Curtin University — Department of Computing

Introduction to Software Engineering (ISAD1000)

2015, semester 1

Assignment

Due: Week 10 – Monday 4 May, 2:00 pm

Weight: 20% of the unit mark.

Introduction

Kickstarter is a company that helps put *creators* (of almost anything) together with *backers*, who may be able to contribute money. The idea is that creators propose *projects*, each of which has a funding target and a funding deadline. Backers pledge to donate money to specific projects, and if/when the funding target is reached before the deadline, that money is transferred to the creator. (If the funding target is not reached before the deadline expires, no money is transferred.)

You want to set up a rival to Kickstarter, to perform essentially the same task. Your company will rely on a single software system to coordinate all its activities. For this assignment, you'll be specifying the functional requirements and some of the test cases for this software system.

Consult the following pages (and any others you're able to find) for more details on how Kickstarter works:

- Creating projects: www.kickstarter.com/help/faq/creator+questions;
- Backing projects: www.kickstarter.com/help/faq/backer+questions.

Note: these pages contain crucial information. You will not be able to complete the assignment without reading them. They are also available on Blackboard.

Your Tasks

As you are aware, your assignment mark is determined in part by your practical work (as detailed in the unit outline). Therefore, the practical worksheet signoffs are considered part of the assignment. You do not need to submit your practical work – simply continue to have it signed off.

The other part of the assignment is described here. Start as soon as possible, and keep at it. If you need clarification, the ask the lecturer good, specific questions, and ask them *early*.

Task I: Requirements

Your first task is as follows:

(a) Determine all the actors involved in the system.

(b) Determine as many significant functional requirements as you can, and express them as user stories.

What does "significant" mean? Your set of functional requirements should cover everything needed for the system to work in a useful and reasonable manner. Anything omitted should be relatively inconsequential. For our purposes, you may ignore the slight differences for users in different countries.

(c) Pick four user stories, substantively different from each other, and develop each one into a complete use case.

(d) For each of your four use cases, come up with rough plan for testing that functionality. In a paragraph of plain English, explain how you would verify whether the final system correctly implements the use case.

We're not after formal test cases at this point.

Note: *please* do not ask "what does this mean?" unless you've attempted to work it out for yourself first!

Task II: Testing

Development of the system has moved on, as planned, and your team is now busy designing submodules and test cases.

Your team has identified the need for a submodule for each of the following tasks:

- Checking whether a backer's pledge will push the total amount pledged "over the line" (equal to or more than the funding target).
- Searching for a project whose description contains a particular keyword.
- Determining the amount of funding per day required (purely for informational purposes), based on the total amount pledged so far, the funding target, and the funding deadline.

  This is the minimum amount that backers would need to pledge per day, on average, from now on, to meet the target within the deadline. If no more pledges are required, this is simply zero. A negative number has no meaning (except perhaps to indicate special situations).

These submodules will take some of their information from a database. To facilitate testing, your team will create extra submodules to support the test code. For example, `createFakeDB` will create a fake, empty database, and `addFakeProject` will take a set of project details and add them to the database. Other such submodules will be required as well: some to help set up test cases, and others to help verify test results.

Thus, your second task is as follows:

(a) Provide a partial design of each of the three production code submodules. You *do not* need to provide algorithms. For each submodule, state its name, its imports (names and datatypes) and exports (where applicable), and its database inputs and outputs. Also describe briefly what the submodule does with its imports and inputs, and what it gives back in its export and outputs.

(b) Design test cases for each production code submodule, using equivalence partitioning.

(c) Determine what set of test-support submodules (e.g. `createFakeDB`) you will need. Provide their names, imports, exports, inputs and outputs (whichever are applicable), and a short description.

(d) Implement your test cases using JUnit (using test-support submodules as needed).

This goes a little above and beyond what you've already done. You have some flexibility in how you design the various submodules – use it wisely!

Submission

Prepare a report containing your complete solutions to the above tasks, including all answers, discussion and source code. Your report must be neatly presented, easy to read and contain appropriate references (where you make use of external sources). Include your name and student ID.

Submit your report to the Assignment area on Blackboard in PDF format.

After submitting, please verify that your submission worked by downloading and opening your own report. It is *your responsibility* to ensure that you submit correctly.

You must also submit a completed, signed "Declaration of Originality" form to the lecturer in person (*not* electronically). Blank forms are available outside the lecturer's office (314.343).

Mark Allocation

Here is a rough breakdown of how marks will be awarded. The percentages are of the total possible assignment mark:

- 30% – Use cases.
- 20% – Other parts of Task I.
- 30% – Test design (Task II).
- 20% – Test code (Task II).
- -10% – for each practical signoff mark of zero (pro-rata for fractional marks; e.g. -7.5% for each signoff mark of $\frac{1}{4}$).

Academic Misconduct – Plagiarism and Collusion

Copying material (from other students, websites or other sources) and presenting it as your own work is **plagiarism**. Even with your own modifications, it is still plagiarism.

Exchanging assignment solutions, or parts thereof, with other students is **collusion**.

Plagiarism and collusion are both forms of academic misconduct and may lead to a grade of ANN (Result Annulled Due to Academic Misconduct) being awarded for the unit. Serious or repeated offences may result in termination or expulsion.

End of Assignment