primary" %>
        </td>
        <td>
          <%= link_to "Edit", edit_customer_path(customer), class: "btn
btn-warning", style: "color: white;"%>
        </td>
        <td>
          <%= button_to "Destroy", customer, method: :delete, class:
"btn btn-danger"%>
        </td>
      </tr>
    <% end %>
    </tbody>
  </table>

  <%= link_to "Create New Customer", new_customer_path, class: "btn btn-
success" %>
</div>
<br><br>
```

app/views/customers/new.html.erb

```
<h1>New Customer</h1>

<%= render "form", customer: @customer %>

<br>

<div>
  <%= link_to "Back", customers_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/employees/new.html.erb

```
<h1>New Employee</h1>

<%= render "form", employee: @employee %>

<br>

<div>
  <%= link_to "Back", employees_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/employees/_form.html.erb

```erb
<%= form_with(model: employee) do |form| %>
  <% if employee.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(employee.errors.count, "error") %> prohibited
this employee from being saved:</h2>

      <ul>
        <% employee.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-group">
    <%= form.label :name, style: "display: block" %>
    <%= form.text_field :name, class: "form-control", style: "width:
200px;"  %>
  </div>

  <div class="form-group">
    <%= form.label :age, style: "display: block" %>
    <%= form.number_field :age, class: "form-control", style: "width:
200px;"  %>
  </div>

  <div class="form-group">
    <%= form.label :phone_number, style: "display: block" %>
    <%= form.text_field :phone_number, class: "form-control", style:
"width: 200px;"  %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/employees/_employee.html.erb

```erb
<div id="<%= dom_id employee %>">
  <p>
    <strong>Name:</strong>
    <%= employee.name %>
  </p>

  <p>
    <strong>Age:</strong>
    <%= employee.age %>
  </p>

  <p>
    <strong>Phone number:</strong>
    <%= employee.phone_number %>
  </p>

</div>
```

app/views/employees/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Employees</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Employee Name</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @employees.each do |employee| %>
      <tr>
        <td><%= employee.name %></td>
        <td>
          <%= link_to "Show", employee, class: "btn btn-primary" %>
        </td>
        <td>
          <%= link_to "Edit", edit_employee_path(employee), class: "btn
btn-warning", style: "color: white;"%>
        </td>
        <td>
          <%= button_to "Destroy", employee, method: :delete, class:
"btn btn-danger"%>
        </td>
      </tr>
    <% end %>
    </tbody>
  </table>

  <%= link_to "Create New Employee", new_employee_path, class: "btn btn-
success" %>
</div>
<br><br>
```

app/views/employees/edit.html.erb

```
<h1>Edit Employee</h1>

<%= render "form", employee: @employee %>

<br>

<div>
  <%= link_to "Back", employees_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/employees/edit.html.erb

```
<%= render @employee %>


<%= link_to "Back", employees_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/genres/_form.html.erb

```
<%= form_with(model: genre) do |form| %>
  <% if genre.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(genre.errors.count, "error") %> prohibited this
genre from being saved:</h2>

      <ul>
        <% genre.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-group">
    <%= form.label :genre_name, style: "display: block" %>
    <%= form.text_field :genre_name, class: "form-control", style:
"width: 200px;" %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/genres/new.html.erb

```
<h1>New Genre</h1>
<br>
<%= render "form", genre: @genre %>

<br>

<div>
  <%= link_to "Back", genres_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/genres/edit.html.erb

```
<h1>Edit Genre</h1>
<br>
<%= render "form", genre: @genre %>

<br>

<div>
  <%= link_to "Back", genres_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/genres/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Genres</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Genre Name</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @genres.each do |genre| %>
      <tr>
        <td><%= genre.genre_name %></td>
        <td>
          <%= link_to "Show", genre, class: "btn btn-primary" %>
        </td>
        <% if user_signed_in? %>
          <td>
            <%= link_to "Edit", edit_genre_path(genre), class: "btn btn-
warning", style: "color: white;"%>
          </td>
          <td>
            <%= button_to "Destroy", genre, method: :delete, class: "btn
btn-danger"%>
          </td>
        <% end %>
      </tr>
    <% end %>
    </tbody>
  </table>
  <% if user_signed_in? %>
    <%= link_to "Create New Genre", new_genre_path, class: "btn btn-
success" %>
  <% end %>
</div>
<br><br>
```

app/views/genres/_genre.html.erb

```
<div id="<%= dom_id genre %>">
  <p>
    <strong>Genre name:</strong>
    <%= genre.genre_name %>
  </p>

</div>
```

app/views/genres/show.html.erb

```
<%= render @genre %>

<div>
  <%= link_to "Back", genres_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/orders/_form.html.erb

```
<%= form_with(model: order) do |form| %>
  <% if order.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(order.errors.count, "error") %> prohibited this order from being
saved:</h2>

      <ul>
        <% order.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div>
    <%= form.label :payment_method_id, style: "display: block" %>
    <%= form.select :payment_method_id, Paymentmethod.all.map { |paymentmethod|
[paymentmethod.name, paymentmethod.paymentMethod_id] }, class: "form-control" %>
  </div>

  <div>
    <%= form.label :employee_id, style: "display: block" %>
    <%= form.select :employee_id, Employee.all.map { |employee| [employee.name,
employee.employee_id] }, class: "form-control" %>
  </div>

  <div>
    <%= form.label :customer_id, style: "display: block" %>
    <%= form.select :customer_id, Customer.all.map { |customer| [customer.name,
customer.customer_id] }, class: "form-control" %>
  </div>

  <div>
    <%= form.label :"Book 1", style: "display: block" %>
    <% book_options = Book.all.map { |book| [book.title, book.book_id] } %>
    <% book_options.unshift(["None", nil]) %>
    <%= form.select :book_id_1, book_options, class: "form-control" %>
  </div>

  <div>
    <%= form.label :"Book 2", style: "display: block" %>
    <% book_options = Book.all.map { |book| [book.title, book.book_id] } %>
    <% book_options.unshift(["None", nil]) %>
    <%= form.select :book_id_2, book_options, class: "form-control" %>
  </div>

  <div>
    <%= form.label :"Book 3", style: "display: block" %>
    <% book_options = Book.all.map { |book| [book.title, book.book_id] } %>
    <% book_options.unshift(["None", nil]) %>
    <%= form.select :book_id_3, book_options, class: "form-control" %>
  </div>

  <div>
    <%= form.label :"Book 4", style: "display: block" %>
    <% book_options = Book.all.map { |book| [book.title, book.book_id] } %>
    <% book_options.unshift(["None", nil]) %>
    <%= form.select :book_id_4, book_options, class: "form-control" %>
  </div>

  <div>
    <%= form.label :"Book 5", style: "display: block" %>
    <% book_options = Book.all.map { |book| [book.title, book.book_id] } %>
    <% book_options.unshift(["None", nil]) %>
    <%= form.select :book_id_5, book_options, class: "form-control" %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/orders/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Orders</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Order Id</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @orders.each do |order| %>
      <tr>
        <td><%= order.order_id %></td>
        <td>
          <%= link_to "Show", order, class: "btn btn-primary" %>
        </td>
        <td>
          <%= button_to "Destroy", order, method: :delete, class: "btn
btn-danger"%>
        </td>
      </tr>
    <% end %>
    </tbody>
  </table>

  <%= link_to "Create New Order", new_order_path, class: "btn btn-
success" %>
</div>
<br><br>
```

app/views/orders/_order.html.erb

```
<div id="<%= dom_id order %>">
  <p>
    <strong>Payment method:</strong>
    <%= order.payment_method_id %>
  </p>

  <p>
    <strong>Employee:</strong>
    <%= order.employee_id %>
  </p>

  <p>
    <strong>Customer:</strong>
    <%= order.customer_id %>
  </p>

</div>
```

app/views/orders/show.html.erb

```
<p><strong>Payment Method: </strong><%=
Paymentmethod.find(@order.payment_method_id).name %></p>
<p><strong>Employee: </strong><%= Employee.find(@order.employee_id).name
%></p>
<p><strong>Customer: </strong><%= Customer.find(@order.customer_id).name
%></p>
<p><strong>Books: </strong>
  <% Ordersdetail.where(order_id: @order.order_id).each_with_index do
|order_details, index| %>
  <%= Book.find(order_details.book_id).title %>
    <% unless index == Ordersdetail.where(order_id:
@order.order_id).count - 1 %> | <% end %>
  <% end %>
</p>

<p><strong>Total Price: </strong>
  <% total_price = 0 %>
  <% Ordersdetail.where(order_id: @order.order_id).each do
|order_details| %>
    <% total_price += Book.find(order_details.book_id).price %>
  <% end %>
  <%= total_price %>
</p>

<div>
  <%= link_to "Back", orders_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/orders/new.html.erb

```
<h1>New Order</h1>

<%= render "form", order: @order %>

<br>

<div>
  <%= link_to "Back", orders_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/orders/edit.html.erb

```
<h1>Edit Order</h1>

<%= render "form", order: @order %>

<br>

<div>
  <%= link_to "Back", orders_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/orders/edit.html.erb

```
<h1>Edit Order</h1>

<%= render "form", order: @order %>

<br>

<div>
  <%= link_to "Back", orders_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/paymentmethods/index.html.erb

```
<div class="container">
  <h1 class="text-center">List of Payment Methods</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Payment Method Name</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @paymentmethods.each do |paymentmethod| %>
      <tr>
        <td><%= paymentmethod.name %></td>
        <td>
          <%= link_to "Show", paymentmethod, class: "btn btn-primary" %>
        </td>
        <% if user_signed_in? %>
          <td>
            <%= link_to "Edit", edit_paymentmethod_path(paymentmethod),
class: "btn btn-warning", style: "color: white;"%>
          </td>
          <td>
            <%= button_to "Destroy", paymentmethod, method: :delete,
class: "btn btn-danger"%>
          </td>
        <% end %>
      </tr>
    <% end %>
    </tbody>
  </table>
  <% if user_signed_in? %>
    <%= link_to "Create New Payment Method", new_paymentmethod_path,
class: "btn btn-success" %>
  <% end %>
</div>
<br><br>
```

app/views/paymentmethods/show.html.erb

```
<%= render @paymentmethod %>

<div>
  <%= link_to "Back", paymentmethods_path, class: "btn btn-info btn-sm",
style: "color: white;"%>
</div>
```

app/views/paymentmethods/_form.html.erb

```
<%= form_with(model: paymentmethod) do |form| %>
  <% if paymentmethod.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(paymentmethod.errors.count, "error") %>
prohibited this paymentmethod from being saved:</h2>

      <ul>
        <% paymentmethod.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div class="form-group">
    <%= form.label :name, style: "display: block" %>
    <%= form.text_field :name, class: "form-control", style: "width:
200px;"  %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/paymentmethods/new.html.erb

```
<h1>New Payment Method</h1>

<%= render "form", paymentmethod: @paymentmethod %>

<br>

<div>
  <%= link_to "Back", paymentmethods_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/paymentmethods/edit.html.erb

```
<h1>Edit Payment Method</h1>

<%= render "form", paymentmethod: @paymentmethod %>

<br>

<div>
  <%= link_to "Back", paymentmethods_path, class: "btn btn-info btn-sm",
style: "color: white"%>
</div>
```

app/views/paymentmethods/_paymentmethod.html.erb

```
<div id="<%= dom_id paymentmethod %>">
  <p>
    <strong>Name:</strong>
    <%= paymentmethod.name %>
  </p>
</div>
```

app/views/ratings/_form.html.erb

```
<%= form_with(model: rating) do |form| %>
  <% if rating.errors.any? %>
    <div style="color: red">
      <h2><%= pluralize(rating.errors.count, "error") %> prohibited this
rating from being saved:</h2>

      <ul>
        <% rating.errors.each do |error| %>
          <li><%= error.full_message %></li>
        <% end %>
      </ul>
    </div>
  <% end %>

  <div>
    <%= form.label :book_id, "Book", style: "display: block" %>
    <%= form.select :book_id, Book.all.map { |book| [book.title,
book.id] }, class: "form-control" %>
  </div>

  <div>
    <%= form.label :comments, style: "display: block" %>
    <%= form.text_area :comments, class: "form-control", style: "width:
200px;" %>
  </div>

  <div>
    <%= form.label :number_of_stars, style: "display: block" %>
    <%= form.text_field :number_of_stars, class: "form-control", style:
"width: 200px;" %>
  </div>

  <br>
  <div class="form-group">
    <%= form.submit class: "btn btn-success"%>
  </div>
<% end %>
```

app/views/ratings/edit.html.erb

```
<h1>Edit Rating</h1>

<%= render "form", rating: @rating %>

<br>

<div>
  <%= link_to "Back", ratings_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/ratings/_rating.html.erb

```
<div id="<%= dom_id rating %>">
</div>
```

app/views/ratings/index.html.erb

```erb
<div class="container">
  <h1 class="text-center">List of Ratings</h1>

  <table class="table table-striped">
    <thead>
    <tr>
      <th style="font-size: 20px">Rating Id</th>
      <th></th>
    </tr>
    </thead>
    <tbody class="table-group-divider">
    <% @ratings.each do |rating| %>
      <tr>
        <td><%= rating.review_id %></td>
        <td>
          <%= link_to "Show", rating, class: "btn btn-primary" %>
        </td>
        <% if user_signed_in? %>
          <td>
            <%= link_to "Edit", edit_rating_path(rating), class: "btn
btn-warning", style: "color: white;"%>
          </td>
          <td>
            <%= button_to "Destroy", rating, method: :delete, class:
"btn btn-danger"%>
          </td>
        <% end %>
      </tr>
    <% end %>
    </tbody>
  </table>
  <% if user_signed_in? %>
    <%= link_to "Create New Rating", new_rating_path, class: "btn btn-
success" %>
  <% end %>
</div>
<br><br>
```

app/views/ratings/new.html.erb

```erb
<h1>New Rating</h1>

<%= render "form", rating: @rating %>

<br>

<div>
  <%= link_to "Back", ratings_path, class: "btn btn-info btn-sm", style:
"color: white"%>
</div>
```

app/views/ratings/show.html.erb

```erb
<p><strong>Book: </strong><%= Book.find(@rating.book_id).title %></p>
<p><strong>Comments: </strong><%= @rating.comments %></p>
<p><strong>Number of Stars: </strong><%= @rating.number_of_stars %></p>

<div>
  <%= link_to "Back", ratings_path, class: "btn btn-info btn-sm", style:
"color: white;"%>
</div>
```

app/views/home/_header.html.erb

```
<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <%= link_to root_path, class: "navbar-brand" do %>
      <%= image_tag('bookstore_image.png', alt: 'BookstoreImage', class: 'img-fluid',
style: 'max-width: 100px; height: auto;') %>
    <% end %>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <%= link_to 'Authors', authors_path, class: "nav-link active" %>
        </li>
        <li class="nav-item">
          <%= link_to 'Books', books_path, class: "nav-link active" %>
        </li>
        <li class="nav-item">
          <%= link_to 'Genres', genres_path, class: "nav-link active" %>
        </li>
        <li class="nav-item">
          <%= link_to 'Ratings', ratings_path, class: "nav-link active" %>
        </li>
        <li class="nav-item">
          <%= link_to 'Payment Methods', paymentmethods_path, class: "nav-link active" %>
        </li>
        <% if user_signed_in? %>
          <li class="nav-item">
            <%= link_to 'Customers', customers_path, class: "nav-link active" %>
          </li>
          <li class="nav-item">
            <%= link_to 'Employees', employees_path, class: "nav-link active" %>
          </li>
          <li class="nav-item">
            <%= link_to 'Orders', orders_path, class: "nav-link active" %>
          </li>
        <% end %>
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" id="dropdown" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
            Account
          </a>
          <ul class="dropdown-menu" aria-labelledby="dropdown">
            <% if user_signed_in? %>
              <li class="dropdown-item">
                <%= link_to 'Edit Account', edit_user_registration_path, class: "nav-link
active" %>
              </li>
              <li class="dropdown-item">
                <%= link_to 'Sign Out', destroy_user_session_path, method: :delete, class:
"nav-link active" %>
              </li>
            <% else %>
              <li class="dropdown-item">
                <%= link_to 'Sign Up', new_user_registration_path, class: "nav-link
active" %>
              </li>
              <li class="dropdown-item">
                <%= link_to 'Sign In', new_user_session_path, class: "nav-link active" %>
              </li>
            <% end %>
          </ul>
        </li>
        <li class="nav-item">
          <%= link_to 'About', home_about_path, class: "nav-link active" %>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

app/views/home/about.html.erb

```
<h1>About this Website</h1>

This is a Demo Website for a bookshop. Currently, it is only usable with
Administrator privileges.

<br>

The purpose of this website is to put emphasis on security!
```

app/views/home/index.html.erb

```
<h1>Hello!</h1>

<br>

<h2>Welcome to my Bookshop Website!</h2>
```

app/views/devise/shared/_links.html.erb

```
<%- if controller_name != 'sessions' %>
  <div> Already Have an Account?</div>
  <br>
  <%= link_to "Sign in", new_session_path(resource_name), class:"btn
btn-secondary" %><br />
<% end %>

<%- if devise_mapping.registerable? && controller_name !=
'registrations' %>
  <div> New to My Bookstore? </div>
  <br>
  <%= link_to "Sign up", new_registration_path(resource_name),
class:"btn btn-secondary" %><br />
<% end %>

<%- if devise_mapping.confirmable? && controller_name != 'confirmations'
%>
  <%= link_to "Didn't receive confirmation instructions?",
new_confirmation_path(resource_name) %><br />
<% end %>

<%- if devise_mapping.lockable? &&
resource_class.unlock_strategy_enabled?(:email) && controller_name !=
'unlocks' %>
  <%= link_to "Didn't receive unlock instructions?",
new_unlock_path(resource_name) %><br />
<% end %>

<%- if devise_mapping.omniauthable? %>
  <%- resource_class.omniauth_providers.each do |provider| %>
    <%= button_to "Sign in with #{OmniAuth::Utils.camelize(provider)}",
omniauth_authorize_path(resource_name, provider), data: { turbo: false }
%><br />
  <% end %>
<% end %>
```

app/views/devise/registrations/new.html.erb

```
<div class="card">
  <div class="card-header">
    <h2>Sign Up</h2>
  </div>
  <div class="card-body">
    <%= form_for(resource, as: resource_name, url:
registration_path(resource_name)) do |f| %>
      <%= render "devise/shared/error_messages", resource: resource %>

      <div class="field form-group">
        <%= f.label :email %><br />
        <%= f.email_field :email, class: "form-control", style: "width:
200px;", autofocus: true, autocomplete: "email" %>
      </div>

      <div class="field form-group">
        <%= f.label :password %>
        <% if @minimum_password_length %>
          <em>(<%= @minimum_password_length %> characters minimum)</em>
        <% end %><br />
        <%= f.password_field :password, class: "form-control", style:
"width: 200px;", autocomplete: "new-password" %>
      </div>

      <div class="field form-group">
        <%= f.label :password_confirmation %><br />
        <%= f.password_field :password_confirmation, class: "form-
control", style: "width: 200px;", autocomplete: "new-password" %>
      </div>
      <br>
      <div class="actions">
        <%= f.submit "Sign up", class:"btn btn-primary" %>
      </div>
    <% end %>
  </div>
</div>

 <br>
  <div class="card">
    <div class="card-header">
      <h4>Sign In</h4>
    </div>
    <div class="card-body">
      <%= render "devise/shared/links" %>
    </div>
  </div>
```

app/views/devise/shared/_error_messages.html.erb

```
<% if resource.errors.any? %>
  <div id="error_explanation" data-turbo-cache="false">
    <h2>
      <%= I18n.t("errors.messages.not_saved",
                 count: resource.errors.count,
                 resource: resource.class.model_name.human.downcase)
       %>
    </h2>
    <ul>
      <% resource.errors.full_messages.each do |message| %>
        <li><%= message %></li>
      <% end %>
    </ul>
  </div>
<% end %>
```

app/views/devise/registrations/edit.html.erb

```
<div class="card">
  <div class="card-header">
    <h2>Edit Account</h2>
  </div>
  <div class="card-body">
    <%= form_for(resource, as: resource_name, url:
registration_path(resource_name), html: { method: :put }) do |f| %>
      <%= render "devise/shared/error_messages", resource: resource %>

      <div class="field form-group">
        <%= f.label :email %><br />
        <%= f.email_field :email, class: "form-control", style: "width: 200px;",
autofocus: true, autocomplete: "email" %>
      </div>

      <% if devise_mapping.confirmable? && resource.pending_reconfirmation? %>
        <div>Currently waiting confirmation for: <%= resource.unconfirmed_email
%></div>
      <% end %>

      <div class="field form-group">
        <a>New </a><%= f.label :password %><em> (<%= @minimum_password_length %>
characters minimum)</em><br />
        <%= f.password_field :password, class: "form-control", style: "width:
200px;", autocomplete: "new-password" %>
      </div>

      <div class="field form-group">
        <a>New </a><%= f.label :password_confirmation %><br />
        <%= f.password_field :password_confirmation, class: "form-control",
style: "width: 200px;", autocomplete: "new-password" %>
      </div>

      <br>
      <div class="field form-group">
        <%= f.label :current_password %> <i>(we need your current password to
confirm your changes)</i><br />
        <%= f.password_field :current_password, class: "form-control", style:
"width: 200px;", autocomplete: "current-password" %>
      </div>

      <br>
      <div class="actions">
        <%= f.submit "Update", class:"btn btn-warning" %>
      </div>
    <% end %>
  </div>
</div>
<br>
<div class="card">
  <div class="card-header">
    <h3>Cancel Account</h3>
  </div>
  <div class="card-body">
    <div>Unhappy?</div>
    <br>
    <%= button_to "Cancel Account", registration_path(resource_name), data: {
confirm: "Are you sure?", turbo_confirm: "Are you sure?" }, method: :delete,
class:"btn btn-danger" %>
  </div>
</div>
<br><br>
```

app/views/devise/sessions/new.html.erb

```
<div class="card">
  <div class="card-header">
    <h2>Sign In</h2>
  </div>
  <div class="card-body">
    <%= form_for(resource, as: resource_name, url:
session_path(resource_name)) do |f| %>
      <div class="field form-group">
        <%= f.label :email %><br />
        <%= f.email_field :email, class: "form-control", style: "width:
200px;", autofocus: true, autocomplete: "email" %>
      </div>

      <div class="field form-group">
        <%= f.label :password %><br />
        <%= f.password_field :password, class: "form-control", style:
"width: 200px;", autocomplete: "current-password" %>
      </div>

      <% if devise_mapping.rememberable? %>
        <div class="field form-check">
          <%= f.check_box :remember_me, class: "form-check-input" %>
          <%= f.label :remember_me %>
        </div>
      <% end %>

      <br>
      <div class="actions">
        <%= f.submit "Sign in", class:"btn btn-primary" %>
      </div>

    <% end %>
  </div>
</div>
<br>
<div class="card">
  <div class="card-header">
    <h4>Sign Up</h4>
  </div>
  <div class="card-body">
    <%= render "devise/shared/links" %>
  </div>
</div>
```

config/initializers/filter_parameter_logging.rb

```
# Be sure to restart your server when you modify this file.

# Configure parameters to be partially matched (e.g. passw matches
password) and filtered from the log file.
# Use this to limit dissemination of sensitive information.
# See the ActiveSupport::ParameterFilter documentation for supported
notations and behaviors.

Rails.application.config.filter_parameters += [
  :passw, :secret, :token, :_key, :crypt, :salt, :certificate, :otp,
:ssn
]
```

config/environments/development.rb

```
require "active_support/core_ext/integer/time"

Rails.application.configure do
  # Settings specified here will take precedence over those in config/application.rb.

  # In the development environment your application's code is reloaded any time
  # it changes. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.enable_reloading = true

  # Do not eager load code on boot.
  config.eager_load = false

  # Show full error reports.
  config.consider_all_requests_local = true

  # Enable server timing
  config.server_timing = true

  # Enable/disable caching. By default caching is disabled.
  # Run rails dev:cache to toggle caching.
  if Rails.root.join("tmp/caching-dev.txt").exist?
    config.action_controller.perform_caching = true
    config.action_controller.enable_fragment_cache_logging = true

    config.cache_store = :memory_store
    config.public_file_server.headers = {
      "Cache-Control" => "public, max-age=#{2.days.to_i}"
    }
  else
    config.action_controller.perform_caching = false

    config.cache_store = :null_store
  end

  # Store uploaded files on the local file system (see config/storage.yml for options).
  config.active_storage.service = :local

  # Don't care if the mailer can't send.
  config.action_mailer.raise_delivery_errors = false

  config.action_mailer.perform_caching = false

  # Print deprecation notices to the Rails logger.
  config.active_support.deprecation = :log

  # Raise exceptions for disallowed deprecations.
  config.active_support.disallowed_deprecation = :raise

  # Tell Active Support which deprecation messages to disallow.
  config.active_support.disallowed_deprecation_warnings = []

  # Raise an error on page load if there are pending migrations.
  config.active_record.migration_error = :page_load

  # Highlight code that triggered database queries in logs.
  config.active_record.verbose_query_logs = true

  # Highlight code that enqueued background job in logs.
  config.active_job.verbose_enqueue_logs = true

  # Suppress logger output for asset requests.
  config.assets.quiet = true

  # Raises error for missing translations.
  # config.i18n.raise_on_missing_translations = true

  # Annotate rendered view with file names.
  # config.action_view.annotate_rendered_view_with_filenames = true

  # Uncomment if you wish to allow Action Cable access from any origin.
  # config.action_cable.disable_request_forgery_protection = true

  # Raise error when a before_action's only/except options reference missing actions
  config.action_controller.raise_on_missing_callback_actions = true

  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

  config.action_dispatch.signed_cookie_digest = "SHA256"

  config.action_dispatch.cookies_rotations.tap do |cookies|
    cookies.rotate :signed, digest: "SHA1"
  end

  config.filter_parameters << :password

  config.action_dispatch.perform_deep_munge = true

  config.action_dispatch.default_headers = {
    'X-Frame-Options' => 'SAMEORIGIN',
    'X-XSS-Protection' => '0',
    'X-Content-Type-Options' => 'nosniff',
    'X-Permitted-Cross-Domain-Policies' => 'none',
    'Referrer-Policy' => 'strict-origin-when-cross-origin'
  }

end
```

config/environments/development.rb

```ruby
require "active_support/core_ext/integer/time"

Rails.application.configure do
  # Settings specified here will take precedence over those in config/application.rb.

  # In the development environment your application's code is reloaded any time
  # it changes. This slows down response time but is perfect for development
  # since you don't have to restart the web server when you make code changes.
  config.enable_reloading = true

  # Do not eager load code on boot.
  config.eager_load = false

  # Show full error reports.
  config.consider_all_requests_local = true

  # Enable server timing
  config.server_timing = true

  # Enable/disable caching. By default caching is disabled.
  # Run rails dev:cache to toggle caching.
  if Rails.root.join("tmp/caching-dev.txt").exist?
    config.action_controller.perform_caching = true
    config.action_controller.enable_fragment_cache_logging = true

    config.cache_store = :memory_store
    config.public_file_server.headers = {
      "Cache-Control" => "public, max-age=#{2.days.to_i}"
    }
  else
    config.action_controller.perform_caching = false

    config.cache_store = :null_store
  end

  # Store uploaded files on the local file system (see config/storage.yml for options).
  config.active_storage.service = :local

  # Don't care if the mailer can't send.
  config.action_mailer.raise_delivery_errors = false

  config.action_mailer.perform_caching = false

  # Print deprecation notices to the Rails logger.
  config.active_support.deprecation = :log

  # Raise exceptions for disallowed deprecations.
  config.active_support.disallowed_deprecation = :raise

  # Tell Active Support which deprecation messages to disallow.
  config.active_support.disallowed_deprecation_warnings = []

  # Raise an error on page load if there are pending migrations.
  config.active_record.migration_error = :page_load

  # Highlight code that triggered database queries in logs.
  config.active_record.verbose_query_logs = true

  # Highlight code that enqueued background job in logs.
  config.active_job.verbose_enqueue_logs = true

  # Suppress logger output for asset requests.
  config.assets.quiet = true

  # Raises error for missing translations.
  # config.i18n.raise_on_missing_translations = true

  # Annotate rendered view with file names.
  # config.action_view.annotate_rendered_view_with_filenames = true

  # Uncomment if you wish to allow Action Cable access from any origin.
  # config.action_cable.disable_request_forgery_protection = true

  # Raise error when a before_action's only/except options reference missing actions
  config.action_controller.raise_on_missing_callback_actions = true

  config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }

  config.action_dispatch.signed_cookie_digest = "SHA256"

  config.action_dispatch.cookies_rotations.tap do |cookies|
    cookies.rotate :signed, digest: "SHA1"
  end

  config.filter_parameters << :password

  config.action_dispatch.perform_deep_munge = true

  config.action_dispatch.default_headers = {
    'X-Frame-Options' => 'SAMEORIGIN',
    'X-XSS-Protection' => '0',
    'X-Content-Type-Options' => 'nosniff',
    'X-Permitted-Cross-Domain-Policies' => 'none',
    'Referrer-Policy' => 'strict-origin-when-cross-origin'
  }

end
```

config/initializers/content_security_policy.rb

```
# Be sure to restart your server when you modify this file.

# Define an application-wide content security policy.
# See the Securing Rails Applications Guide for more information:
# https://guides.rubyonrails.org/security.html#content-security-policy-
header

 Rails.application.configure do
   config.content_security_policy do |policy|
     policy.default_src :self, :https
     policy.font_src    :self, :https, :data
     policy.img_src     :self, :https, :data
     policy.object_src  :none
     policy.script_src  :self, :https
     policy.style_src   :self, :https
     # Specify URI for violation reports
     policy.report_uri "/csp-violation-report-endpoint"
   end

   # Generate session nonces for permitted importmap, inline scripts,
and inline styles.
#    config.content_security_policy_nonce_generator = ->(request) {
request.session.id.to_s }
#    config.content_security_policy_nonce_directives = %w(script-src
style-src)

   # Report violations without enforcing the policy.
    config.content_security_policy_report_only = true
end
```

config/application.rb

```
require_relative "boot"

require "rails/all"

# Require the gems listed in Gemfile, including any gems
# you've limited to :test, :development, or :production.
Bundler.require(*Rails.groups)

module Project
  class Application < Rails::Application
    # Initialize configuration defaults for originally generated Rails version.
    config.load_defaults 7.1

    # Please, add to the `ignore` list any other `lib` subdirectories that do
    # not contain `.rb` files, or that should not be reloaded or eager loaded.
    # Common ones are `templates`, `generators`, or `middleware`, for example.
    config.autoload_lib(ignore: %w(assets tasks))

    # Configuration for the application, engines, and railties goes here.
    #
    # These settings can be overridden in specific environments using the files
    # in config/environments, which are processed later.
    #
    # config.time_zone = "Central Time (US & Canada)"
    # config.eager_load_paths << Rails.root.join("extras")

    # config.force_ssl = true
    config.filter_parameters << :password

    config.action_dispatch.signed_cookie_digest = "SHA256"

    config.action_dispatch.cookies_rotations.tap do |cookies|
      cookies.rotate :signed, digest: "SHA1"
    end

    config.action_dispatch.perform_deep_munge = true

    config.action_dispatch.default_headers = {
      'X-Frame-Options' => 'SAMEORIGIN',
      'X-XSS-Protection' => '0',
      'X-Content-Type-Options' => 'nosniff',
      'X-Permitted-Cross-Domain-Policies' => 'none',
      'Referrer-Policy' => 'strict-origin-when-cross-origin'
    }

  end
end
```

config/database.yml

```
default: &default
  adapter: mysql2
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  timeout: 5000
  database: bookshop
  username: root
  password: root
  host: localhost
  port: 3306

development:
  <<: *default
  database: bookshop

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.

production:
  <<: *default
  database: bookshop
```

config/routes.rb

```
Rails.application.routes.draw do
  devise_for :users
  devise_scope :user do
    get '/users/sign_out' => 'devise/sessions#destroy'
  end

  #authenticated :user do
    resources :ratings
    resources :paymentmethods
    resources :ordersdetails
    resources :orders
    resources :genres
    resources :employees
    resources :customers
    resources :books
    resources :authors
  #end

  # get 'home/index'
  get 'home/about'

  root 'home#index'
end
```

db/schema.rb

```ruby
# This file is auto-generated from the current state of the database. Instead of editing this file, please use the migrations feature of Active
Record to incrementally modify your database, and then regenerate this schema definition.
# This file is the source Rails uses to define your schema when running `bin/rails

ActiveRecord::Schema[7.1].define(version: 2023_12_22_191946) do
  create_table "authors", primary_key: "author_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 70, null: false
    t.string "nationality", limit: 45, null: false
    t.index ["author_id"], name: "author_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
  end

  create_table "books", primary_key: "book_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "title", limit: 100, null: false
    t.decimal "price", precision: 53, scale: 2, null: false
    t.integer "author_id", null: false
    t.integer "genre_id", null: false
    t.index ["author_id"], name: "author_id"
    t.index ["book_id"], name: "book_id_UNIQUE", unique: true
    t.index ["genre_id"], name: "genre_id_idx"
  end

  create_table "customers", primary_key: "customer_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 50, null: false
    t.string "phone_number", limit: 15
    t.index ["customer_id"], name: "customer_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
  end

  create_table "database_structures", charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
  end

  create_table "employees", primary_key: "employee_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "name", limit: 50, null: false
    t.integer "age", null: false
    t.string "phone_number", limit: 25, null: false
    t.index ["employee_id"], name: "employee_id_UNIQUE", unique: true
    t.index ["name"], name: "name_UNIQUE", unique: true
    t.check_constraint "(`age` > 15) and (`age` < 71)", name: "employees_chk_1"
  end

  create_table "genres", primary_key: "genre_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "genre_name", limit: 50, null: false
    t.index ["genre_id"], name: "genre_id_UNIQUE", unique: true
    t.index ["genre_name"], name: "genre_name_UNIQUE", unique: true
  end

  create_table "orders", primary_key: "order_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.integer "payment_method_id", null: false
    t.integer "employee_id", null: false
    t.integer "customer_id", null: false
    t.index ["customer_id"], name: "customer_id_idx"
    t.index ["employee_id"], name: "employee_id_idx"
    t.index ["order_id"], name: "order_id_UNIQUE", unique: true
    t.index ["payment_method_id"], name: "payment_method_id_idx"
  end

  create_table "ordersdetails", primary_key: "ordersDetails_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force:
:cascade do |t|
    t.integer "order_id", null: false
    t.integer "book_id", null: false
    t.index ["book_id"], name: "book_id_idx"
    t.index ["order_id"], name: "order_id_idx"
    t.index ["ordersDetails_id"], name: "ordersDetails_id_UNIQUE", unique: true
  end

  create_table "paymentmethods", primary_key: "paymentMethod_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force:
:cascade do |t|
    t.string "name", limit: 15, null: false
    t.index ["name"], name: "name_UNIQUE", unique: true
    t.index ["paymentMethod_id"], name: "paymentMethod_id_UNIQUE", unique: true
  end

  create_table "ratings", primary_key: "review_id", id: :integer, charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.integer "book_id", null: false
    t.string "comments", limit: 500
    t.integer "number_of_stars", null: false
    t.index ["book_id"], name: "book_id_idx"
    t.index ["review_id"], name: "review_id_UNIQUE", unique: true
    t.check_constraint "(`number_of_stars` > 0) and (`number_of_stars` < 6)", name: "ratings_chk_1"
  end

  create_table "users", charset: "utf8mb4", collation: "utf8mb4_0900_ai_ci", force: :cascade do |t|
    t.string "email", default: "", null: false
    t.string "encrypted_password", default: "", null: false
    t.string "reset_password_token"
    t.datetime "reset_password_sent_at"
    t.datetime "remember_created_at"
    t.datetime "created_at", null: false
    t.datetime "updated_at", null: false
    t.index ["email"], name: "index_users_on_email", unique: true
    t.index ["reset_password_token"], name: "index_users_on_reset_password_token", unique: true
  end

end
```

Gemfile

```
source "https://rubygems.org"

ruby "3.2.2"

gem "rails", "~> 7.1.1"
gem "sprockets-rails"
gem "yaml_db"
gem "schema_to_scaffold"
gem "puma", ">= 5.0"
gem "importmap-rails"
gem "turbo-rails"
gem "stimulus-rails"
gem "jbuilder"
gem "tzinfo-data", platforms: %i[ windows jruby ]
gem "bootsnap", require: false
gem 'devise', '~> 4.9', '>= 4.9.3'
gem 'jquery-rails', '~> 4.6'
gem 'jquery-ui-rails', '~> 6.0', '>= 6.0.1'
gem "mysql2"

group :development, :test do
  gem "debug", platforms: %i[ mri windows ]
end

group :development do
  gem "mysql2"
  gem "web-console"
end

group :production do
  gem 'pg', '~> 1.5', '>= 1.5.4'
end

group :test do
  gem "capybara"
  gem "selenium-webdriver"
end
```

4) REFERENCES

- [1] “Securing Rails Applications” [Online] https://guides.rubyonrails.org/security.html#permitted-lists-versus-restricted-lists
- [2] Le Wagon, “How to build a secure web applications with Ruby on Rails” [Online] https://www.youtube.com/watch?v=_-Dnys7_ESE
- [3] freeCodeCamp.org, “Learn Ruby on Rails - Full Course” [Online] https://www.youtube.com/watch?v=fmyvWz5TUWg
- [4] “Get started with Bootstrap” [Online] https://getbootstrap.com/docs/5.3/getting-started/introduction/
- [5] Janos Rusiczki, “Starting a Ruby on Rails project from existing data” [Online] https://medium.com/@kitsched/starting-a-ruby-on-rails-project-from-existing-data-7dda5044c85f
- [6] Armando Fox and David Patterson, “Engineering Software as a Service: An Agile Approach Using Cloud Computing Second Edition, 2.0b7” [Book]
- [7] ChatGPT [Online] https://chat.openai.com/
- [8] Visual Paradigm (for creating diagrams) [Online] https://online.visual-paradigm.com/