Project Contract

This is your project contract form. The deadline is **6/11/2020**. Remember to tick "Send me an email receipt of my responses" at the end of this page to receive a confirmation email. Please note that you need to forward the "confirmation email" to your supervisor in order to validate this submission.

1.Student Name: Lampros Karseras

2.P-number: P2424629

3.Programme: Computer Science

4.Email address: [p2424629@my365.dmu.ac.uk](mailto:p2424629@my365.dmu.ac.uk)

5.Project Title: Automatic Testing Generator

6.Project Proposer: Supervisor

7.Supervisor: David Smallwood

8.Introduction (max. 100 words):

The Automatic Testing Generator (ATG) will be an application that will generate tests against a given codebase. The ATG will be simple to use, without substantial configuration, where Developers will be able to point the ATG to their code files with a set of restricted inputs and expected outputs. If the inputs are within the predefine constrains, it will generate unit tests using BlackBox techniques. ATG will also be able to test regression of the system, for example after a code change, using saved inputs and outputs.

9.Project Background (max. 300 words): *A brief description providing the project background/context. e.g. is it based on a business need? a technical need? does it arise from the interests of a particular person/company?*

Since increased companies are adapting the Agile methodologies, software testing is becoming a paramount component of the Software Development Life Cycle, adding another task for developers to complete. But, many developer teams, including the one that I worked as an Intern, do not follow the Test-Driven Development style. Instead the tests are written afterwards against the code and many times are not extensive enough, leaving bugs that propagate down the SDLC.

By making an Automated Testing Generator, I want to give to developers another tool in their arsenal and eliminate the need of manually writing multiple tests.

McConnell and Steve (2004) in their book also found that it will be 10-100x more expensive to fix a problem post-release than if it had already been found in initial stages of development. But exhausting testing of all possible inputs and outputs of big and complex applications is not possible. That is why ATG will make use of the BlackBox testing techniques to test the general behaviour of the application and its functionality as a system, by producing tests using Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transition Testing etc.

10.Aims (max. 100 words): *A statement of the overall aims of the project.*

The main aim of the project is to generate the minimum tests using the BlackBox testing techniques, based on given inputs and outputs of the user, to test the application sufficiently and correctly. The tests should adhere to best practices of the underlining unit test framework. The usage of the ATG will be simple and intuitive, while also providing complex options to give the developers the ability to configure it to their needs if they want to.

11.Objectives (max. 200 words): *A list of specific, measurable objectives, each of which is likely to result in a deliverable. They specify all the work tasks to be undertaken to meet the stated aim. They will vary from project to project, as every project is different, but some examples are provided below. All projects will need to review and report on the literature in a chosen area. Projects might include such general objectives as: To investigate system requirements and produce a Requirements Specification. To research and write a report on good practice in HCI design. To design an interface using the findings from the HCI report. To design and execute a suitable test plan. Or they might be more specific, e.g.: To review and report on how mathematical simulation techniques could be applied to a traffic simulator.*

* Research the topic of Software Testing. Why it’s needed, different techniques and their differences, pros and cons of major testing methods, how to implement tests for Python, what is already developed in the scope of automating software testing and use the research findings to produce a good literature review.
* Learn about best-practice for unit testing and design the application to produce tests that adhere to these practices.
* Using the findings from the earlier research create a backlog or stories, epics, sprints tickets and attach them to Kanban Board
* Produce the diagrams needed for the Software Lifecycle. These includes UML, functional requirements, test cases, system design documentation.
* Developed a mock application to write tests against it. Create manual tests for comparison.
* Developed the different components of Automatic Test Generator, adhere to TDD style of coding and Agile methodology. These will include:
  + Settings and argument parser.
  + GUI using pyQt
  + Output component
  + Error handling
  + Core component
  + Load/Save component
  + Setup component
* Run the ATG against a bigger application, probably a FOSS software, with already implemented test cases. Compare the results. Investigate difference or low coverage of the generated tests.

12.Deliverables (max. 100 words): *A list of your Project’s deliverables with some general description could be found in the module specification.*

First deliverable should include:

* A Literature Review based on Software Testing
* The Functional Requirements for the Automatic Test Generator software.
* The status of the project, including test cases, system design documentation
* Implementation report

Showcase the current implementation to supervisor after the first deliverable.

Final deliverable should include:

• 8000 work limit report which must include:

* Diagrams
* Description of major components
* Description of the development lifecycle
* Story boards/Interface Design
* Design Documentation
* Test plan
* Critical evaluation of your product and your design choices
* Appendices

• The implementation of the Automatic Test Generator

• Viva

• Meeting Notes

13.Resources and Constraints (max. 100 words): *A list of any specific resources that the project requires; for example, hardware and software; access to people or organisations. A list of any known constraints, for example, availability of certain resources.*

* The project will use git with a GitHub repository for code version control, management and review, shared with the supervisor.
* Also, a OneDrive shared folder is created for the project documents.
* My main IDE will be PyCharm from JetBrains.
* Libraries that I might use:
* Unit Testing Frameworks like unittest, py.test, nose2
* Standalone application converter like py2exe or PyInstaller
* pyQt using the OSS of Qt to develop the GUI component
* Functionality libraries like regex, glob, yaml etc.

14.Sources of Information (max. 100 words): *A list of sources you intend to use. These could include: Specific books/journals if you already know of them; Library/Internet; Organisations or individuals you intend to contact.*

Docfiles of the unit test framework.

StackOverflow/Google

Google Scholar for research in software testing.

Udemy/LinkedIn Learning for courses in unit testing and best practices.

15. Risk Analysis (max. 100 words): *What could endanger your project, what will you do if it happens.*

* The project will be version controlled on GitHub with frequent push commits, which make it unlikely, but not impossible, to get lost if the GitHub servers experience catastrophic failure.
* I will be using the professional versions of JetBrains IDEs with my student licence. There is an extremely small risk that my licence will be revoke. In that case, I will change to their equivalent free versions of the IDEs.
* Development will be done in my personal laptop. This is a single point of failure, especially now with Covid19 restrictions, which makes the use of University PCs difficult. If it fails, I will have to try and buy a new one.

16.Schedule of Activities (max. 300 words): *Having defined the tasks to be undertaken in the list of objectives, you need to prepare a Project Plan to show how you intend to carry them out.*

* For the Literature Review/Main Report:
  + Writing the Literature Review will be a continues work. Gathering the different sources, like journals, books, articles etc. will take until the end of November. Afterwards, I will start filtering the sources and writing the review, which will be ready by the end of December.
  + By the 20th of April 2021, the Main Report will be mostly ready since most of the functionalities will be developed by then.
* Documentation:
  + The System Design Documentation which includes UML diagrams, UI design, System Architecture, will be done by end of November.
  + Functional Requirements, like use cases, will be written by 15th of December
  + Test plan, along with test cases, will be done by the end of December
  + Implementation Report demonstrating the application (front-end, back-end) will be done by 10th of January
* Project:
  + Create a Kanban Board to keep track of development time. Populate the Backlog with Epics/Stories etc.
  + Creating a small mock application for testing the functionality of Automatic Test Generator will be done by 10th of November
  + The development of the application will start with core functionalities which will be done by the end of November. Simple inputs and outputs with simple test cases export. Restricted input types.
  + First usable UI done by the 20th of December
  + By January 2021 more functionality will be added to application. Build more complex input parser, analyse functions arguments, apply some BlackBox method techniques like Equivalent Partitioning, Boundary Value Analysis etc.
  + By the end of March add more BlackBox techniques to application, like Decision Table Testing, State Transition Testing, Error Guessing, Comparison Testing, Regression Testing. Add more input types. Create a better error handling, parser.
  + Probably small redesigns for ease of use

17.Student Signature: Lampros Karseras

18.Supervisor Signature: D Smallwood

19.Date: