University of Glasgow | School of Computing Science

Assessed Coursework

Course Name	**Web Application Development 2**			
Coursework Number	**1 (of 5) – Rango application**			
Deadline	**Time:**	**6.30pm**	**Date:**	**9 February 2018**
% Contribution to final course mark	**10**		**This should take at most this many hours:**	**15**
Solo or Group ✓	**Solo**	✓	**Group**	
Submission Instructions	**See under "How to submit" on Page 4**			
Who Will Mark This? ✓	**Lecturer** ✓	**Tutor**	**Other**	
Feedback Type? ✓	**Written** ✓	**Oral**	**Both**	
Individual or Generic? ✓	**Generic**	**Individual** ✓	**Both**	
Other Feedback Notes				
Please Note: This Coursework cannot be Re-Done				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below. The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

Marking Criteria
See Pages 3-4

Web Application Development 2

Rango Application (10%)

Introduction

The course is based on the e-book "Tango with Django: A beginners guide to web development", by Leif Azzopardi and David Maxwell, published by Lean Publishing (henceforth "TWD"). The latest version has been updated for Django 1.11. You will be using this book primarily during the lab sessions, especially in the first half of the course.

TWD contains a step-by-step tutorial that guides the reader through the development of a web application called Rango, built using Python and Django. Rango lets users browse through user-defined categories to access various web pages. During the first five lab sessions in particular you will be working through the chapters of TWD to develop your own version of Rango.

We recommend that you work through all 20 chapters in TWD, however only your work on the first 10 chapters will be assessed. The development of your Rango application will account for 10% of your overall mark for the course. Moreover, successful completion of Rango will give you the skills that you need in order to work on the WAD2 team project later, which accounts for 40% of your overall mark.

The IDLE and PyCharm IDEs are available for Python development; you may however prefer to develop Rango within PythonAnywhere. A recommended development schedule is given below, together with some important guidance in relation to working on Rango.

Working on Rango – Important Guidance

The following guidelines are very important: be sure to read them carefully as they relate to the assessment of your Rango application.

1. You MUST make regular commits to your Git repository: AT LEAST once per chapter. We will be looking for evidence of regular commits when it comes to marking your Rango application.

2. You should ensure that any messages are conveyed exactly as described in TWD. For example, if your web page is supposed to include the message "Rango says hello world" then you must include exactly that message, and not "Rango says hey there" or "Rango says Hello World". Similarly, files must be named exactly as stated – if you are required to include an image named rango.jpg, this MUST be the filename and not Rango.JPG, for example. Automated testing will be carried out when it comes to marking your Rango application (see Chapter 18 of TWD) and you may lose marks if tests fail due to inconsistencies of the types described here.

3. We expect you to do the exercises at the ends of the chapters in TWD (up to Chapter 10) – some of the automated tests are based on successful completion of these exercises.

4. You may find completed versions of Rango in various places on the web. If you download such a version and submit it as your own developed application, you will find that you will not score highly for this exercise. This is because the automated tests are specifically designed so that they check out the step-by-step development of your application. As the app changes over time, some tests from earlier chapters will fail. This is why it is so important to commit regularly (and at least once per chapter) because we will run our tests against all versions of your code in your commit history.

To explain this latter point further, suppose you have two commits of your code: one at time point A, and one at the later time point B, and suppose that automated test T is to be run against your code. As long as test T passes on at least one of versions A and B, you will be given the mark for this test.

Automated testing

Automated testing of your Rango application (taking code from your Github repository) will be carried out shortly after two checkpoints, as follows:

A. **Rango checkpoint 1** (worth 10% of the overall mark): end of week 3 (26 January 2018). The automated tests for Chapters 3-10 will be run on your Github repository. You will receive feedback (via an automated email) on which tests you passed and which tests you failed.

   This part of the exercise will be summatively assessed, though the main aim is simply to ensure participation in checkpoint 1. You will receive full marks if you pass at least 1 test, and no credit otherwise (with no other marking outcomes available).

   You can then use the feedback from the automated tests carried out at checkpoint 1 to improve your Rango application before checkpoint 2. Automated tests for all chapters will be run at checkpoint 1, though we recommend that by the end of week 3 you have completed up to the end of Chapter 6. The test script will look at all your commits up to 6.30pm on 26 January 2018.

B. **Rango checkpoint 2** (worth 90% of the overall mark): end of week 5 (9 Feburary 2018). The automated tests for Chapters 3-10 will be run again on your Github repository and this time the results of running these tests will form the basis of 90% of your mark for the Rango application. You will again receive a summary (via an automated email) of the tests that you passed and those that you failed. The test script will look at all your commits up to 6.30pm on 9 February 2018.

Rango development schedule

By the end of week 5 you should have completed the development of your Rango application up to the end of Chapter 10, as Rango checkpoint 2 will take place at this time. As a guideline to help you plan your work, our expectation is that you will have completed the chapters shown in each row of the following table by the end of the week in question:

Week no.	Week ending	TWD chapters	Remarks
1	12/01/2018	1, appendix on Git	
2	19/01/2018	3, 4	
3	26/01/2018	5, 6	Rango checkpoint 1
4	02/02/2018	7, 8	
5	09/02/2018	9, 10	Rango checkpoint 2

You are of course free to develop Rango ahead of this recommended schedule, but you should make sure that you do not fall behind due to the fixed checkpoints in weeks 3 and 5.

Carrying out your own testing

In addition to the automated tests that we will carry out on your code, you should carry out your own testing. This will allow you to gain feedback on how many tests you are passing and failing in advance of the two checkpoints.

Some tests are publicly available for you to use and can be obtained from https://github.com/gerac83/rango_tests (courtesy of Dr Gerardo Aragón-Camarasa). You should clone this repository, work on your Django virtual environment and then issue the following command to run the tests:

```
python run_tests.py -u "Your GitHub repository" -s "student number"
-d "YYYY-MM-DD"
```

without the quotes. This will run the tests against all of your commits up to and including the date given by YYYY-MM-DD, putting the results in a subfolder whose name is your student number, contained in the output folder. Here is an example usage of this command:

```
python run_tests.py -u https://github.com/wad2/tango_with_django.git
-s 2099999z -d 2018-01-20
```

The automated tests that will be executed after the two Rango checkpoints will include those in the Github repository https://github.com/gerac83/rango_tests (the "public" tests). There will additionally be automated live server tests that will be carried out using Selenium (the "private" tests). You may wish to replicate these tests yourself, and you can find out more about them in the Appendix (the code for these tests will not, however, be made available to students).

It is highly recommended to run the public tests yourself in order to gain frequent and prompt feedback on the development of your Rango application.

Note: do not use the tests associated with Chapter 18 of TWD! These are out of date and do not correspond to the automated tests that will be run against your submission.

How to submit

You should provide details of the location of your Github repository by the two Rango checkpoint deadlines. The precise timings are:

Rango checkpoint 1: Friday 26 January 2018 at 6.30pm
Rango checkpoint 2: Friday 9 February 2018 at 6.30pm

This information should be supplied via a custom-built web app that may be accessed via https://wad2app.pythonanywhere.com/. You will need to login using with the same username as your GUID (i.e., your student number plus the first initial of your surname) and using the password that will have been sent to you by email. Then supply your Github repository URL, which should be along the following lines:
https://github.com/<username>/tango_with_django_project.git.

Marking scheme

When the automated tests are run after Rango checkpoint 2, you will gain 1 mark for each test passed and 0 marks for each test failed (or terminated early due to an error). The total marks you gained will then be expressed as a percentage of the total number of tests N (currently N=68 though the final number of tests may vary slightly from this).

Your percentage P will then be adjusted to take account of the number of commits C that you made. Since you are supposed to commit at least once per chapter, you should have at least 8 commits from Chapters 3-10. Your overall percentage Q for checkpoint 2 will then be P multiplied by min{C,8}/8. So, for example, if you made only 4 commits, Q will be P divided by 2.

Your final percentage R will be made up of your mark from checkpoint 1 weighted at 10% and your mark (i.e., Q) from checkpoint 2 weighted at 90%. R will then be converted to a band which will be your mark for this component of the assessment. You will receive an automated email with details of how many (and which) tests you passed and failed, together with more detailed feedback on any test errors or failures.

Common pitfalls

Based on prior experience, there are some scenarios that cause the tests not to run properly. We want to avoid a situation where you have done a lot of work developing Rango, but then find that you are passing none of the tests! You will be able to determine yourself if this is likely to happen by running the tests for yourself (see under "Carrying out your own testing", above). There are some common reasons as to why the tests might not run properly, so if you are in this position it would be worth checking the following list to see if any of these reasons apply to you:

- You are running the tests from within a Dropbox folder (or similar cloud service provider) – it is more reliable to run the tests from a local drive or networked drive.
- You receive the following error: "fatal: destination path 'temporal' already exists and is not an empty directory" – delete the folder "temporal".
- You added app_name='rango' and you were namespacing your URLs. Although this is good practice in general, please avoid this for the purposes of automated testing!
- Your code is not in the master branch of your Github repository.
- You committed your virtual environment – you must not do this.
- You gave an incorrect Github URL.
- Your Github repository is private – it must be public!
- You developed Rango on PythonAnywhere – it is fine to use PythonAnywhere for the development but you should ensure that Rango can be deployed successfully on localhost before you make your Git commits.
- You have omitted to include __init.py__ in the relevant project folders (thus git add has not picked up all the files that should be committed).
- The module manage.py is missing from one of the commits – this could cause all of the tests to fail.
- The module urls.py is missing from one of the commits – this could cause all of the tests to fail.
- Your repository setup is unexpected. Refer back to the first lab sheet for an explanation as to how your repository should be structured on Github.
- The process timed out – this can happen if you made far too many commits (typically over 40). It is recommended to commit early and commit often (at least once per chapter), but you must not take this to excess! Try to aim for between 8-20 commits.

Appendix 1: Live Server Tests

The objective of these tests is to simulate the interaction of a potential visitor in your Rango app. We will run automated tests to verify the functionality of your web application using Selenium for Python (http://selenium-python.readthedocs.io/). You can either attempt to create your own tests that automate the functionality described here, or act as a user and verify that the interaction is successful.

Chapter 3

3.1 Navigate from index to about pages and from about to index pages

This test consists of navigating from the Index page to the About page. Make sure you can navigate between these pages using the “Index” and “About” links provided.

Chapter 4

There are no live server tests for this chapter.

Chapter 5

5.1 Population Script

The aim of this test is to verify that you have created your population script correctly. The test comprises:

1. Run the population script
2. Navigate to the admin page
3. Login as admin (the tests generate a local admin login, so no need to worry about this)
4. Click on “Categories” (Make sure that the text displayed is “Categories” – the test is case sensitive)
5. Check for the categories saved by the population script – The test will look for “Other Frameworks”, “Django” and “Python”
6. Check the pages saved by the population script

5.2 Admin page contains the page’s title, url and category

The aim of this test is to verify that you have completed the exercise at the end of Chapter 5. The test comprises:

1. Run the population script
2. Navigate to the admin page
3. Login as admin (the tests generate a local admin login, so no need to worry about this)
4. Click on “Pages” (Make sure that the text displayed is “Pages” – the tests is case sensitive)
5. The test will verify that each pag’s title and url are displayed
    a. Make sure that the following urls are defined in your population script:
        i. http://www.djangorocks.com/

ii. http://flask.pocoo.org

5.3 Create new category via the admin site

The objective is to create a new category using Django administration pages. This test will create the “Test Driven Development” category and verify that the new category displays the category’s name in the browser.

Chapter 6

6.1 Category on the admin page contains the slug field

This test verifies that a slug field exists for a category and checks if the slug is defined correctly. The tests consists of:

1. Run the population script
2. Navigate to the admin page
3. Login as admin (the tests generate a local admin login, so no need to worry about this)
4. Click on “Categories” (Make sure that the text displayed is “Categories” – the tests is case sensitive)
5. Click on “Other Frameworks” (Make sure the text displayed is “Other Frameworks”)
6. Check that the slug field contains: “other-frameworks”

6.2 Category navigates to the desired page

This test is about navigating and accessing categories defined in the population script. The test will check that the url of each category is displayed correctly (i.e. using slugs). The categories the test will check are:

1. Python – slug: python
2. Django – slug: django
3. Other Frameworks – slug: other-frameworks

Chapter 7

7.1. Add new category form saves a category

The test will attempt to create a new category by interacting with the add new category form developed in this chapter. The workflow is as follows:

1. Access the Index page
2. Click on “Add a New Category” (the test will only work if the html link text is “Add a New Category” or “Add New Category” – case sensitive)
3. Create a category named “New Category” and click submit.
4. Checks that the category name appears in the index page.

7.2. Send error when category field is empty

This test checks if an error is displayed if the category name field in the form is empty after clicking “submit”. The test comprises:

1. Access the Index page
2. Click on “Add a New Category” (the test will only work if the html link text is “Add a New Category” or “Add New Category” – case sensitive)

3. Click submit without entering category name
4. Checks that the error message appears on screen (i.e. check for the word “required” appears in the rendered HTML output).

7.3. Send error when category exists

This test checks if an error is displayed if the category already exists in the database. The test comprises:

1. Access the Index page
2. Click on “Add a New Category” (the test will only work if the html link text is “Add a New Category” or “Add New Category” – case sensitive)
3. Create a category named “New Category” and click submit.
4. Checks that the error message is “Category with this Name already exists.” – this test is not case sensitive but be aware of your spelling!

7.4. Add new page form saves a page (Django url mappings)

The test will attempt to create a new page by interacting with the add new page form developed in this chapter. The workflow is as follows:

1. The test will create 11 different categories.
2. For each category
    a. access the “add_page” page form of that category using Django’s url mappings
    b. Inputs the name of the page (random) and url (random)
    c. Click submit.
    d. Verify that the page appears in the category page

7.5 Add new page form saves a page (Direct url)

This test is essentially the same as the test above. The difference is that step 2.a is accessed as:

- …/rango/category/<category.slug>/add_page/

Chapter 8

There are no live server tests for this chapter.

Chapter 9

9.1 Register a new user

This test will register a new user using the user registration form developed in this chapter. Make sure that the HTML template contains the following text (not case sensitive)

- “Sign Up”
- “Rango says: thank you for registering!”
- “Return to the homepage.”

Remember to check your spelling, otherwise the test will fail!

9.2 Admin page contains user profile

With the registered user above, this test will verify that it has been recorded correctly in the database. The workflow is:

1. Navigate to the admin page
2. Login as admin (the tests generate a local admin login, so no need to worry about this)
3. Click on User profiles (Make sure that the text displayed is “User profiles” – the tests is case sensitive)
4. Check that the user has been registered correctly

9.3 Restricted links in the Index page appear when logged in

This test aims to verify that restricted links (e.g. Restricted Page”) only appear when a user is logged in. This test will check in the Index page that the HTML link texts contain (case sensitive):

- “Add a New Category”
- “Restricted Page”
- “Logout”
- “About”

Likewise, the test will check that the following HTML link texts do not appear in the Index page (case sensitive):

- “Sign In”
- “Sign Up”

9.4 Not-Restricted links in the Index page appear when not logged in

Similar as above but when the user has not logged in. Not-restricted HTML link texts should be (case sensitive):

- “Sign In”
- “Sign Up”
- “About”

Hence, the following should not appear in the Index page (case sensitive):

- “Add a New Category”
- “Restricted Page”
- “Logout”

9.5 Logout link

This test will check that after a user logins, the user can logout. The test will successfully finish if the HTML link test “Sign In” appears in the displayed page (case sensitive)

9.6 Add category when logged

This test is similar to test 7.1. The difference is that adding a new category is now possible only when a user logs in.

9.7 Add page when logged

This test is similar to test 7.5. The difference is that adding a new page is now possible only when a user logs in.

9.8 Add page when not logged

This test checks that it is not possible to add a page when the user had not logged in.

9.9 Access restricted page when logged

This test verifies that the user can access the restricted page developed in this chapter only when logins. The test expects the following message (no case sensitive):

- “Since you're logged in, you can see this text!”

9.10 Access restricted page when not logged

This test checks that the user cannot access the restricted page. If the user attempts to access it, the browser should display the login form.

9.11 Logged user name is displayed in Index

This tests simply checks that after a user logs in, the user’s name is displayed in the Index page.

Chapter 10

There are no live server tests for this chapter.