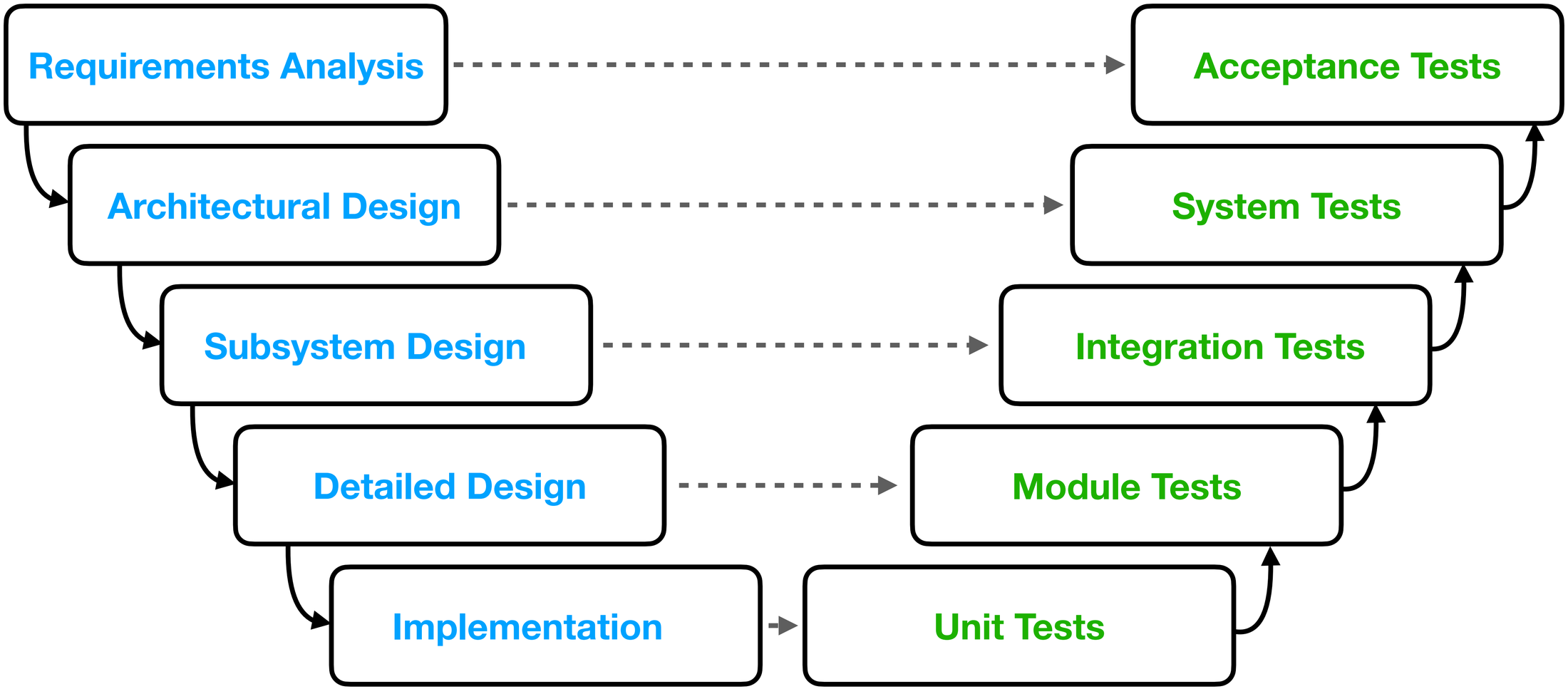
COM 1001

Introduction to Software Engineering

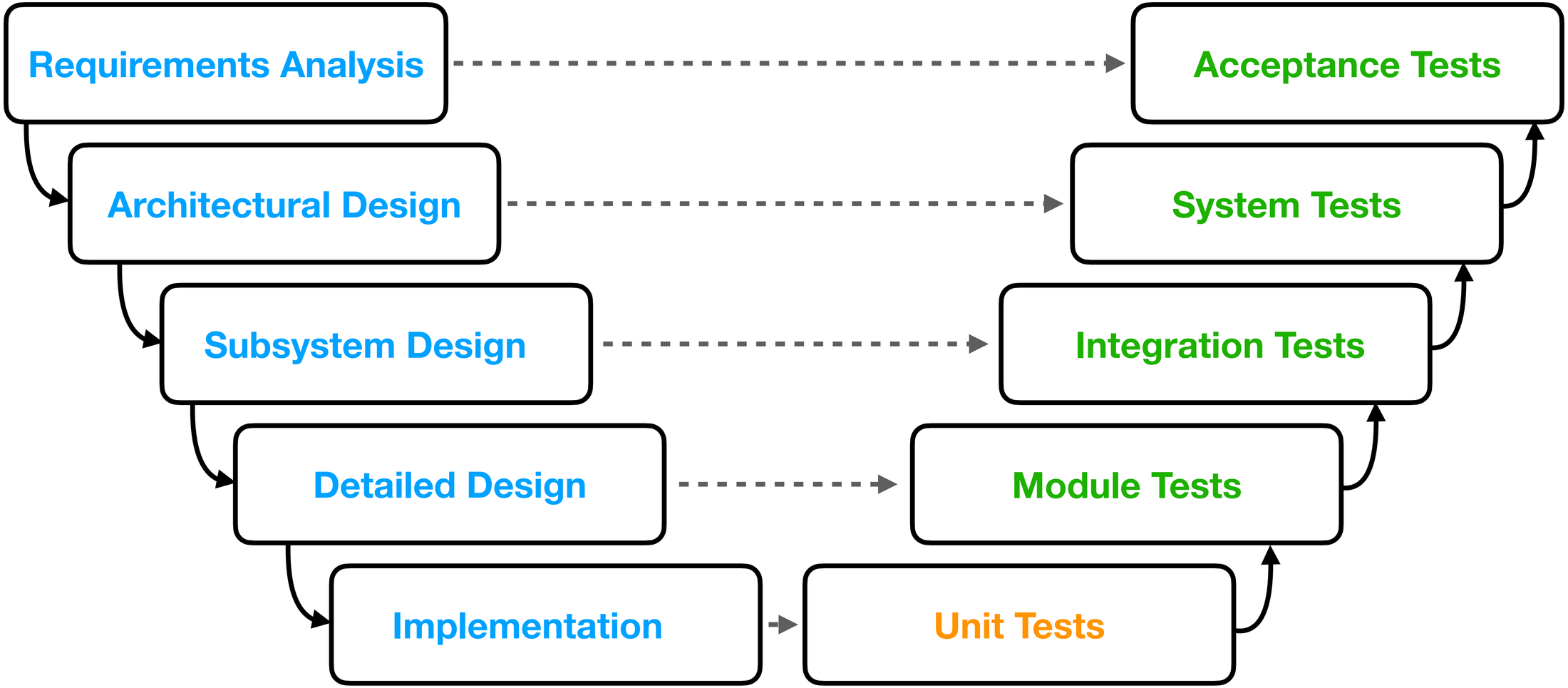
Professor Phil McMinn

End-to-End Testing

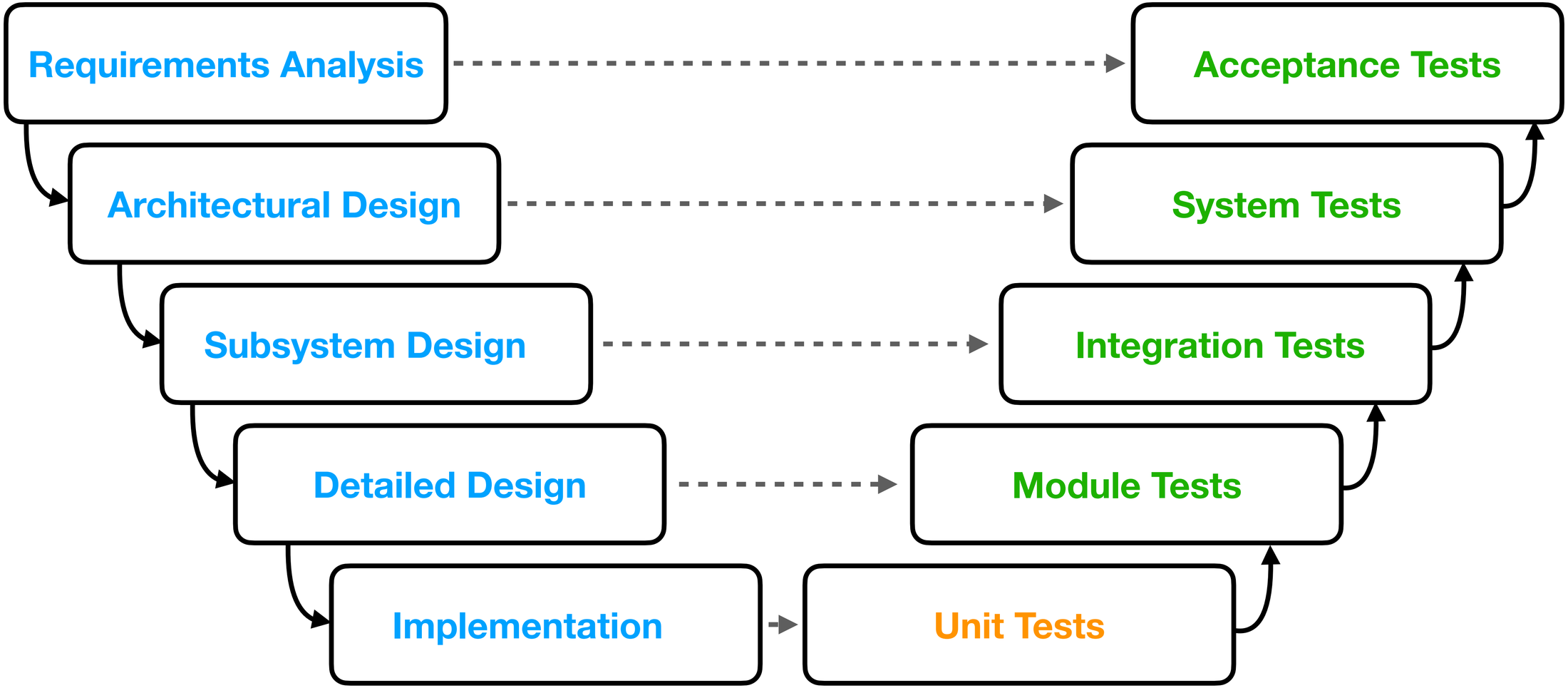
Types of Testing – The V Model



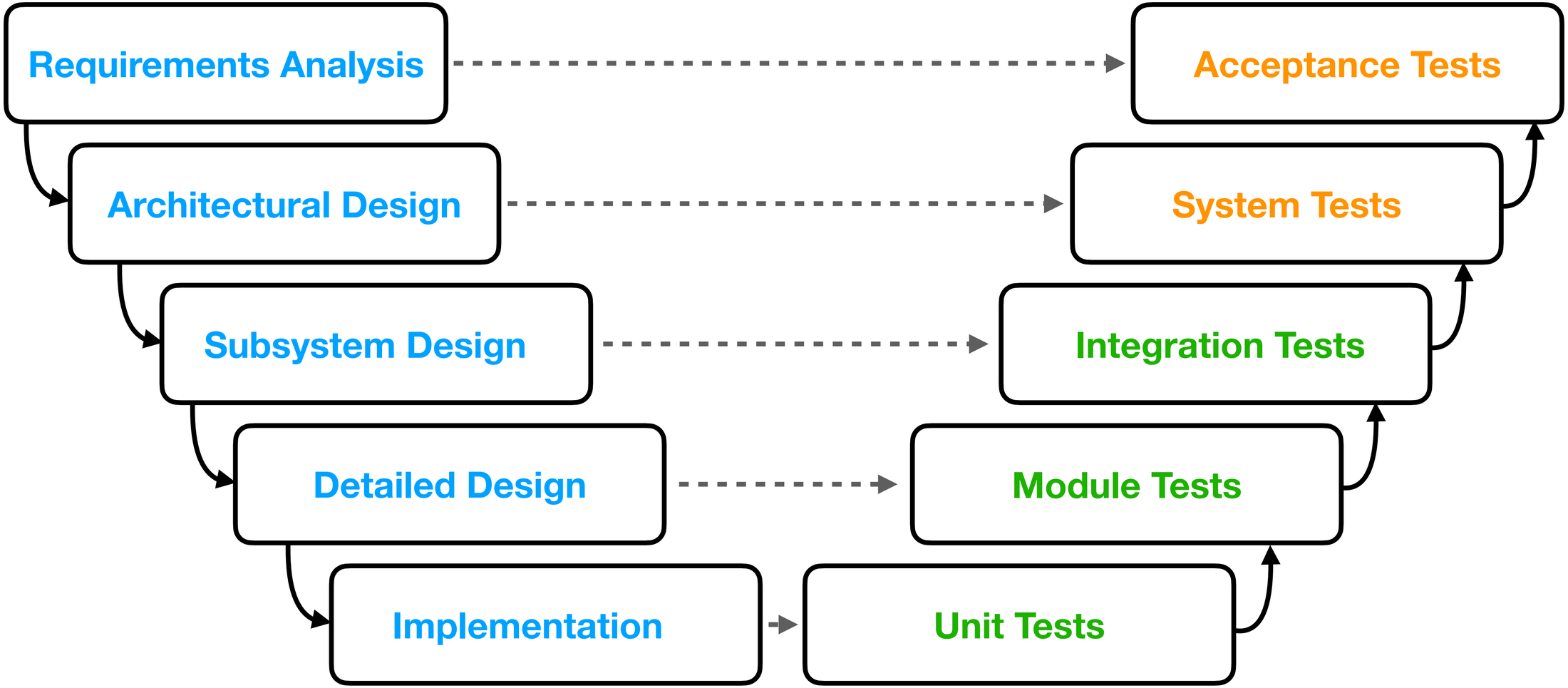
Types of Testing – The V Model



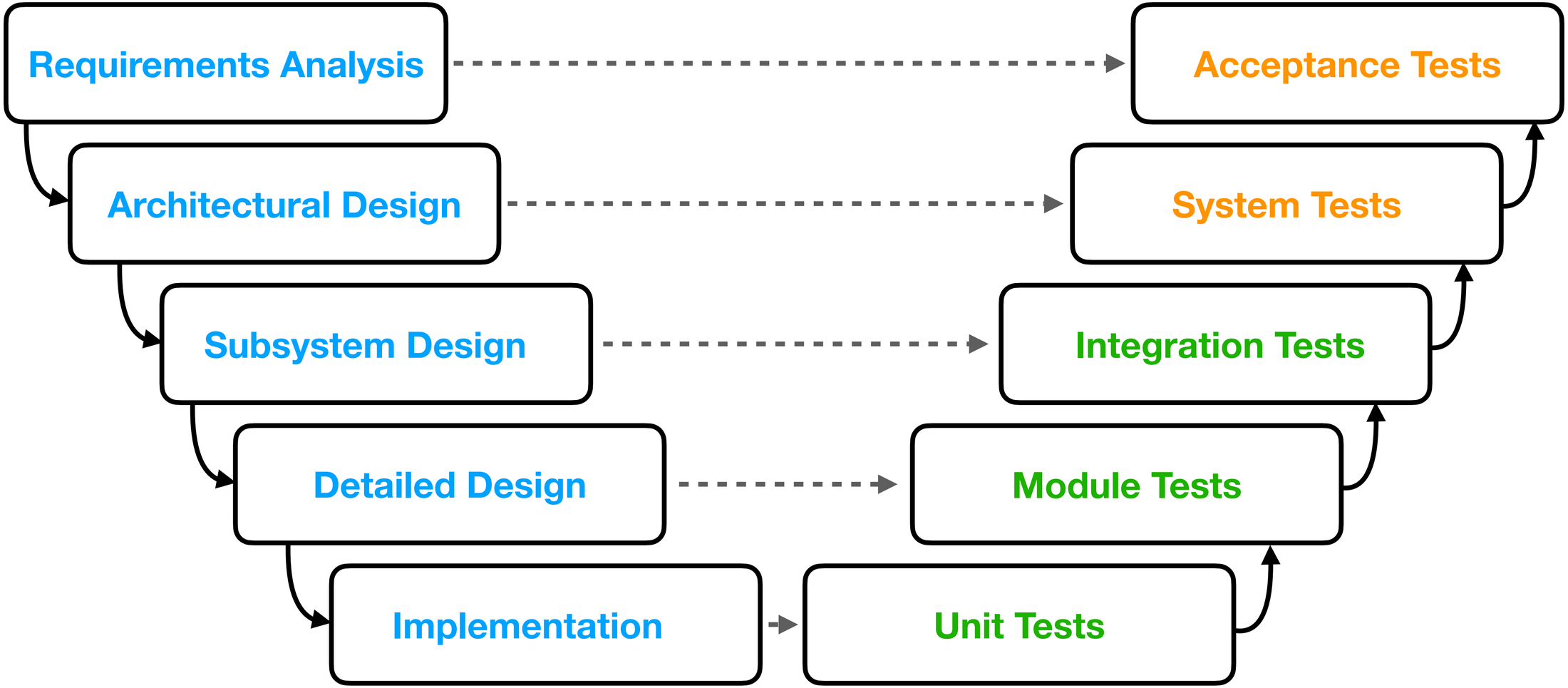
Types of Testing – The V Model



Types of Testing – The V Model



Types of Testing – The V Model



Acceptance Testing

Acceptance testing means testing that the requirements of the software have all been implemented correctly.

A tester derives the scenarios for testing from the story cards themselves.

They are sometimes also called **feature tests** because they describe and test individual software features that result from user stories.

System Tests

System tests on the other hand test the whole system, but test more generally that that system does and *doesn't do* things as it should, and that the components it is made up of all function together correctly.

End-to-End Tests

Both **acceptance tests** and **system tests** are sometimes both described as **“end-to-end” tests**.

This is because they are written from the **end-user’s point of view** by simulating real user scenarios – from one end of the system to another.

User Steps with Capybara

Capybara is a gem that lets us simulate a user interacting with a web page with Ruby code.

The following simulates interaction with the Add Player page of the `forms/football_players` app from the last lecture.

Simulates a user typing in a URL into their browser

This simulates the user filling in different form fields with data. The field name comes from the `name` attribute in the HTML

Finally, clicking the submit button. Again the name here comes from the `name` attribute in the HTML. We must write it exactly as it appears, including putting the characters in the right case.

```
visit "/add"
fill_in "first_name", with: "George"
fill_in "surname", with: "Test"
fill_in "gender", with: "M"
fill_in "club", with: "Mantester Utd"
fill_in "country", with: "Northern RSpec"
fill_in "position", with: "Midfield"
fill_in "date_of_birth", with: "1946-05-22"
click_button "Submit"
```

Because adding a player to the database is something we need to do frequently in our tests, we're going to make these steps into a method.

```
visit "/add"
fill_in "first_name", with: "George"
fill_in "surname", with: "Test"
fill_in "gender", with: "M"
fill_in "club", with: "Mantester Utd"
fill_in "country", with: "Northern RSpec"
fill_in "position", with: "Midfield"
fill_in "date_of_birth", with: "1946-05-22"
click_button "Submit"
```

Because adding a player to the database is something we need to do frequently in our tests, we're going to make these steps into a method.

```
def add_test_player
  visit "/add"
  fill_in "first_name", with: "George"
  fill_in "surname", with: "Test"
  fill_in "gender", with: "M"
  fill_in "club", with: "Mantester Utd"
  fill_in "country", with: "Northern RSpec"
  fill_in "position", with: "Midfield"
  fill_in "date_of_birth", with: "1946-05-22"
  click_button "Submit"
end
```

Capybara with RSpec

```ruby
require_relative "../spec_helper"

describe "the add page" do
  it "is accessible from the search page" do
    visit "/search"
    click_link "Add a new player to the database"
    expect(page).to have_content "Add Player"
  end

  it "will not add a player with no details" do
    visit "/add"
    click_button "Submit"
    expect(page).to have_content "Please correct the errors below"
  end

  it "adds a player when all details are entered" do
    add_test_player
    expect(page).to have_content "George Test"
    clear_database
  end
end
```

forms/football_players/spec/features/add_spec.rb

The test is of a familiar format, since we're embedding the Capybara steps within RSpec

Checking the page has content is written slightly differently with Capybara compared to how we saw it before.

Adds a player by filling out the form, as we just saw in the last slide

These types of test live in the `spec/features` directory

Contains the `add_test_player` method we just defined, plus code for setting up RSpec and Capybara, setting `ENV["APP_ENV"]` to `"test"` so that the test database is used, and so on.

Simulates the user clicking a link. The link text must be exactly as it appears on the page, including casing.

This is another method that lives in `spec_helper.rb` to reset the database after a test. We could have also simulated the user going to the delete page for the player.

Running RSpec End-to-End Tests

Running RSpec acceptance tests is no different to any other RSpec test, we invoke the `rspec` command at the command line.

Although Capybara can drive a real web browser to do the testing (e.g., through the use of Selenium), we've got it configured in **"headless"** mode. This is faster, and works on Codio.

However, because we cannot "see" the tests running, it's sometimes hard to understand why a test fails, because we cannot see what the test is seeing:

```
codio@north-mister:~/workspace/com1001-code/forms/football_players$ rspec spec/features/add_spec.rb
..F

Failures:

  1) the add page adds a player when all details are entered
     Failure/Error: expect(page).to have_content "George Best"
       expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
     # ./spec/features/add_spec.rb:18:in `block (2 levels) in <top (required)>'

Finished in 0.24422 seconds (files took 2.71 seconds to load)
3 examples, 1 failure

Failed examples:

rspec ./spec/features/add_spec.rb:16 # the add page adds a player when all details are entered
```

Debugging Tests with `save_page`

This will save the HTML page that Capybara sees to the filestore

Note that `save_page` must come before the failing `expect` statement.

```
it "adds a player when all details are entered" do
  add_test_player
  save_page # temporary line of code to work out what's going on
  expect(page).to have_content "George Best"
  clear_database
end
```

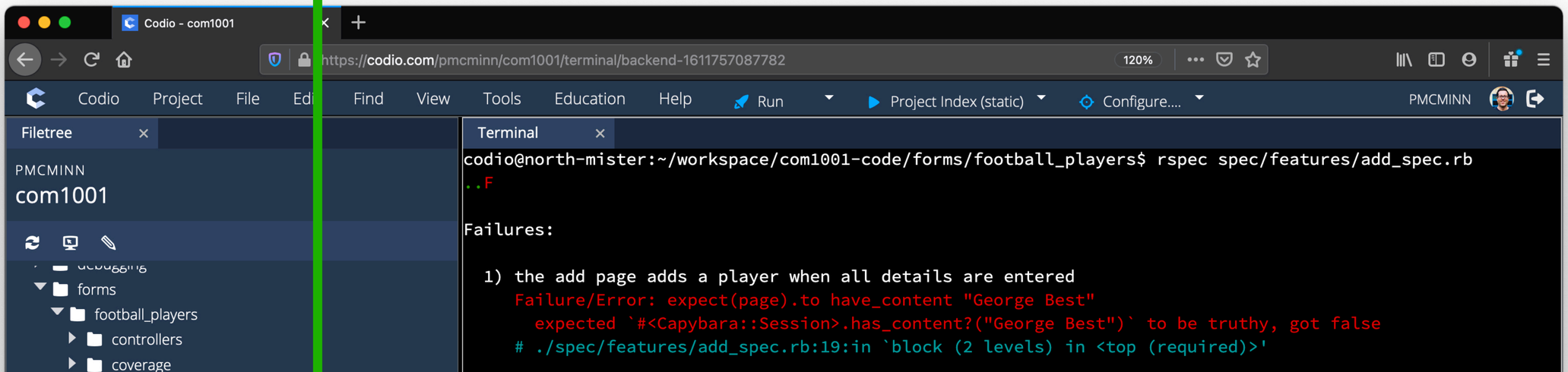
Deliberate mistake to demonstrate the point in this slide. The added test player is was a legend in his time, the one and only “`George Test`”.

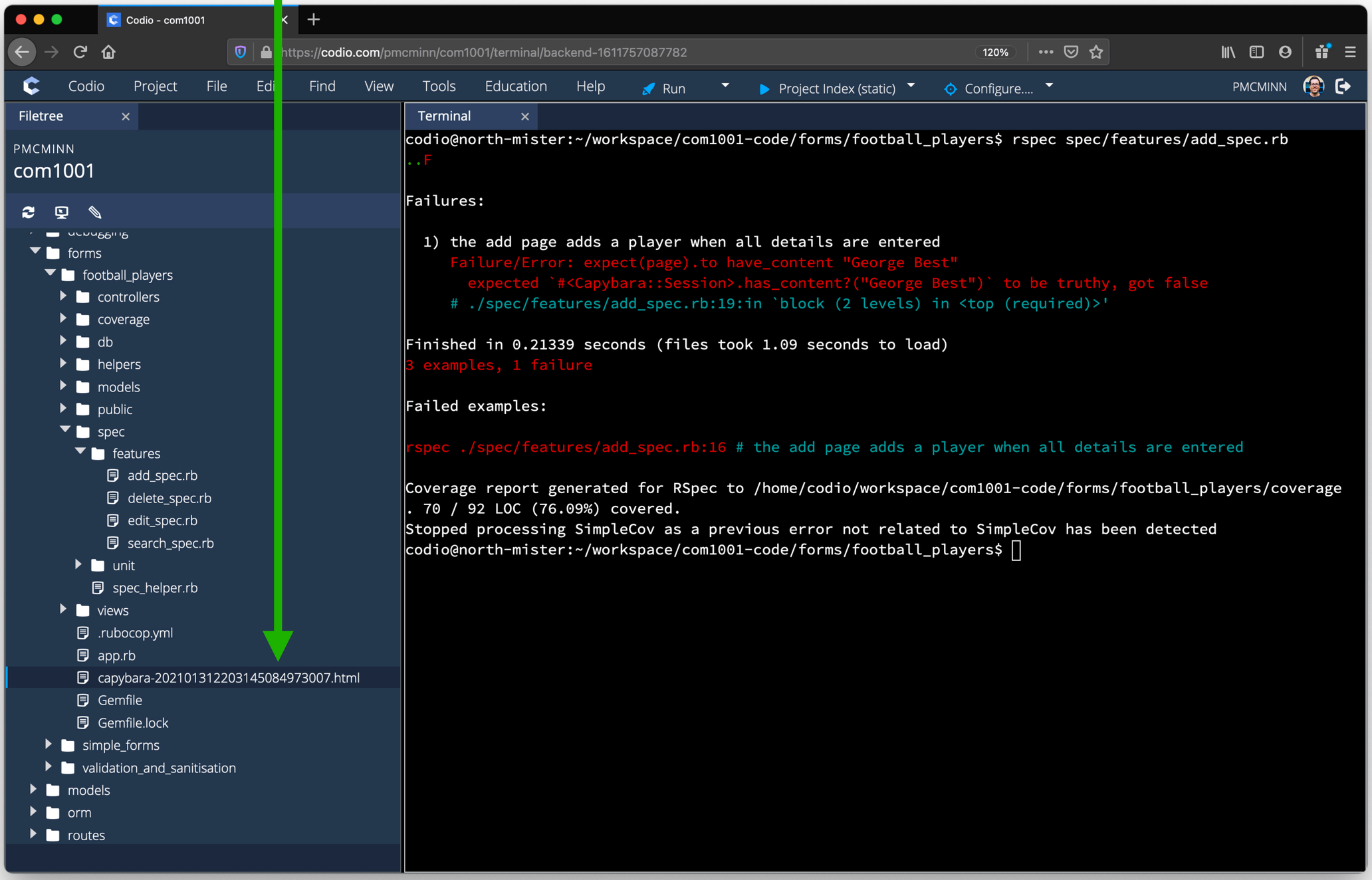
Debugging Tests with save_page

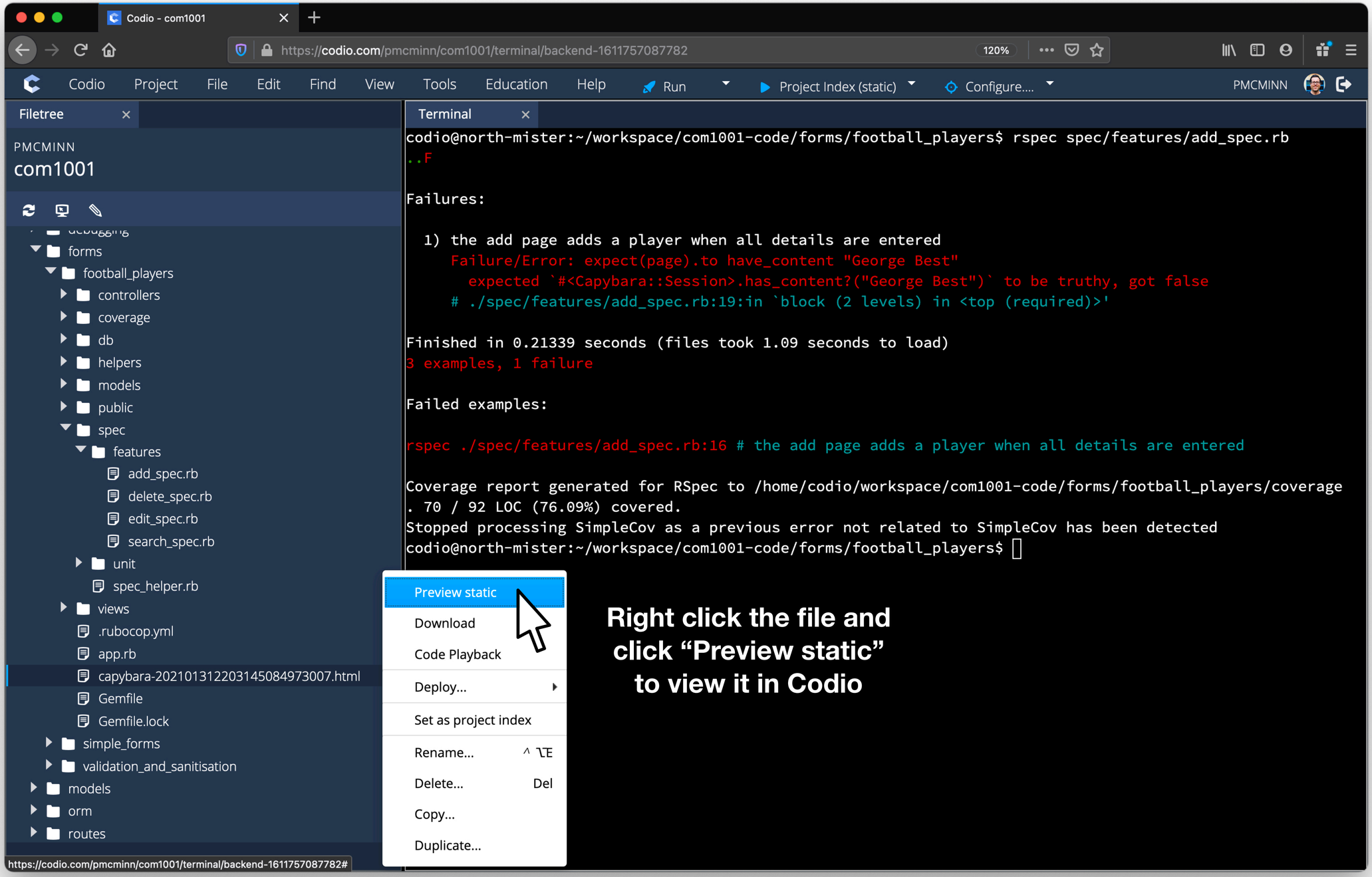
This will save the HTML page that Capybara sees to the filestore

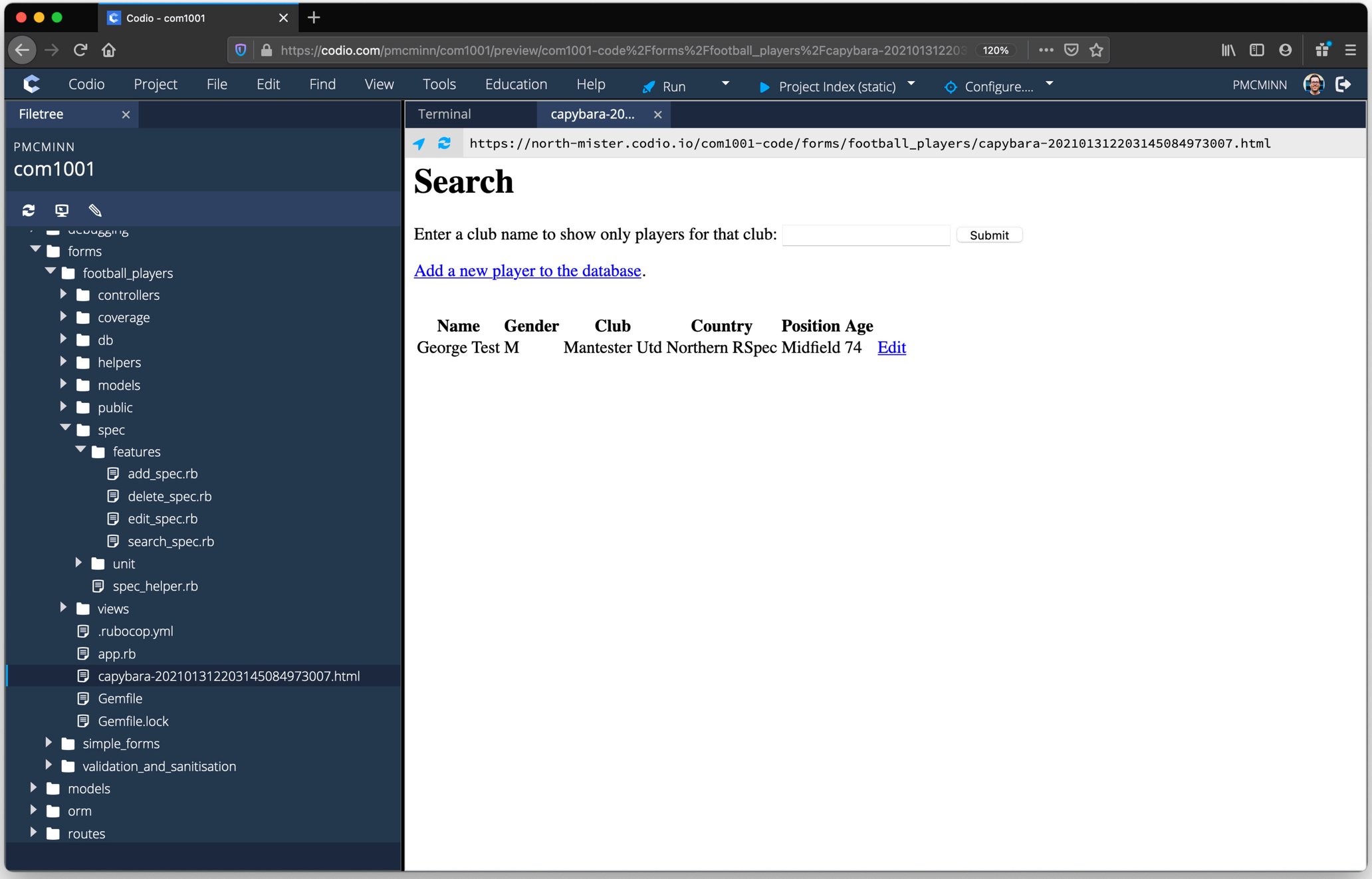
```
it "adds a player when all details are entered" do
  add_test_player
  save_page # temporary line of code to work out what's going on
  expect(page).to have_content "George Best"
  clear_database
end
```

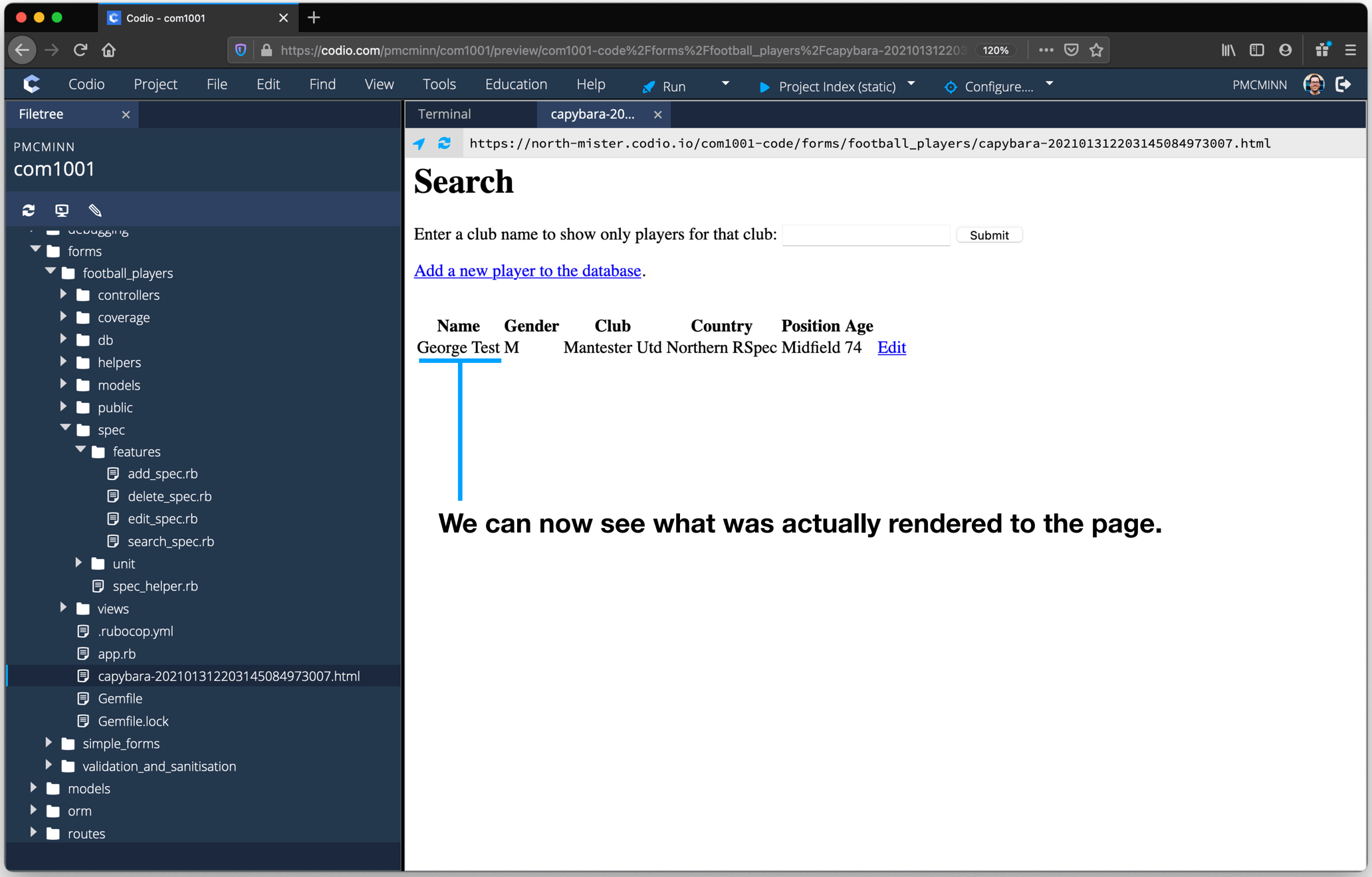
Deliberate mistake to demonstrate the point in this slide. The added test player is was a legend in his time, the one and only “George Test”.




Codio - com1001
https://codio.com/pmcminn/com1001/terminal/backend-1611757087782
120%
Codio
Project
File
Edi
Find
View
Tools
Education
Help
Run
Project Index (static)
Configure....
PMCMINN
Filetree
PMCMINN
com1001
forms
football_players
controllers
coverage
db
helpers
models
public
spec
features
add_spec.rb
delete_spec.rb
edit_spec.rb
search_spec.rb
unit
spec_helper.rb
views
.rubocop.yml
app.rb
capybara-20210131220314508497300 7.html
Gemfile
Gemfile.lock
simple_forms
validation_and_sanitisation
models
orm
routes
Terminal
codio@north-mister:~/workspace/com1001-code/forms/football_players$ rspec spec/features/add_spec.rb
..F
Failures:
1) the add page adds a player when all details are entered
Failure/Error: expect(page).to have_content "George Best"
expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
./spec/features/add_spec.rb:19:in `block (2 levels) in <top (required)>'
Finished in 0.21339 seconds (files took 1.09 seconds to load)
3 examples, 1 failure
Failed examples:
rspec ./spec/features/add_spec.rb:16 # the add page adds a player when all details are entered
Coverage report generated for RSpec to /home/codio/workspace/com1001-code/forms/football_players/coverage
. 70 / 92 LOC (76.09%) covered.
Stopped processing SimpleCov as a previous error not related to SimpleCov has been detected
codio@north-mister:~/workspace/com1001-code/forms/football_players$


Codio - com1001
https://codio.com/pmcminn/com1001/terminal/backend-1611757087782
120%
Codio
Project
File
Edit
Find
View
Tools
Education
Help
Run
Project Index (static)
Configure....
PMCMINN
Filetree
PMCMINN
com1001
forms
football_players
controllers
coverage
db
helpers
models
public
spec
features
add_spec.rb
delete_spec.rb
edit_spec.rb
search_spec.rb
unit
spec_helper.rb
views
.rubocop.yml
app.rb
capybara-20210131220314508497 3007.html
Gemfile
Gemfile.lock
simple_forms
validation_and_sanitisation
models
orm
routes
Terminal
codio@north-mister:~/workspace/com1001-code/forms/football_players$ rspec spec/features/add_spec.rb
..F
Failures:
1) the add page adds a player when all details are entered
Failure/Error: expect(page).to have_content "George Best"
expected `#<Capybara::Session>.has_content?("George Best")` to be truthy, got false
./spec/features/add_spec.rb:19:in `block (2 levels) in <top (required)>'
Finished in 0.21339 seconds (files took 1.09 seconds to load)
3 examples, 1 failure
Failed examples:
rspec ./spec/features/add_spec.rb:16 # the add page adds a player when all details are entered
Coverage report generated for RSpec to /home/codio/workspace/com1001-code/forms/football_players/coverage
. 70 / 92 LOC (76.09%) covered.
Stopped processing SimpleCov as a previous error not related to SimpleCov has been detected
codio@north-mister:~/workspace/com1001-code/forms/football_players$
Preview static
Download
Code Playback
Deploy...
Set as project index
Rename... ^ ⌥E
Delete... Del
Copy...
Duplicate...
https://codio.com/pmcminn/com1001/terminal/backend-1611757087782#
Right click the file and
click “Preview static”
to view it in Codio


Codio - com1001
https://codio.com/pmcminn/com1001/preview/com1001-code%2Fforms%2Ffootball_players%2Fcapybara-20210131220
120%
Codio
Project
File
Edit
Find
View
Tools
Education
Help
Run
Project Index (static)
Configure....
PMCMINN
Filetree
PMCMINN
com1001
forms
football_players
controllers
coverage
db
helpers
models
public
spec
features
add_spec.rb
delete_spec.rb
edit_spec.rb
search_spec.rb
unit
spec_helper.rb
views
.rubocop.yml
app.rb
capybara-20210131220314508497 3007.html
Gemfile
Gemfile.lock
simple_forms
validation_and_sanitisation
models
orm
routes
Terminal
capybara-20...
https://north-mister.codio.io/com1001-code/forms/football_players/capybara-20210131220314508497 3007.html
Search
Enter a club name to show only players for that club:
Submit
Add a new player to the database.
Name Gender Club Country Position Age
George Test M Mantester Utd Northern RSpec Midfield 74 Edit


Codio - com1001
https://codio.com/pmcminn/com1001/preview/com1001-code%2Fforms%2Ffootball_players%2Fcapybara-20210131220
120%
Codio
Project
File
Edit
Find
View
Tools
Education
Help
Run
Project Index (static)
Configure....
PMCMINN
Filetree
PMCMINN
com1001
forms
football_players
controllers
coverage
db
helpers
models
public
spec
features
add_spec.rb
delete_spec.rb
edit_spec.rb
search_spec.rb
unit
spec_helper.rb
views
.rubocop.yml
app.rb
capybara-20210131220314508497 3007.html
Gemfile
Gemfile.lock
simple_forms
validation_and_sanitisation
models
orm
routes
Terminal
capybara-20...
https://north-mister.codio.io/com1001-code/forms/football_players/capybara-20210131220314508497 3007.html
Search
Enter a club name to show only players for that club:
Submit
Add a new player to the database.
Name Gender Club Country Position Age
George Test M Mantester Utd Northern RSpec Midfield 74 Edit
We can now see what was actually rendered to the page.

More Examples

Check out the `forms/football_players/spec/features` directory in the code examples repository for more examples of using Capybara with RSpec to test other parts of the app, including the edit, search and delete pages.

For more on Capybara, see

- Its web page: https://teamcapybara.github.io/capybara/

- A cheat sheet for using Capybara: https://devhints.io/capybara