Moving App

T2A2 - Marketplace Project

Table of Contents

Problem & Solution

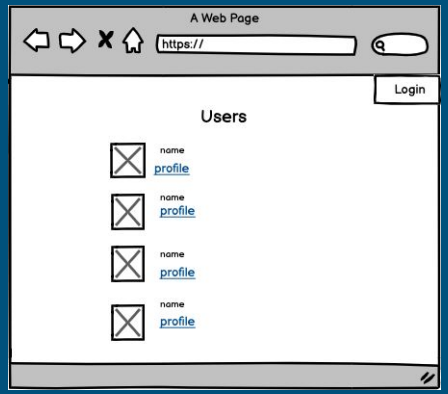
Moving Nightmares!

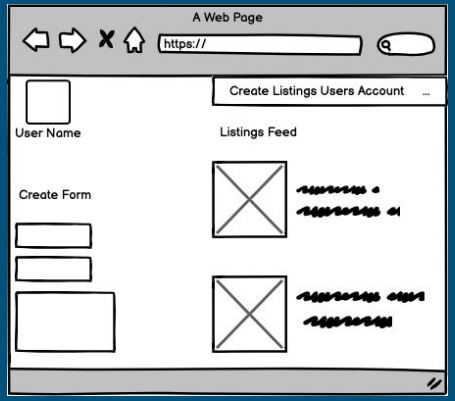
- Packing Boxes
- Shoppers Remorse
- Expensive Moving Companies
- Lifting Boxes

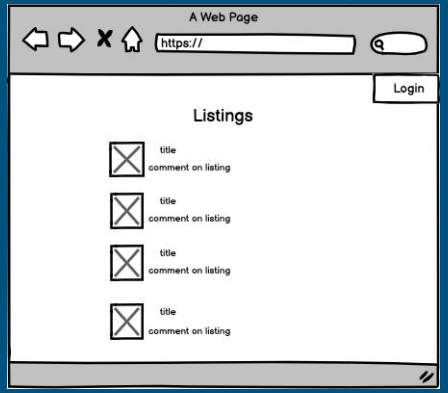
Solution!

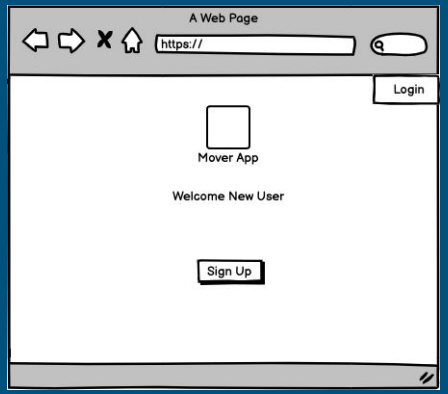
Create an online community of others willing to help move, pack or even share supplies, helps solve this problem.

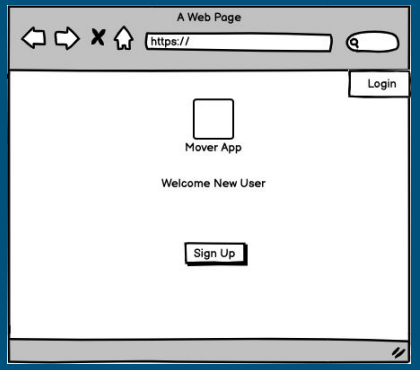
App Planning



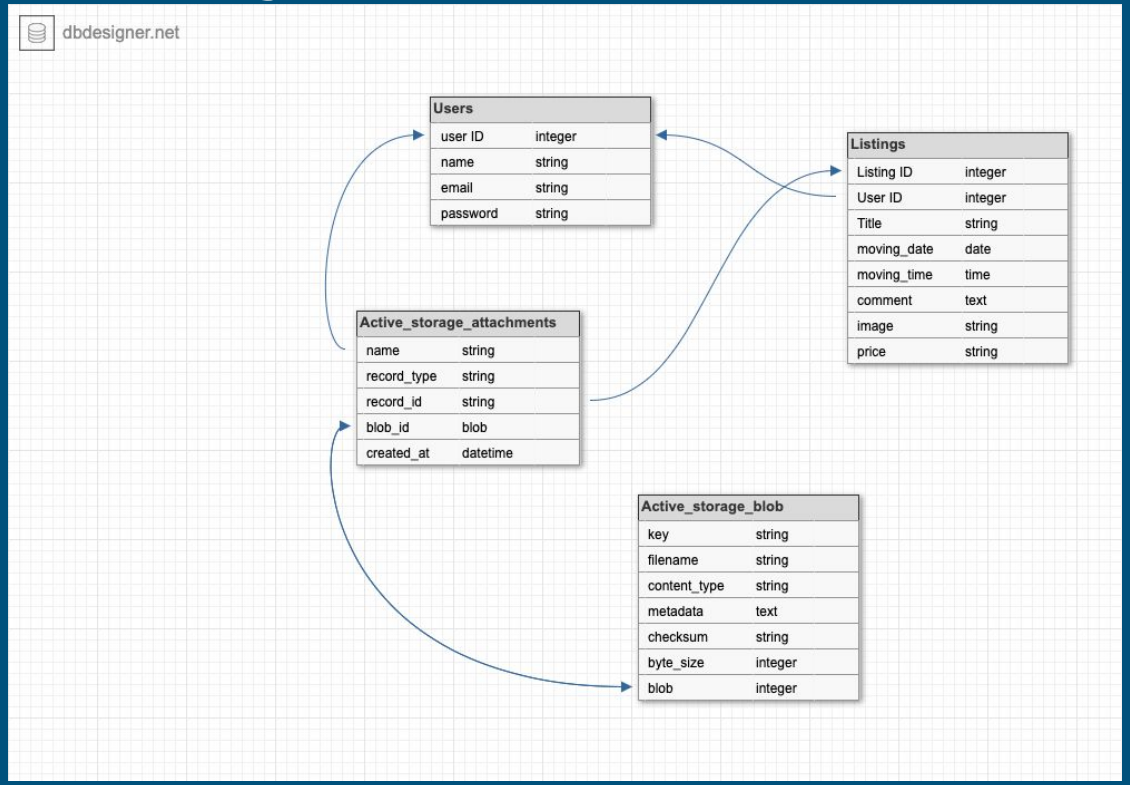




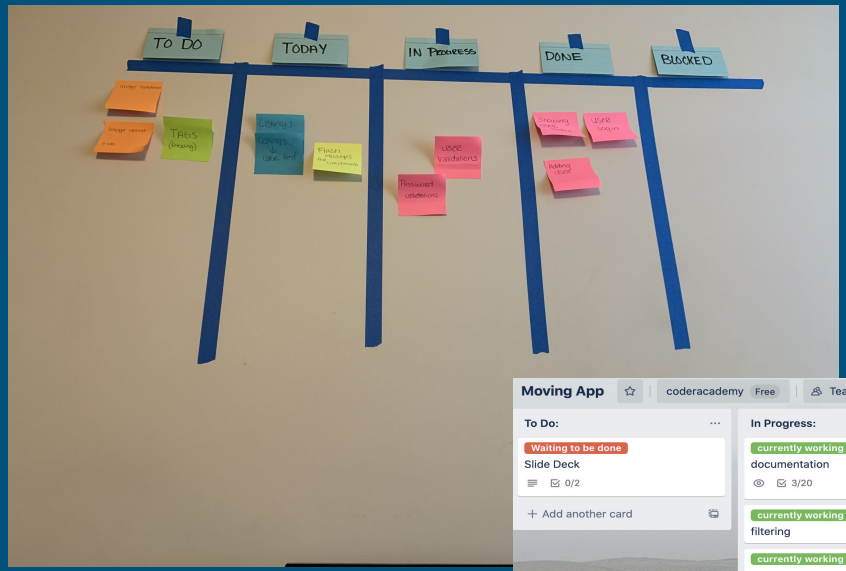


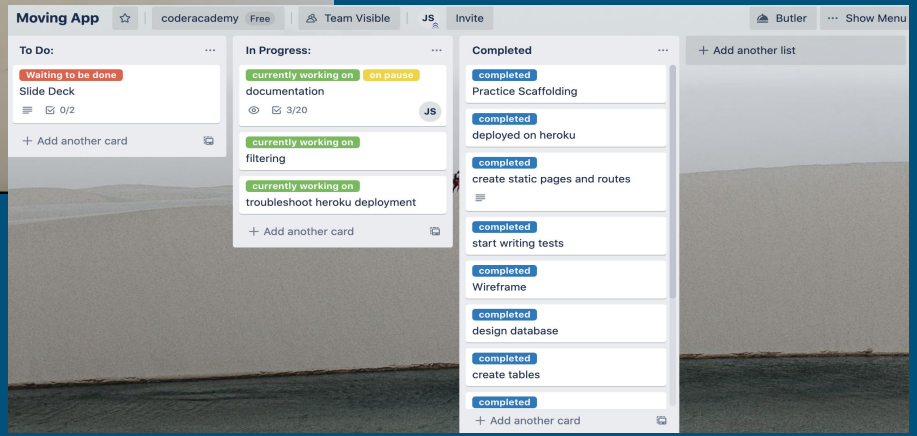


App Planning (cont)

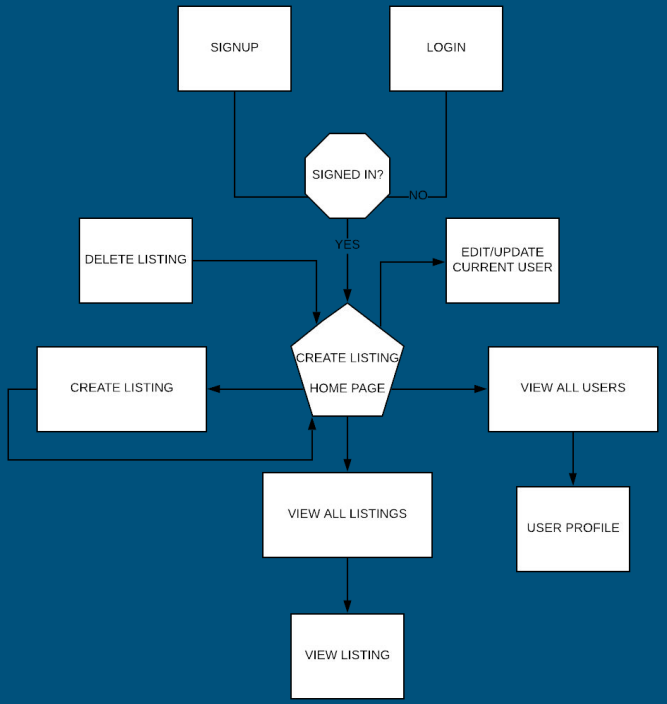


App Planning (cont)





Sitemap


SIGNUP
LOGIN
SIGNED IN?
NO
YES
DELETE LISTING
EDIT/UPDATE CURRENT USER
CREATE LISTING
HOME PAGE
CREATE LISTING
VIEW ALL USERS
VIEW ALL LISTINGS
USER PROFILE
VIEW LISTING

Testing

Checking Routes

```ruby
require 'test_helper'

class StaticPagesControllerTest < ActionDispatch::IntegrationTest
  test "should get root" do
    get root_path
    assert_response :success
  end

  test "should get home" do
    get home_path
    assert_response :success
  end

  test "should get help" do
    get help_path
    assert_response :success
  end

  test " should get about" do
    get about_path
    assert_response :success
  end

  test "should get contact" do
    get contact_path
    assert_response :success
  end

end
```

User Login

```ruby
require 'test_helper'

class UsersLoginTest < ActionDispatch::IntegrationTest

  test "login with invalid information" do
    get login_path
    assert_template 'sessions/new'
    post login_path, params: {session: {email: "", password: ""} }
    assert_template 'sessions/new'
    assert_not flash.empty?
    get root_path
    assert flash.empty?
  end
end
```

Testing (cont)

User Tests

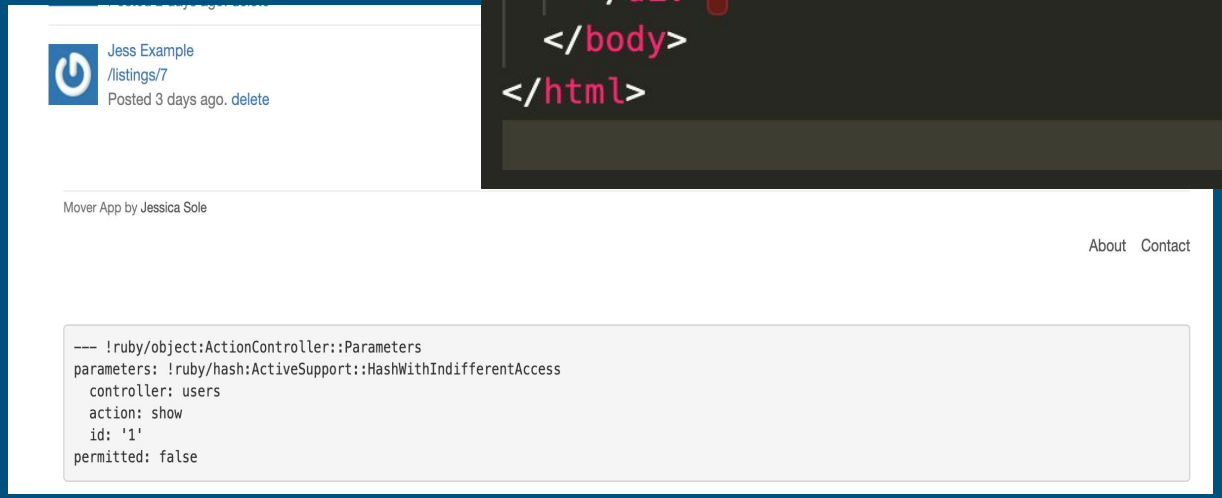
```
require 'test_helper'
class UserTest < ActiveSupport::TestCase

  def setup
    @user = User.new(name: "Example User", email: "user@example.com",
    password: "foobar", password_confirmation: "foobar")
  end

  test "name should be present" do
    @user.name = "       "
    assert_not @user.valid?
    end

    test "email should be present" do
    @user.email = "       "
    assert_not @user.valid?
    end

    test "name should not be too long" do
    @user.name = "a" * 51
    assert_not @user.valid?
    end

    test "email should not be too long" do
    @user.email = "a" * 244 + "@example.com"
    assert_not @user.valid?
    end
```

User Tests

```
  test "email validation should accept valid addresses" do
    valid_addresses = %w[user@example.com USER@foo.com A_US-ER@foo.ber.org
                         first.last@foo.jp alice+bob@baz.cn]
    valid_addresses.each do |valid_address|
      @user.email = valid_address
      assert @user.valid?, "#{valid_address.inspect} should be valid"
    end
  end

  test "email validation should reject invalid addresses" do
    invalid_addresses = %w[user@example,com user_at_foo.org user.name@example.
                         foo@bar_baz.com foo@bar+baz.com]
    invalid_addresses.each do |invalid_address|
      @user.email = invalid_address
      assert_not @user.valid?, "#{invalid_address.inspect} should be invalid"
    end
  end

  test "password should be present (nonblank)" do
    @user.password = @user.password_confirmation = " " * 6
    assert_not @user.valid?
  end

  test "password should have a minimum length" do
    @user.password = @user.password_confirmation = "a" * 5
    assert_not @user.valid?
  end

  test "associated listings should be destroyed" do
    @user.save
    @user.listings.create!(comment: "Lorem ipsum")
    assert_difference 'Listing.count', -1 do
    @user.destroy
    end
  end
end
```

Debugging

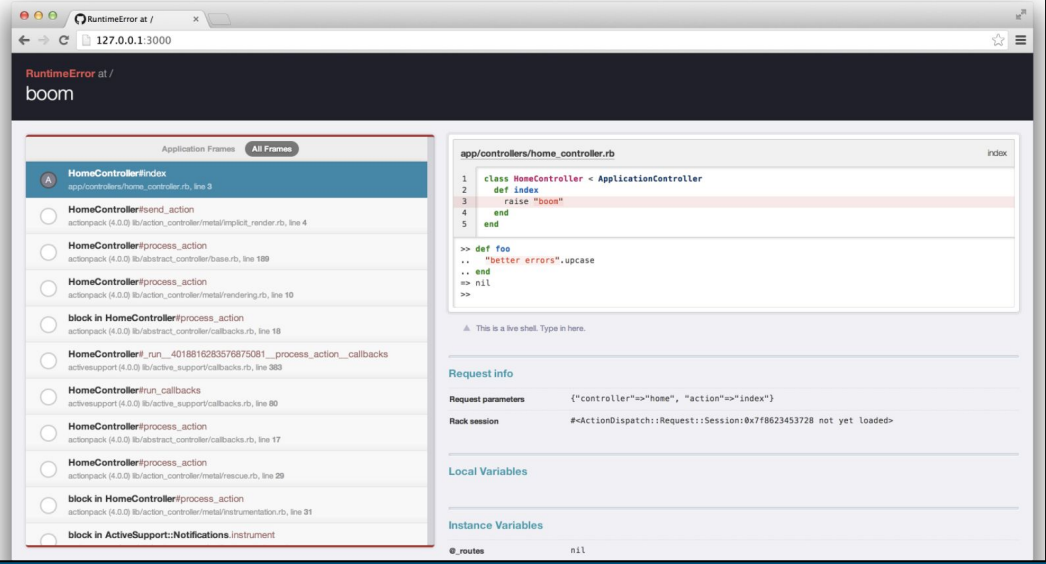
```
        <%= yield %>
        <%= render 'layouts/footer'%>
        <%# <%= debug(params) if Rails.env.development? %>
      </div>
    </body>
</html>
```



Gems Used

Better Errors

Better Errors replaces the standard Rails error page with a much better and more useful error page. It is also usab
outside of Rails in any Rack app as Rack middleware.



Gems Used (cont)

README.md

MiniMagick

gem v4.10.1 | downloads 52M | build passing | maintainability A

A ruby wrapper for ImageMagick or GraphicsMagick command line.

README.md

bcrypt-ruby

An easy way to keep your users' passwords secure.

- https://github.com/codahale/bcrypt-ruby/tree/master

build passing | build passing

Bootstrap Ruby Gem

build passing | gem v4.5.2

Bootstrap 4 ruby gem for Ruby on Rails (Sprockets) and Hanami (formerly Lotus).

For Sass versions of Bootstrap 3 and 2 see bootstrap-sass instead.

Faker

Authorisations & Validations

```
test "name should be present" do
  @user.name = "       "
  assert_not @user.valid?
  end

  test "email should be present" do
  @user.email = "       "
  assert_not @user.valid?
  end

  test "name should not be too long" do
  @user.name = "a" * 51
  assert_not @user.valid?
  end
```

```
  def user_params
    params.require(:user).permit(:name, :email, :password, :password_confirmation)
  end

  #confirm a logged-in user
  def logged_in_user
    unless logged_in?
      store_location
      flash[:danger] = "please log in"
      redirect_to login_url
    end
  end

  #confirms the correct user
  def correct_user
    @user = User.find(params[:id])
    redirect_to(root_url) unless current_user?(@user)
  end

end
```