**CS 3300 Software Engineering**

**Iteration 4**

Purpose: Learn about deploying to Heroku and implementing Unit Testing with RSpec so you can create your own tests in the next iteration.

Effort:

* Group - Set up the environment .
* Individual - Answer questions on your own.

Points: **40**

Deliverables: Document

**Assignment Description**

In this homework assignment you are going to explore Heroku and RSpec then update your individual software project.You will set up Heroku and begin to implement individual unit tests for your application.

**Part 1: Heroku Exploration and Set-up**

1. Complete [HW06 Part 1: Heroku set up](https://docs.google.com/document/d/1DQT8F0pRs7U6zv2jCRrc_LMBq3nKVe8OGswPdKMIWMA/edit#)
2. Read [Heroku Architecture: How Heroku Works](https://devcenter.heroku.com/articles/how-heroku-works) sections
   1. Defining an application
   2. Knowing what to execute
   3. Deploying applications

**Part 2: Testing Rails with RSpec**

Read <https://github.com/debmhteach/CS3300HW06Rspec/blob/main/README.md> read resources while completing each section below

1. Install Rspec with Rails and read [Understanding Test-Driven Development with RSpec in Ruby on Rails](https://www.microverse.org/blog/test-driven-development-with-rspec-in-ruby-on-rails)
2. Implement Tests and Add Comments to your RSpec Code
   1. Your first spec
   * [The Definitive RSpec Tutorial With Examples](https://www.rubyguides.com/2018/07/rspec-tutorial/)
3. Create a Unit Test for Project Model, read the following and add comments in your code
   * [Model specs - RSpec Rails - RSpec](https://relishapp.com/rspec/rspec-rails/docs/model-specs)
   * [Module ActiveRecord::Validations::ClassMethods](https://api.rubyonrails.org/v6.1.3.1/classes/ActiveRecord/Validations/ClassMethods.html)
   * [Ruby on Rails - User Input Validations](https://www.tutorialspoint.com/ruby-on-rails/rails-input-validations.htm)
   * [before and after hooks - Hooks - RSpec Core - RSpec](https://relishapp.com/rspec/rspec-core/v/2-2/docs/hooks/before-and-after-hooks)
4. Create functional test for Projects controller and integration spec for Projects and read [Controller specs - RSpec Rails - RSpec](https://relishapp.com/rspec/rspec-rails/docs/controller-specs)
5. Create integration specs for
   * Home Page and Projects
   * Read [Generators - RSpec Rails - RSpec](https://relishapp.com/rspec/rspec-rails/docs/generators) and read [Feature spec - Feature specs - RSpec Rails - RSpec](https://relishapp.com/rspec/rspec-rails/docs/feature-specs/feature-spec)
   * Open Gemfile and notice gem 'capybara', '>= 2.15', and Read [Using Capybara with RSpec](https://rubydoc.info/github/teamcapybara/capybara/master#using-capybara-with-rspec)
6. Add Test Coverage
   1. Add simplecov and read [SimpleCov](https://www.learnhowtoprogram.com/ruby-and-rails/authentication-and-authorization/simplecov)

**Part 3: Answer Questions and include any resources that helped.**

1. CMake sure your latest code you implemented from this iteration is pushed to Github and deployed to Heroku. Copy and paste your URLs to your Heroku app and your Github project. Share a screenshot of your code coverage results. (15pts)
2. Add comments to your Rspec code. (5)
3. Explain RSpec and how it relates to BDD and TDD. (5)
4. Explain each of the following code snippets. Include resources you used (15)

Code 1:

RSpec.describe ProjectsController, type: :controller do

context "GET #index" do

it "returns a success response" do

get :index

# expect(response.success).to eq(true)

expect(response).to be\_success

end

end

Code 2:

RSpec.feature "Projects", type: :feature do

context "Create new project" do

before(:each) do

visit new\_project\_path

within("form") do

fill\_in "Title", with: "Test title"

end

end

scenario "should be successful" do

fill\_in "Description", with: "Test description"

click\_button "Create Project"

expect(page).to have\_content("Project was successfully created")

end

scenario "should fail" do

click\_button "Create Project"

expect(page).to have\_content("Description can't be blank")

end

end