RACK::TEST IN SINATRA

TESTING REQUESTS/ RESPONSES WITHOUT HTML

REVIEW: UNIT TESTS

▸ **Action:** method calls

▸ **Result:** return values/object state

▸ **Sample Test:**

```ruby
def test_assigns_attributes_correctly
  task = Task.new({ "title"       => "a title",
                    "description" => "a description",
                    "id"          => 1 })
  assert_equal "a title", task.title
  assert_equal "a description", task.description
  assert_equal 1, task.id
end
```

REVIEW: FEATURE TESTS

▸ Action: interacting with a webpage

▸ Result: a new webpage (HTML)

▸ Sample Test:

```ruby
 7    def test_task_creation_with_valid_attributes
 8      visit '/tasks/new'
 9
10      fill_in 'task[title]', with: 'Example Task'
11      fill_in 'task[description]', with: 'Example Description'
12      click_button 'Submit'
13
14      assert_equal '/tasks', current_path
15
16      within '.task' do
17        assert page.has_content? 'Example Task'
18      end
19    end
```

REVIEW: RUSH HOUR TESTS

▸ **Action:** Submitting curl requests

▸ **Result:** HTTP responses (status, headers, body)

▸ **Sample Test:** ?

RACK::TEST

▸ Ruby gem: 'rack-test'

▸ Gives us a way to test our controllers

▸ Mimics request/response cycle

▸ Doesn't require a view to be rendered for either a submissions or responses (no Capybara)

WHAT DOES RACK::TEST GIVE US?

▸ HTML verbs in our tests: get, post, put, patch, delete

  ▸ get '/'

▸ Ability to pass params hash as a second argument

  ▸ post '/', {title: "New Idea", description: "It's new"}

▸ last_response.status: returns status code of last response (200, 404, 500, etc.)

▸ last_response.body: returns body of last response - can test using string methods like include?, etc.

▸ follow_redirect!: follows any redirects in the response

SETUP

▸ In the Gemfile: gem 'rack-test'

▸ In the test:

  ▸ include Rack::Test::Methods

  ▸ def app
      AppName
    end

CODE ALONG: FILMFILE

INDEPENDENT WORK: TODO