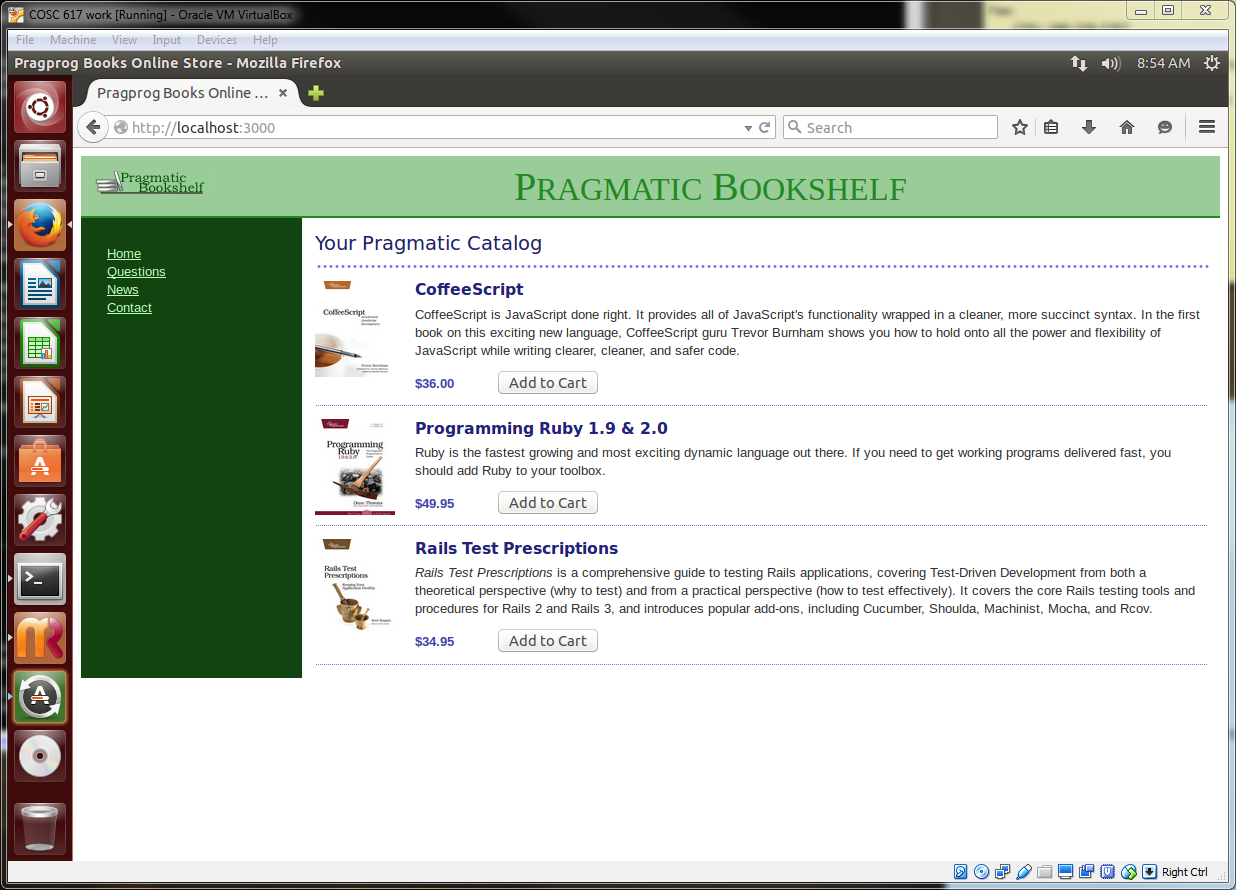
**Advanced Web Development – “Unit and Functional Testing Lab”**

Points: 25

Due: Night (11:59 pm) before next class.

1. Clone the project from <https://github.com/siddharthkaza/depot_p_for_lab.git> under the lab and import into the IDE.
   1. Run the application to make sure its running in the browser.

*Hint:* It’s always a good idea to run ‘bundle update and then ‘rake db:migrate’ and ‘rake db:seed’ first - when using an application from another location. This updates any gems needed and brings the DB to its newest version.

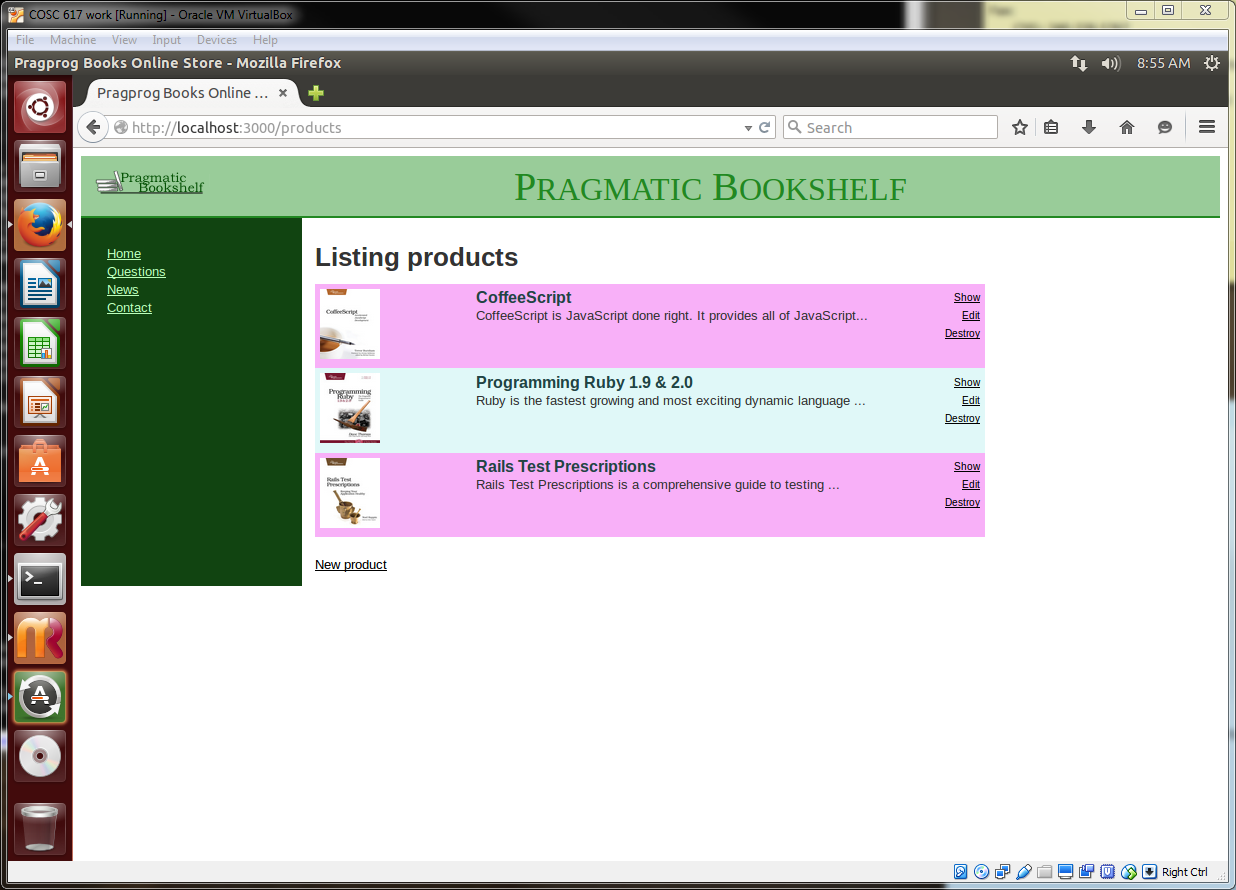


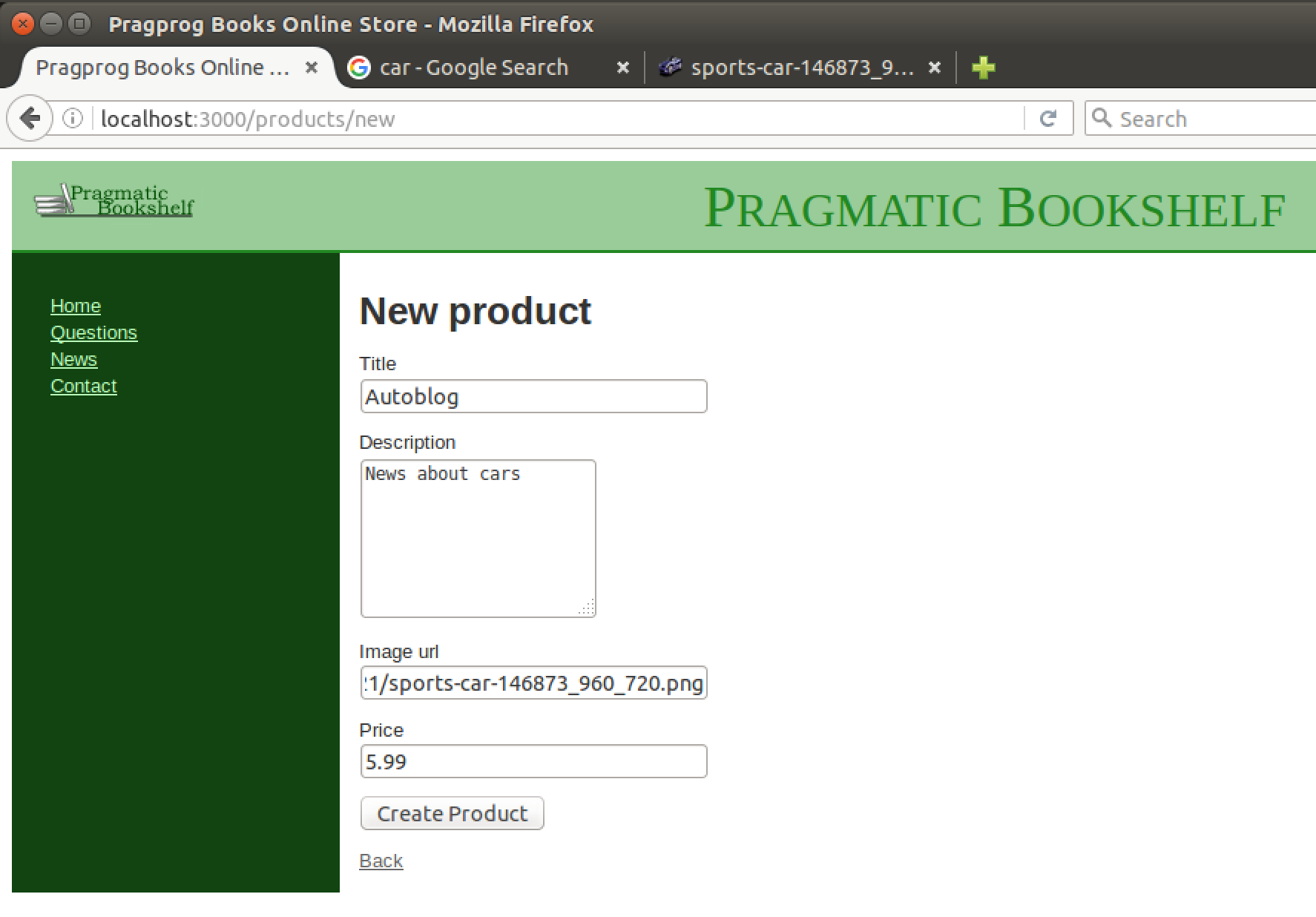
1. **Validations (5 points)**

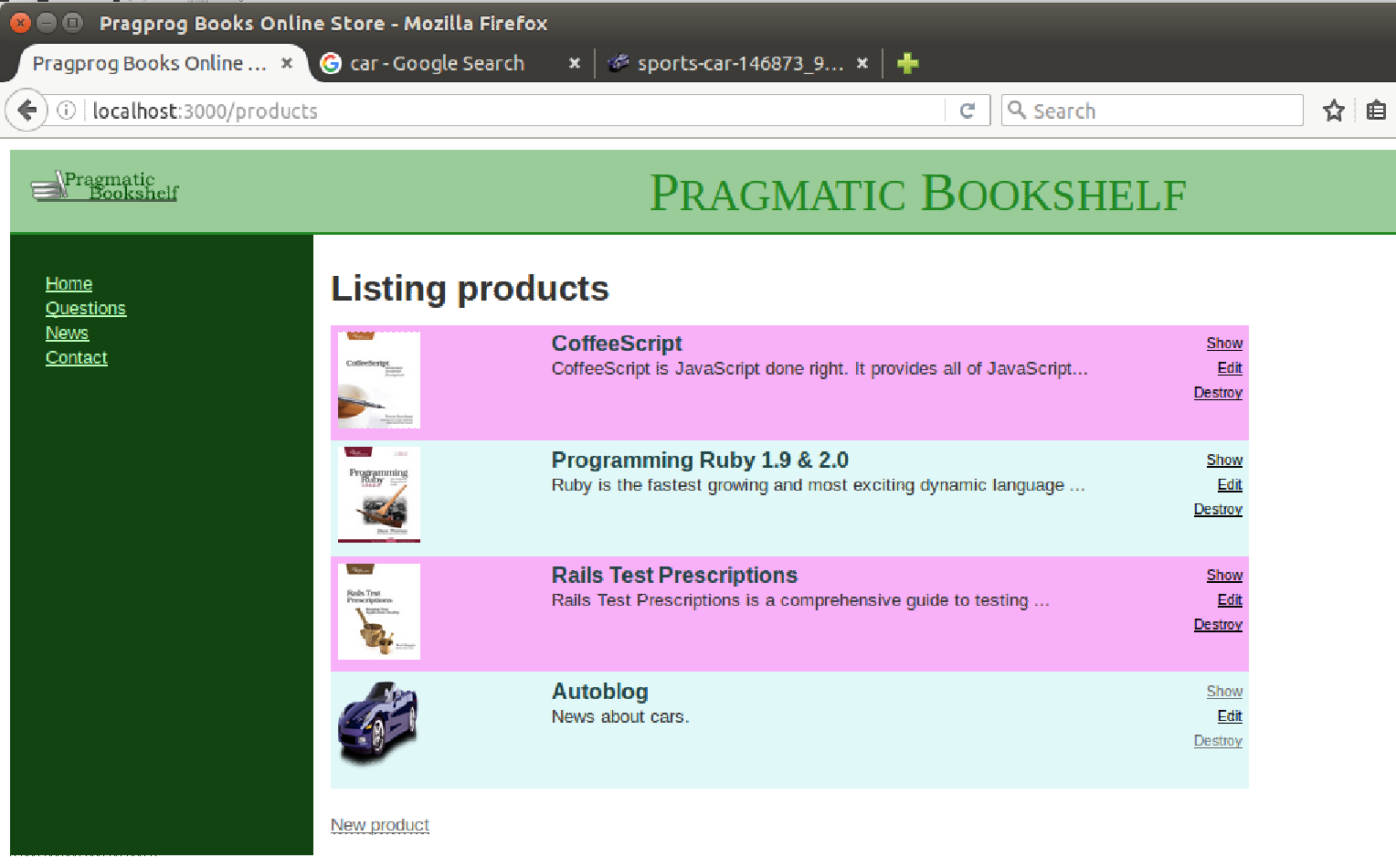
Validations are methods that are placed in the model to validate data before records are entered in the database.

In the depot application, we will put in validations in the product model to prevent bad data from being entered in the database (for instance, ‘a product should always have a title’).

1. Run the depot application, enter a new product in the database (go to /products in the URL to get to the products listing).







1. What validations do you think the product model needs? List them here.

**Validate the book has a title and it is unique**

**Validate the description is not empty**

**Validate the url is a picture**

**Validate the price is not zero or negative**

1. Enter the following statements in the product model (product.rb) one-by-one

validates :title, :description, :image\_url, presence: true

validates :price, numericality: {greater\_than\_or\_equal\_to: 0.01}

validates :title, uniqueness: true

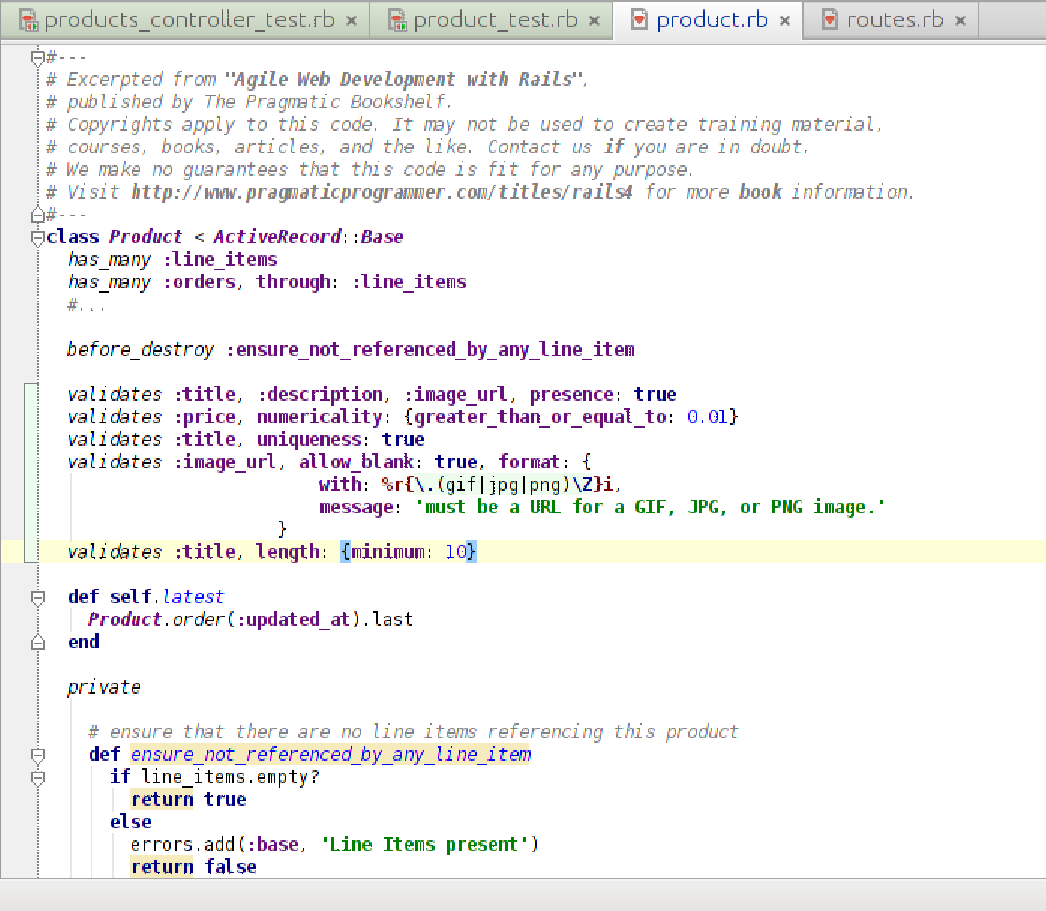
validates :image\_url, allow\_blank: true, format: {

with: %r{\.(gif|jpg|png)\Z}i,

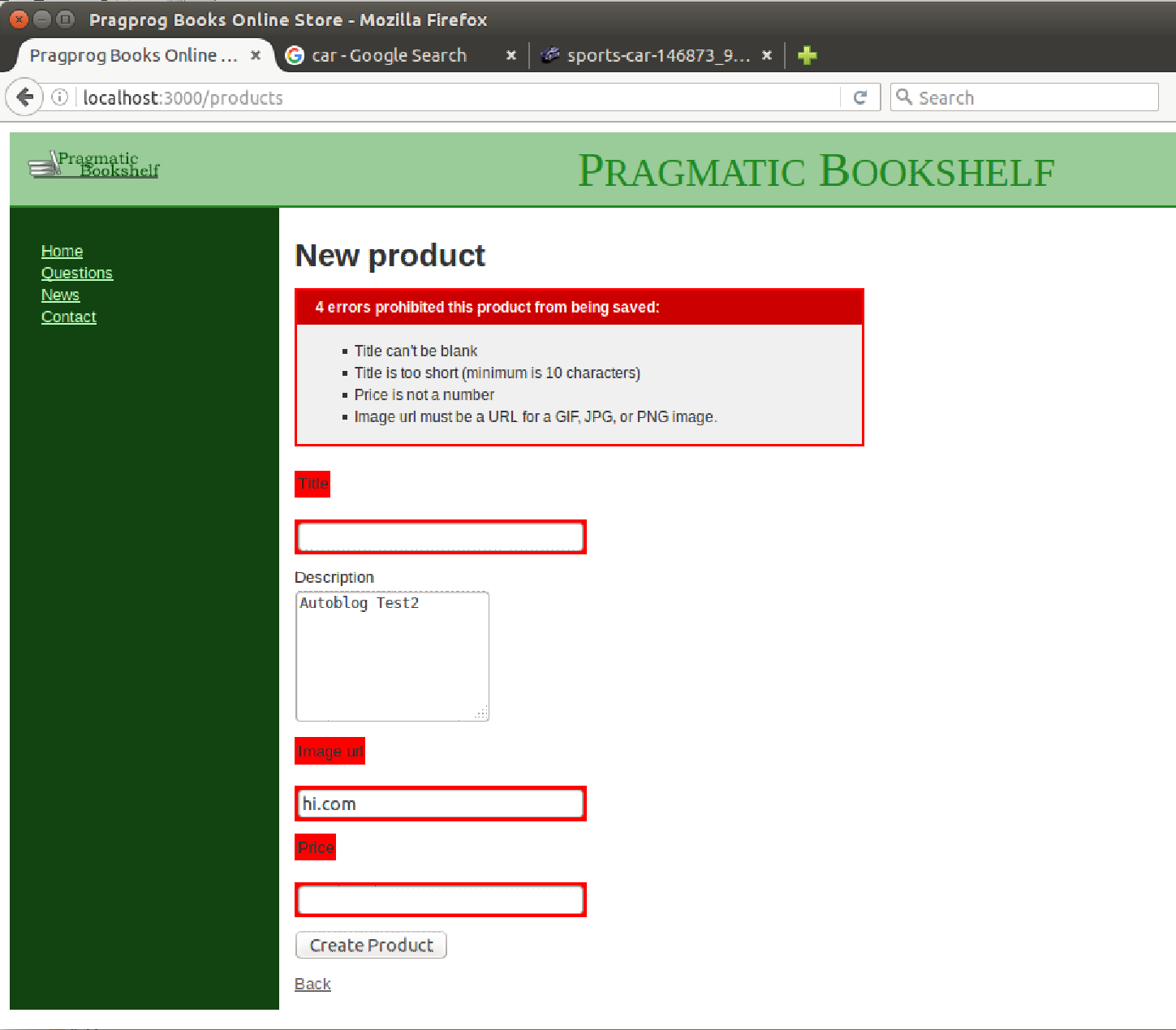
message: 'must be a URL for GIF, JPG or PNG image.'

}

validates :title, length: {minimum: 10}



1. Create another product that fails the validations (for e.g., leave the title empty). You notice that errors show up at the top of the page after you have. How does that work? What data structures are being used in the view to show the errors?

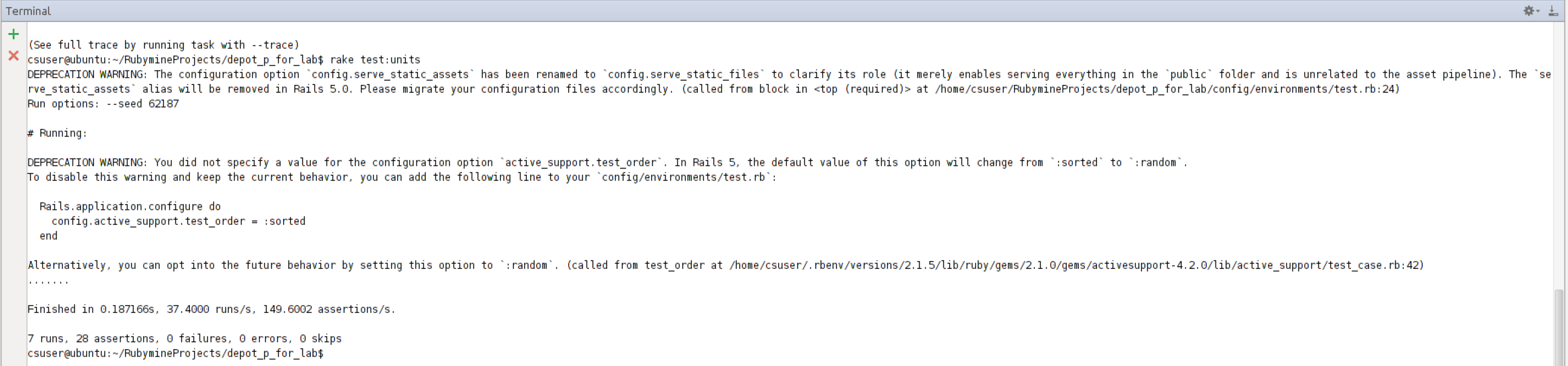


**Validations are run before any of these commands/data are sent to the database. Any validation errors are saved as part of the ActiveRecord object in an array called errors (products.errors in our case). If there are any validation failures, the INSERT or UPDATE is not performed.**

1. Are there any more validations that you want see in the products model? Think about how you would add them. Bring them up when we discuss the lab in the class.

**Validate the url is a valid address/site**

1. **Testing (10 points)**
   1. Study the model tests in the application (they are located under the ‘tests’ folder). Run those model tests and see the results. The command to run all the model tests (also known as ‘unit’ tests) in your application is ‘rake test:units’



Do you think the application needs more unit tests for products? Explain here.

**The title length is a minimum of 10**

**Combination tests where there is more than one error**

**Validity of the url (does it resolve)**

* 1. Study the controller tests in the application. Run the functional tests (rake test:functionals) and see the results.
     1. You will notice that some functional tests fail. Why do they fail? Study the assert\_difference method for one of them. When one of the test fails you get a message:

Product.count didn’t change by -1

<2> expected but was

<4>.

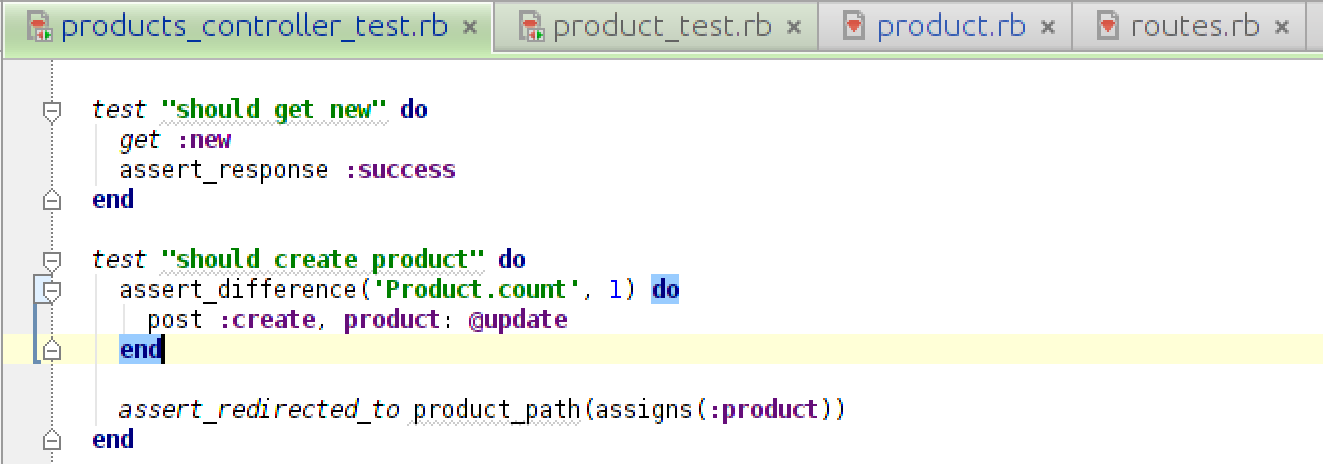
What do the numbers <2> and <4> mean?

**The number <2> was the expected result and <4> was the actual result. Therefore, in this case, the Product.count was not decreased as expected.**

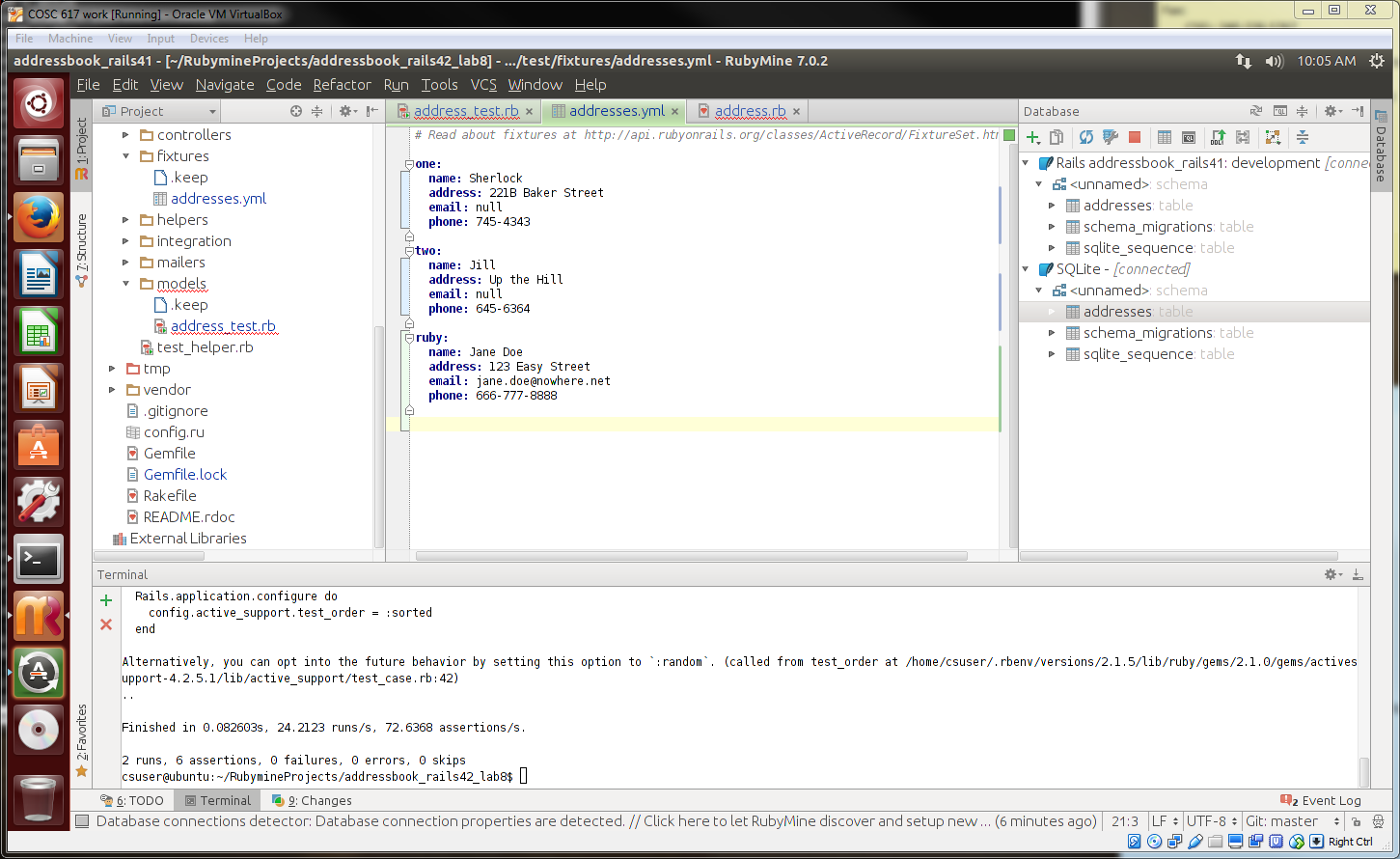
* + 1. Fix the code so that all failing tests pass.

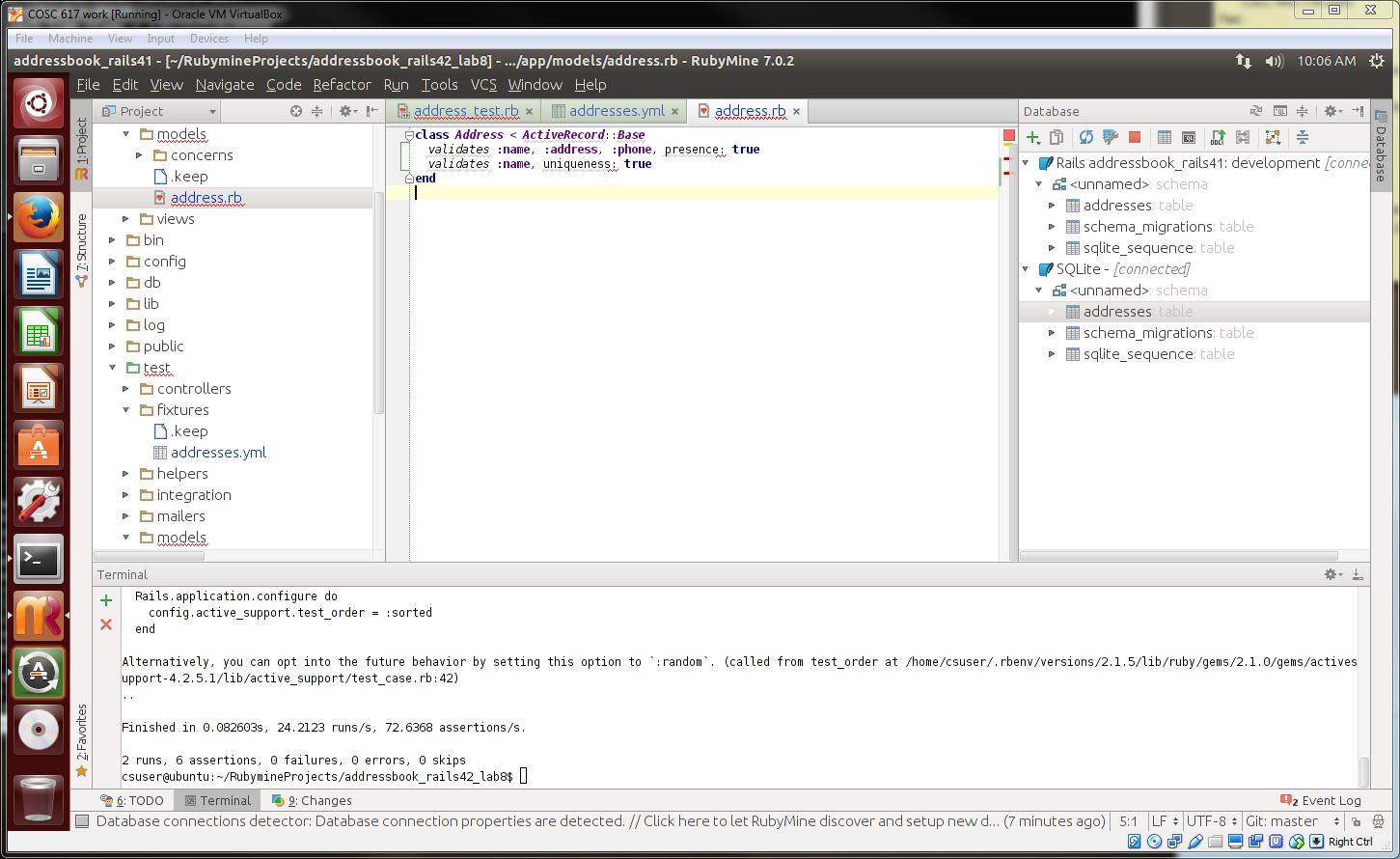
**Update “assert\_difference(‘Product.count’, -*1*) do” to**

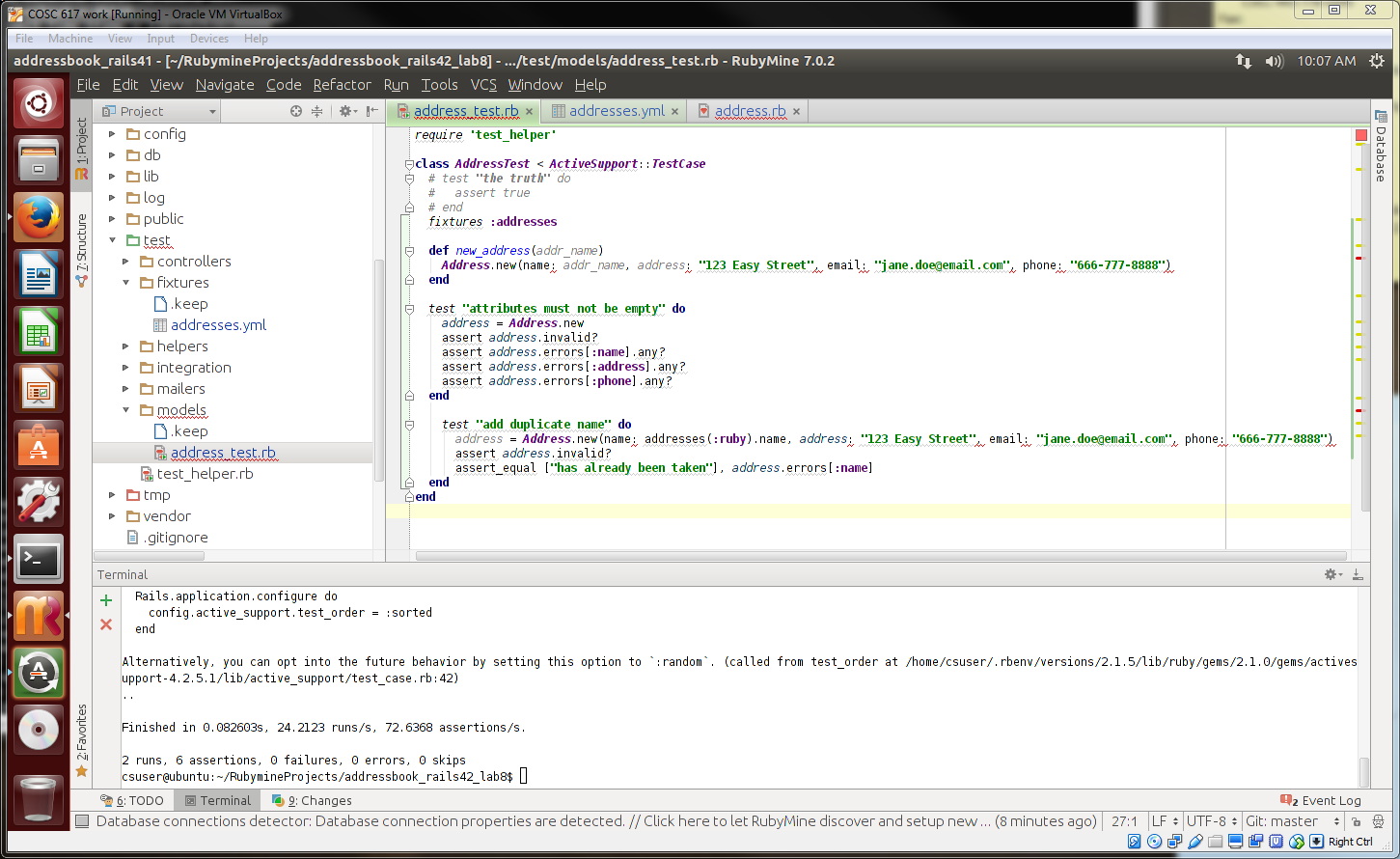
**“assert\_difference(‘Product.count’, *1*) do”**

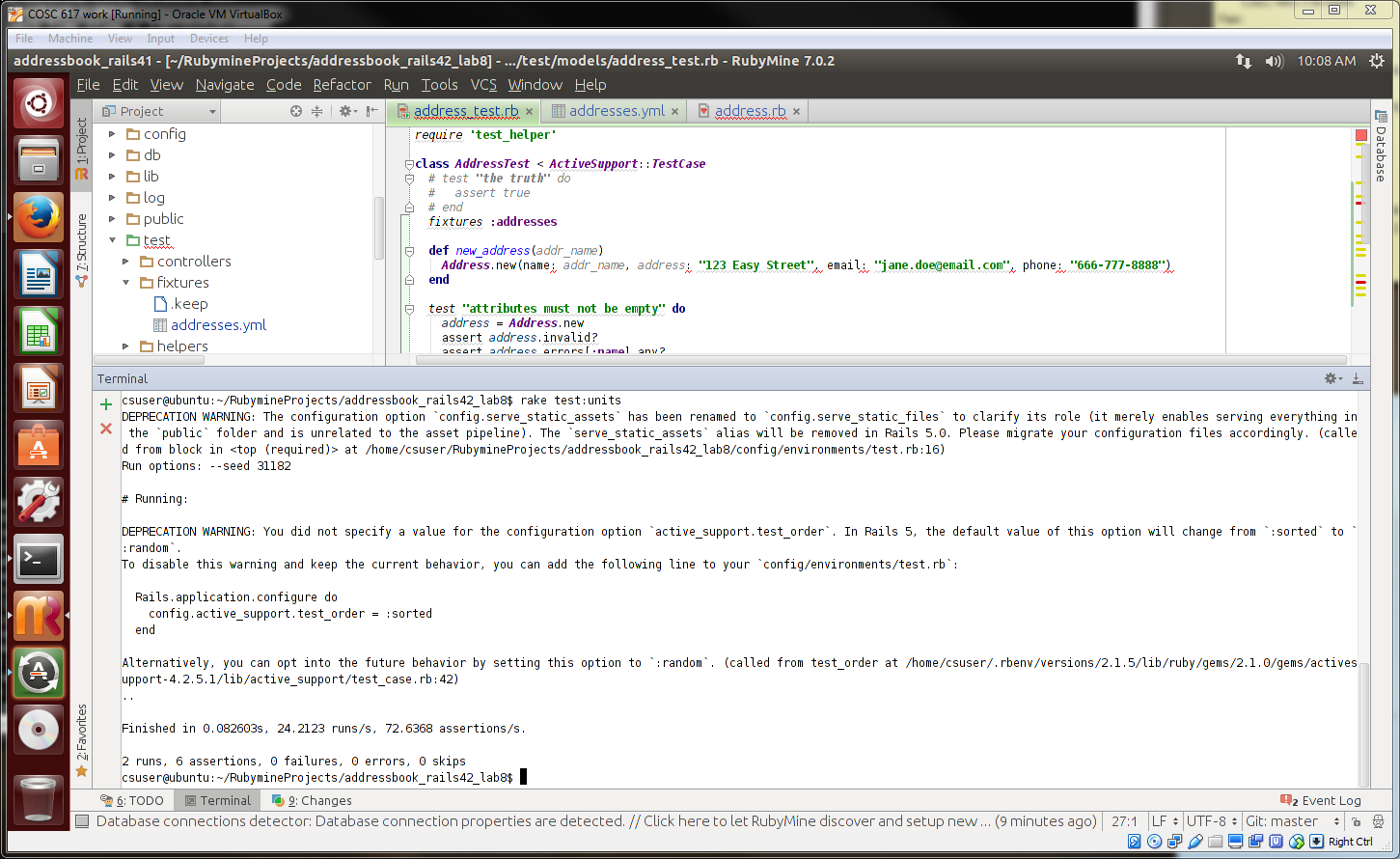
****

1. Download the addressbook application <https://github.com/siddharthkaza/addressbook_rails42>

Write two unit tests and two functional tests for the application. See the examples in depot\_p. You might have to write the validations in the model to write unit tests. (10 points)  
**Unit Tests**  
 







**Functional Tests**  
