```
describe Contact do
  it "is valid with a firstname, lastname and email" do
    contact = Contact.new(
      firstname: 'Aaron',
      lastname: 'Sumner',
      email: 'tester@example.com')
    expect(contact).to be_valid
  end
end                                    # spec/models/contact_spec.rb
```

```
describe Contact do
  it "is invalid without a firstname" do
    contact = Contact.new(firstname: nil)
    contact.valid?
    expect(contact.errors[:firstname]).to \
      include("can't be blank")
  end
end                                    # spec/models/contact_spec.rb
```

```
describe Contact do
  describe "filter last name by letter" do
    before :each do
      @smith =   Contact.create( firstname: 'John',
        lastname: 'Smith',   email: 'jsmith@example.com'
      )
      @jones =   Contact.create( firstname: 'Tim',
        lastname: 'Jones',   email: 'tjones@example.com'
      )
      @johnson = Contact.create( firstname: 'John',
        lastname: 'Johnson', email: 'jjohnson@example.com'
      )
    end
    context "with matching letters" do
      it "returns a sorted array of results that match" do
        expect(Contact.by_letter("J")).to eq [@johnson, @jones]
      end
    end
  end
end                                    # spec/models/contact_spec.rb
```

```
describe ContactsController do
  describe 'GET#index' do
    context 'with params[:letter]' do
      it "populates an array of contacts starting with the letter" do
        smith = create(:contact, lastname: 'Smith')
        jones = create(:contact, lastname: 'Jones')
        get '/contacts', letter: 'S'
        expect(response.body).to     include(smith.lastname)
        expect(response.body).not_to include(jones.lastname)
      end
      it "renders the :index template" do
        get '/contacts', letter: 'S'
        expect(response).to render_template :index
      end
    end
    context 'without params[:letter]' do
      it "populates an array of all contacts" do
        smith = create(:contact, lastname: 'Smith')
        jones = create(:contact, lastname: 'Jones')
        get '/contacts'
        expect(response.body).to include(smith.lastname)
        expect(response.body).to include(jones.lastname)
      end
      it "renders the :index template" do
        get '/contacts'
        expect(response).to render_template :index
      end
    end
  end
end                                        # spec/requests/contacts_management_spec.rb
```

```
feature 'Usermanagement' do
  scenario "adds a new user" do
    admin = create(:admin)
    visit root_path
    click_link 'Log In'
    fill_in 'Email', with: admin.email
    fill_in 'Password', with: admin.password
    click_button 'Log In'
    visit root_path
    expect {
      click_link 'Users'
      click_link 'New User'
      fill_in 'Email', with: 'newuser@example.com'
      find('#password').fill_in 'Password', with: 'secret123'
      find('#password_confirmation').fill_in 'Password confirmation',
        with: 'secret123'
      click_button 'Create User'
    }.to change(User, :count).by(1)
    expect(current_path).to eq users_path
    expect(page).to have_content 'New user created'
    within 'h1' do
      expect(page).to have_content 'Users'
    end
    expect(page).to have_content 'newuser@example.com'
  end
end                                         # spec/features/users_spec.rb

module LoginMacros
  def sign_in admin
    visit root_path
    click_link 'Log In'
    fill_in 'Email', with: admin.email
    fill_in 'Password', with: admin.password
    click_button 'Log In'
  end
end                          # spec/support/login_macros.rb
```