
**Application for Automated Collection of Test Files
for CSS Class via HTTP and for Local Plagiarism Check:
Testrek**

by
Akshat Tandon

Submitted in Partial Fulfillment of the
Requirements for the Degree of
Bachelor of Mechanical Engineering (Information and Automaton Technology)

at the
Czech Technical University in Prague
Faculty of Mechanical Engineering



© Akshat Tandon
Czech Technical University in Prague
Summer 2017



I. PERSONAL AND STUDY INFORMATION

Surname:	Tandon	First name:	Akshat	Personal no.:	426622
Faculty:	Faculty of Mechanical Engineering				
	Department of Instrumentation and Control Engineering				
Study program:	Engineering				
Study field:	Information and Automation Technology				

II. BACHELOR THESIS SPECIFICATION

Bachelor thesis title (in English):

Application for Automated Collection of Test Files via HTTP and for Local Plagiarism Check

Bachelor thesis title (in Czech):

Aplikace pro automatizovaný sběr testových souborů prostřednictvím HTTP s lokální kontrolou plagiátorství

Elaboration guidelines:

- 1) For the purpose of CSS class, develop a practical standalone application that will download all test files to the teacher's computer upon the request of the teacher(=user) from all test attending students (all test files are supposed to be in a given folder structure - just student user names can be entered manually or via a table obtained from KOS).
- 2) Further, program also the functionality of the app for mutual comparison of all documents, spreadsheets, and html test files for the purpose of plagiarism check (between the attending students only) (simple file comparison + possible use of metadata)
- 3) Enhance the application with a simple GUI or other comprehensible reporting feature that will allow the teacher to quickly visually analyse the results for all students (e.g. 3x similarity matrix for documents, spreadsheets, html files).
- 4) Test your app on the faculty servers in the computer classroom 405a or 405b.
- 5) Properly technically document your work in the thesis.

Literature resources:

MOZGOVOY, Maxim. Enhancing computer-aided plagiarism detection. Saarbrücken: VDM Verlag Dr. Müller, 2008. ISBN 978-3-639-09724-5.

Name and affiliation of the thesis supervisor:


doc. Ing. Ivo Bukovský Ph.D., Dpt. of Instrumentation and Control Engineering, FME

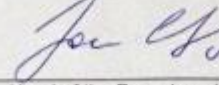
Consultant:

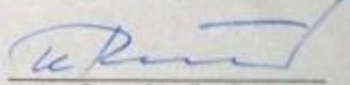
-

Assignment issue date: 19th April 2017 Thesis due date: 16th June 2017

Validity of assignment: -



Thesis Supervisor


Head of the Department


Dean of the Faculty

III. THESIS ASSIGNMENT RECEIVED

The student is aware that the thesis has to be accomplished through an independent and unassisted student's work, supported only by recognized consultations. Literature and other information resources as well as consultants' names have to be acknowledged in the thesis.

19.04.2017	
Date	Student

Annotation List

Authors Name:	Akshat Tandon	
Name of Bachelor's Thesis:	Application for Automated Collection of Test Files for CSS Class via HTTP and for Local Plagiarism Check	
Year:	2017	
Field of study:	Information and Automation Technologies	
Department:	Department of Instrumentation and Control Engineering	
Supervisor:	doc. Ing. Ivo Bukovský, Ph.D.	
Bibliographical data:	Number of pages	49
	Number of figures	10
	Number of tables	2
	Number of attachments	3
Keywords:	python, process automation, string comparison, HTML parsers, plagiarism, Testrek	

Statement

I declare that I have written this thesis independently assuming that the results of the thesis can also be used at the discretion of the supervisor of the thesis as its co-author. I also agree with the potential publication of the results of the thesis or its substantial part, provided I will be listed as the co-author.

In Prague: 16.06.2017

Signature: Akshat Tandon

Abstract

This thesis focuses on analysing and automating the examination process for the ‘Computer Support for Study’ course taught at Faculty of Mechanical Engineering, Czech Technical University in Prague; and running the plagiarism check on answers from the test takers. An application called *Testrek* is built as a part for this thesis which should streamline much of the examination process for the CSS course. The application is written using Python programming language with support from external libraries which are discussed in the thesis. Testrek essentially checks and downloads the files from a publicly accessible network location (public_html folder), where answer files are uploaded by the test takers. The Plagiarism check is run using a library called fuzzywuzzy which in turn uses the Levenshtein distance to calculate the similarity between two strings. At the end, a technical walkthrough of the application is also provided for understanding of the use case.

In conclusion, the required automation of examination process has been achieved. Consequently, the analysis in this thesis open opportunity for further scope of automation and making the application even more dynamic to cover more courses.

Keywords: python, process automation, string comparison, HTML parsers, plagiarism check, Testrek

Acknowledgements

I would like to express my deepest appreciation to Ing. Matouš Cejnek for all the guidance that he has provided during the preparation of this thesis work. Without his persistent help, motivation and immense knowledge in Python, this would not have been as doable as it had been.

I would like to thank Professor Ivo Bukovský for trusting in my ability to work on this thesis topic and for his precious time that he has invested in making this thesis look well presented. Most of all, the Python course taught by him helped me in familiarizing with many aspects of the programming language and its capabilities.

In addition, a big thanks to Kristyna Steidlova from Accenture sro for helping with creating the process maps and for providing her valuable suggestions pertaining to different ways of approaching the automation from her experience and expertise.

Table of Contents

Abstract	v
Acknowledgements.....	vi
Table of Contents.....	vii
List of Acronyms.....	viii
Glossary	ix
Chapter 1. Introduction.....	1
Chapter 2. Process Analysis and comparison of existing tools.....	3
2.1. Process Description.....	4
2.2. Comparison of already available software solutions	4
Chapter 3. Proposed Solutions	6
3.1. Standardization of the process.....	6
3.2. Possible Solutions.....	6
Chapter 4. Technologies and Methodologies.....	10
4.1. Development Environment	10
4.2. Programming Technologies.....	11
4.2.1. Why Python?	14
4.3. External Python libraries.....	14
4.4. Development and Testing Methodology	17
4.5. Plagiarism Check.....	19
4.5.1. Available plagiarism check methods in Testrek	20
Chapter 5. Technical walkthrough	22
5.1. Application runtime	22
5.2. Testrek components	25
Chapter 6. Conclusion	33
Appendix Testrek Application.....	35
Download Success Report	36
Plagiarism Check Report.....	37
References.....	38

List of Acronyms

CTU	Czech Technical University
CSS	Computer Support for Study
GUI	Graphical User Interface
IDE	Integrated Development Environment
PyPI	Python Package Index
TDD	Test Driven Development

Glossary

Python	An Object Oriented and interpreted Programming language
Script	Programs written for a special run-time environment
Open Source	The source code that is made available to the public under a license to study, change, and distribute the software to anyone and for any purpose
Database	a structured set of data held in a computer,
Web Hosting	A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web
PEP8[1]	Python code style conventions
Object Oriented Programming	Computer programming in which we can define not only the data type of a data structure, but also the types of operations (functions) that can be applied to the data structure
Python Interactive Shell[2]	Python's command line utility
snippet	a small piece or brief code
Recursion	A common computer programming tactic is to divide a problem into sub-problems of the same type as the original, solve those sub-problems, and combine the results.
Hash function[3]	A hash function is any function that can be used to map data of arbitrary size to data of fixed size.

Chapter 1. Introduction

The major motivation for this thesis comes from the fact that most of the work surrounding the examining of students is still done manually. This thesis introduces the need, scope and application that is built for the automation of the examination process for reducing the manual work. The software solution described and developed as a part of this thesis is tend to be designed in a way that it utilizes and requires the least resources and infrastructure for its operation. The application is made as dynamic as possible hence it can be used outside the set scope (i.e. for Computer Support for Study course[4]), if the process of examination can be standardized for that course or subject. Different methods of approaching the automation are also described along with their benefits and shortcomings.

For the sake of simplicity, CSS course is used as an example all through this thesis, to analyze and demonstrate the challenges and solution proposals to different aspects of such an automation process.

When conducted manually, the whole examination process procedure is taken care of manually with the help of up to three or more people. From the preparation of question paper to reviewing the answers from students (most of the times in an unorganized way) and assigning grades. If a course or subject is registered by many students (which in fact is a case for the CSS course), for e.g. in order of hundreds then the whole process becomes even more cumbersome and hard to handle, leading to the slowdown of the review process and requirement of even more manual efforts.

The aim of this thesis is to provides a good understanding of the examination process where most of the work is done manually, along with ways to approach the automations of such nature. The problem that has been addressed

in this thesis is mostly concerned with the organization and automation of the answer reviewing system and aid the plagiarism check. Also, a summary of how to achieve operational excellence by the deployment of small and cost effective automations is made part of this thesis.

The software solution *Testrek*, developed mainly in Python as part of this thesis can help in organization of reviewing and running a preliminary plagiarism check on answers from the students taking part in the examination. The designing of the solution is dynamically approached so that the testing of any kind can use this solution if it can be standardized as mentioned in the chapters later. This thesis can provide a base to conduct further research and development on automating the examination sub processes which are out of scope of this bachelor thesis.

There are information systems that already exists and can provide a potential solution for achieving the automation, if modifiable and extensible. For example, *Moodle*[5] is an open-source learning platform which is used by CTU to share study material with students. Although by default, there is no possibility for students to upload their answers during the examination or for teachers to run a plagiarism check, it can be extended in order to do similar tasks that *Testrek* would do but the efforts required for achieving the same results as *Testrek* would require more time, resources, expertise in other front and back end programming languages. Moreover, for integrating the extensible application to current *Moodle*, an extensive Black and White box testing[6] will be required, ranging from regression and system tastings to integration testing. It is for all these reasons that it became necessary to write an autonomous application.

The information in this thesis is ordered in way so that one starts by understanding the aim and challenges faced due to the current process, followed by a description of the solution proposals and technologies used to develop the solution itself.

Chapter 2. Process Analysis and comparison of existing tools

Process Analysis constitute an important part of this thesis as it helped in understanding the different aspects of the problem which are tried to be solved while designing the software solution described in *Chapter 3*.

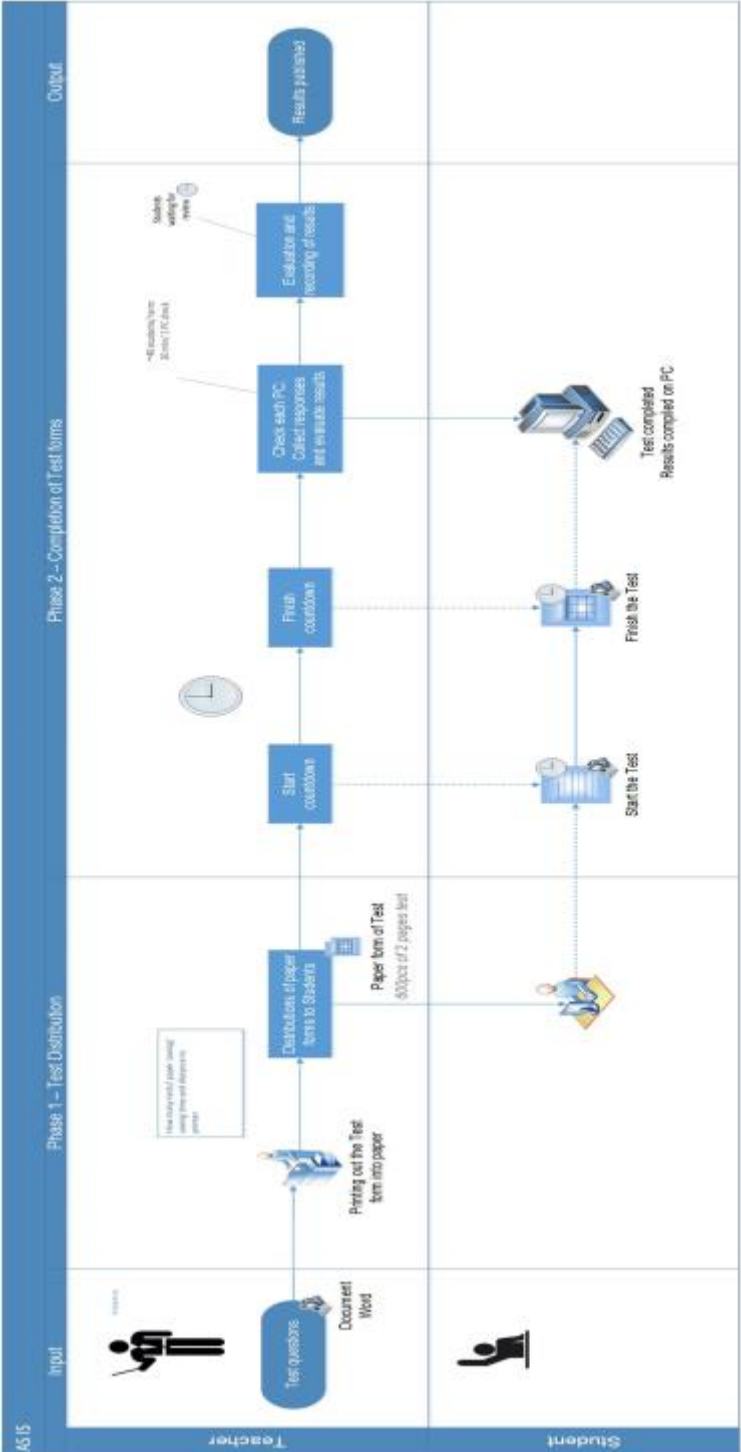


Figure 1 - Current Examination Process workflow

2.1. Process Description

When conducted manually, the process map of whole examination process is explained using a swim lane chart on the top. The chart shows the current process procedure, with the estimations of work and in some cases time required for a sub process to complete. Next chapter contains further information about the scope of the software solution developed for this thesis and a rough estimation of efforts required to automate the sub processes which are not included in the scope.

2.2. Comparison of already available software solutions

There are plenty of tools available in the market which can be used to automate the testing of students and this section contains a summary of the features for some of the tools that were found to be the closest alternatives of *Testrek*.

1. Moodle

Moodle is an open source learning platform, which provides a robust, secure and integrated system under one tool for educational institutions to create a personalized learning platform. Moodle is considered an alternate for *Testrek* because it is easily extensible, light weight, open source and already implemented at the Faculty of Mechanical Engineering. Technologies used to build *Moodle* are PHP and MySQL. It utilizes Apache servers to run itself[7].

As *Moodle* is essentially a learning platform, building a testing system would require to interact many other modules which may make the whole system unstable. It is a server application and would also need some maintenance from time to time in contrast to *Testrek* which is a standalone application. Additionally, an aggressive testing scheme would be necessary

before the test module can be implemented in the main *Moodle* application system so that it is fail safe at all times.

It is also required to have a very good understanding of Moodle's object model before one can start developing on it. The implementation of the whole solution would also look very different on *Moodle* when compared to *Testrek*.

2. Quiz Works

Quiz Works is a subscription based online examination system which can be used to create online testing questions for test takers. It is not open source so extending the application is not an option. It does support a few third-party integrations and web-hooks but none of them are related to the scope of this thesis. Additionally, it does not support plagiarism check. As it is subscription based, the version with full blown features cost \$99 per month, which makes it an expensive option[8].

On the good side, this online tool provides the teacher with test analysis which can help in providing the teachers with an insight into the performance of the test takers in certain areas and improve the study content for those in the upcoming semesters.

3. Easy Test maker

Easy Test Maker is another subscription based online examination system which can be used to generate question papers for testing but it does not support plagiarism check and it is only available online. There are several options available to generate different types of questions which can later be automatically graded[9].

This tool is also not open source which restricts the users to extend its capabilities in contrast to *Testrek* or even *Moodle*.

Chapter 3. Proposed Solutions

3.1. Standardization of the process

Before designing any solution for automating the examination process, it was necessary to standardize it. This in fact proved to be crucial while searching for different ways to approach the solution. The standardization helped in deciding the best possible way to approach the designing of the software solution, in the limited time frame. To make the solution fail safe it was necessary to set many rules for different parts of the process, but the challenges was to still maintain its dynamical nature.

After a thorough analysis, the two most qualified solutions which can be build are listed below with the requirement for standardisation, complexity, maintainability, infrastructure and support. [10]

3.2. Possible Solutions

1st Solution (Testrek) – Preferred

Basic Requirements for this solution to run:

- a. Access to the internet
- b. Disk space of about 50 Mega Bytes

This solution which is essentially a combination of scripts written in Python, is built as a part of this thesis. This solution was preferred because the requirements for running this application were very low and affordable. As the scripts are simple to understand, it would not require a lot of time to modify if the need be.

This solution also does not demand a lot of infrastructure hence it can run on most of the computers of today's standard. Although it does not have a very user interaction to the of GUI, it is still easy to use and maintain. The challenging part for this solution is strictly standardizing many parts of the examination process, for example, where data is expected from the user.

The swim lane chart shows the process workflow if application like *Testrek* will be implemented for automation. A more thorough description of this solution can be found in the upcoming chapters.

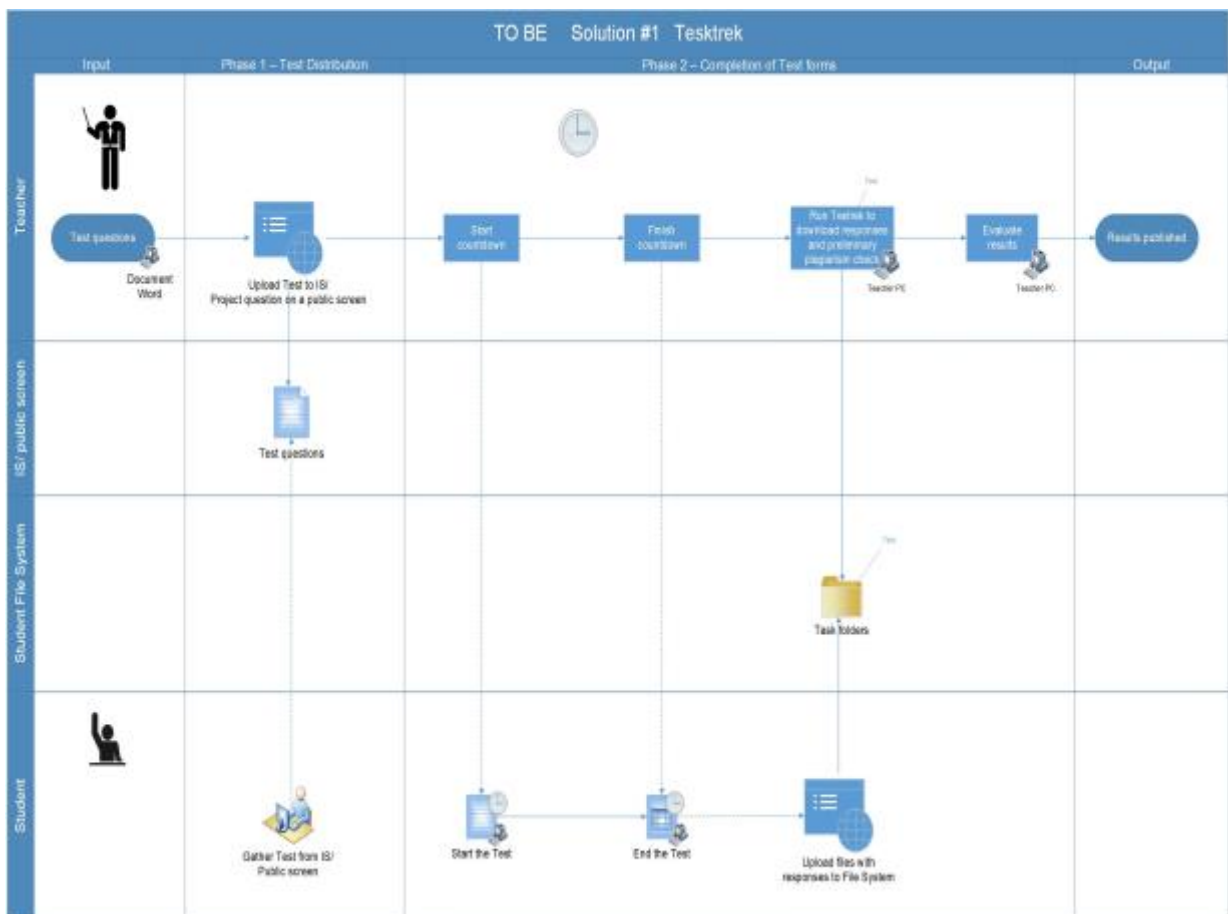


Figure 2 - Workflow for Testrek

2nd Solution – Not preferred

As this solution will utilize a server, it will have a client side and the server side. Basic Requirements for this solution to run:

On client side:

- a. Internet Connection and a modern web browser

On server side:

- a. a database
- b. a server supporting Python
- c. an app hosting platform
- d. maintenance and support

Building this application solution would require a robust web framework for Python, e.g. Flask[11], Django[12] etc. It would utilize a database to store the information and a server to operate itself. Furthermore, the development process would require a lot more time and resources in comparison to the 1st solution.

Though the requirements are on the higher end when compared to the 1st solution, this solution which is essentially a web application will be more robust in its operation and delivery. This application would run on a web browser with an intensive GUI, hence will be more interactive for the users. One other essential characteristic of this solution would be the fact that it will not require as much standardization of the process as does the 1st Solution requires. Several validations can be run on the data as and when it is received, hence diminishing the need for much standardization.

The swim lane chart below shows how the examination process would look like if a web application, written for example in python's web framework Django, can be implemented to automate the examination process.

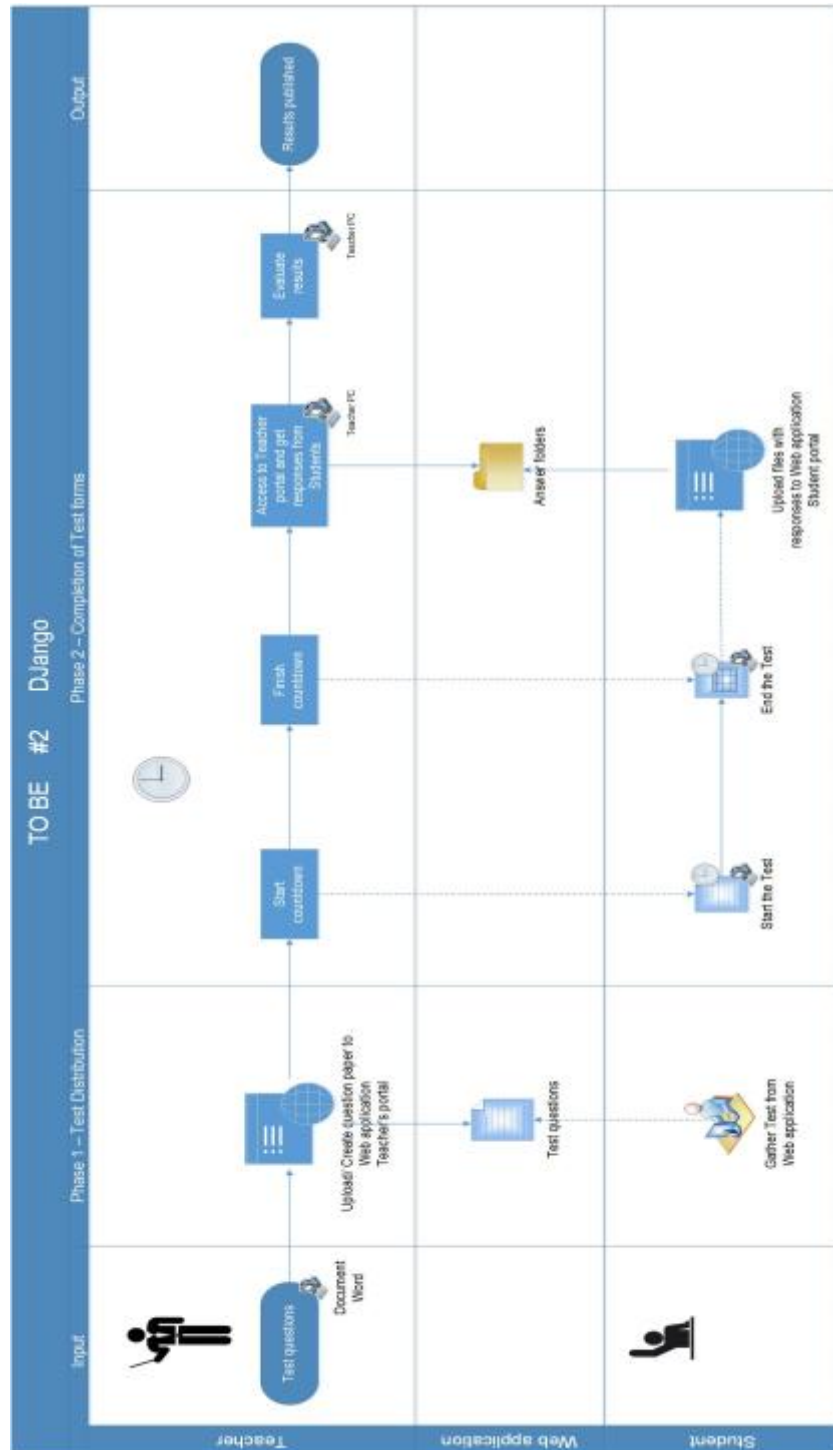


Figure 3 - Workflow for a web application

Chapter 4. Technologies and Methodologies

This chapter guides through different technologies and methodologies used, from development to the deployment of the application solution. The sub sections of this chapter consist of information about the platforms used for development, environment settings, programming technology and the libraries used.

4.1. Development Environment

The solution had been built on a machine running MAC OS and the IDE or integrated Development Environment that was used to build the software solution is called *PyCharm Edu 3.5*[13] provided by JetBrains S.R.O..

This IDE was chosen as it has a great support for Python development and is an open-source software. It is best known for intelligent code completion, on-the-fly error checking and quick-fixes and easy project navigation. It helps keep quality under control with PEP8 checks, testing assistance, smart refactoring, and a host of inspections.

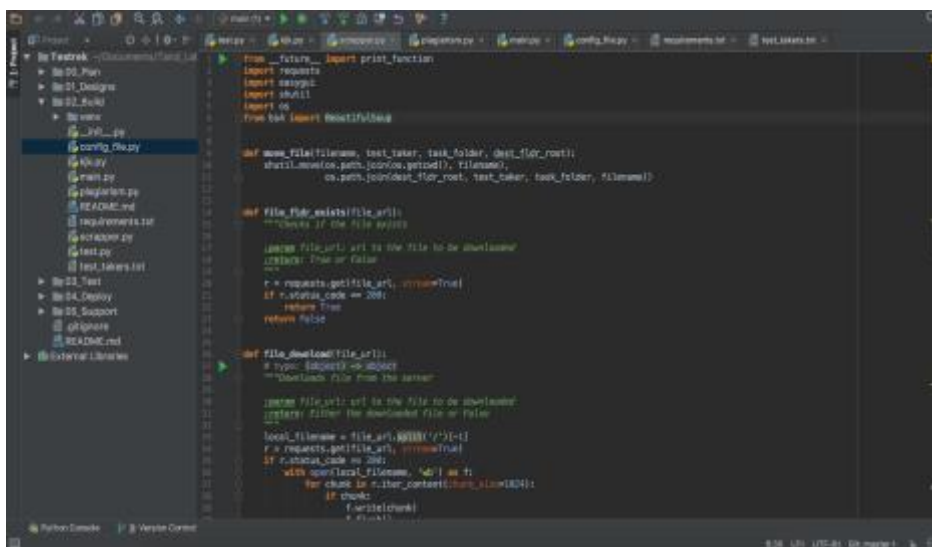


Figure 4 - PyCharm Sample Workspace (adopted from PyCharm)

4.2. Programming Technologies

The software solution *Testrek* is purely written in the programming language called Python. Python is an interpreted, multi-purpose programming language that can be used to write web applications, GUIs, scripts and much more. It is strongly and dynamically typed with focus given to its readability and productivity. With an immense support from the community around it which builds a great range of libraries, it has proved to be a powerful language for scientific use and mathematical modelling. It is a self-contained object oriented programming language that has an interactive shell, strong introspection, cross platform capabilities and a variant for specific use like CPython, JPython, IronPython etc.[14]

The versions of Python used for building and testing the software solution for this thesis are Python 2.7.10 and Python 3.5.2.

Hello World in Python

Writing “Python” in the command line starts the Python interactive shell which can be used to write python commands.

```
#!/usr/bin/env python
print "Hello World!"
```

Indentation is necessary

Unlike most other programming languages, Python cares about the indentation and structure of the code. A sample is shown below:

```
#!/usr/bin/env python
for i in range(1, 10):
    print ("I am number " + str(i))
    if i == 9:
```

```
print ("9, here again!")
```

Comments

Comments in Python are written as shown below. Any string that is not assigned to a variable or function is regarded as a comment.

```
# One line comment
"""
This is a multi-line
comment.
"""
"Any string not assigned to a variable is a comment"
```

Data Types

Python has built in support for primitive data type like strings, Numbers, Null, Booleans, Lists, Tuples and Dictionaries[15]. Python dynamically assign the data type after a variable is initialized, hence declaration of variable is not included in Python.

```
# Strings
address = "This is a string."
address_long = """This is a
long string."""

# Numbers
# Integers
age = 9
year = int("2010")
#Float
pi = 3.14159

# Null
data = None

# Booleans
is_Python = True

# Lists
# initialisation
names = ["Charlie", "Brown", "Chris", 59, True]
```

```

# Appending
names.append("Martin")

# Tuples
# Can't be changed after initialisation
names = ("This", "is", "final", true)

# Dictionaries
# initialisation
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
# Update
dict.update({
    'Gender': 'Female',
    'Hobby': 'Reading',
})

```

Control Flow

Python has support for conditionals, for and while loops, and list comprehension. An example of list comprehension is mentioned below:

```

# List comprehension
div_by_two = {x for x in range(10) if (x%2 == 0)}

```

Classes and Functions

Classes and Functions makes up for an important part of any object-oriented programming language[16]. Classes can inherit from other classes and ultimately from “object” class which is the top-level class in Python from which all classes inherit. Functions can accept argument or not. In Python, a function can be defined as below:

```

# Funtion that excepts an argument
class Any_name(object):
    def foo(i):
        """Function documentation"""
        remainder = i % 5
        if (remainder!= 0):
            return remainder

```

4.2.1. Why Python?

Python is undoubtedly an easy to use programming language with a great community that generously promotes and supports it. Because of the reliability that frameworks like Django, Flasks and Pylons etc. provide, they are being used as a primary platform for development for many software products. Python has no interfaces or real scoping of functions and methods, which lets developer concentrate more of the logic of the application than the syntax of the code itself. Due to all the above reasons, Python proved to be the right choice for developing *Testrek*.

4.3. External Python libraries

Python allows adding external modules (libraries) to a project. Packages are essentially a collection of dynamically written classes with variables and function, which can be re-used in another project. The keyword “import” is used to create a reference to these modules in a project and then the functions from these classes can be used.

```
# importing the whole datetime module and creating an alias
dt
import datetime as dt
# importing only the classes timedelta and date from the
datetime
from datetime import timedelta, date
```

An exhaustive list of all the packages publicly available for Python is available on PyPI[17].

Apart from the several packages that come by default with Python, a few other were also used while developing *Testrek*. I brief introduction to each one of them is mentioned in this section.

Requests (v. 2.14.2)

Requests is a HTTP library for Python, which send HTTP/1.1 requests without the need for much of work that is required to be taken care of while sending a HTTP request. For example, there is no need to manually add query strings the URLs or to form-encode the POST data.[18] Connection pooling is also taken care of in the library itself which reduces the need to custom write the related code again and again.

Testrek utilizes requests library to download files related to each task from the user's filesystem (found under public_html folder) on the University server.

Easygui (v. 0.98.1)

EasyGUI is a simple yet robust GUI written in Python. It is not event driven, instead all the GUI interactions are invoked by simple function calls. This GUI library is used to present dialogs with information during the runtime of *Testrek*. It saves the user from knowing anything about tkinter, frames, widgets, callbacks or lambda, which are core to it. It runs smoothly on Python 2 and 3 and does not have any dependencies.

Fuzzywuzzy (v. 0.15.0)

Fuzzywuzzy is a package used for the string comparison. It uses Levenshtein Distance to calculate the differences between the sequence of strings. It is compatible with python 2.4 or higher. It utilizes difflib library that comes bundled with Python and uses the package python-Levenshtein to deliver results even faster. In *Testrek* python-Levenshtein package is

used along with `fuzzywuzzy` in order to get results as quickly as possible[19].

Tqdm (v. 4.14.0)

Tqdm package is available for Python 2.6 and higher is used to create a progress bar from the number of iterations. For implementation, it must simply wrap with the iterable. The snippet below one can see the implementation in a real-time scenario[20].

```
from tqdm import tqdm
for i in tqdm(range(500)):
    ...
```

Beautifulsoup4 (v. 4.6.0)

Beautifulsoup is an extensively used python package for parsing through a web page. Since its emergence in 2004, it has been under constant development and the latest version provides some great features and robust runtime performance. It is built upon an HTML or XML parser, providing extensive features to iterate, search and modify the parsing tree[21].

In *Testrek*, it is mainly used to check for the existence of the files on the web url before *requests* library can be used to download that file. This was necessary to be done while providing the right results in the download success report.

4.4. Development and Testing Methodology

Most of the development work on *Testrek*, was conducted by me and it was necessary to choose the right development techniques. The list of possible development methodologies was already shorten down to only a few because of this fact. The timeframe available for the development of the application was limited as well and it served as the second condition to limit the list down to one. For all these reasons, test-driven development was chosen. All through the development life cycle, it was necessary to design tests and then write the function definitions. Test driven development[22] has its own challenges and but it did fit nicely for a small project like *Testrek*. The illustration below provides an overview of how the development was approached at different stages

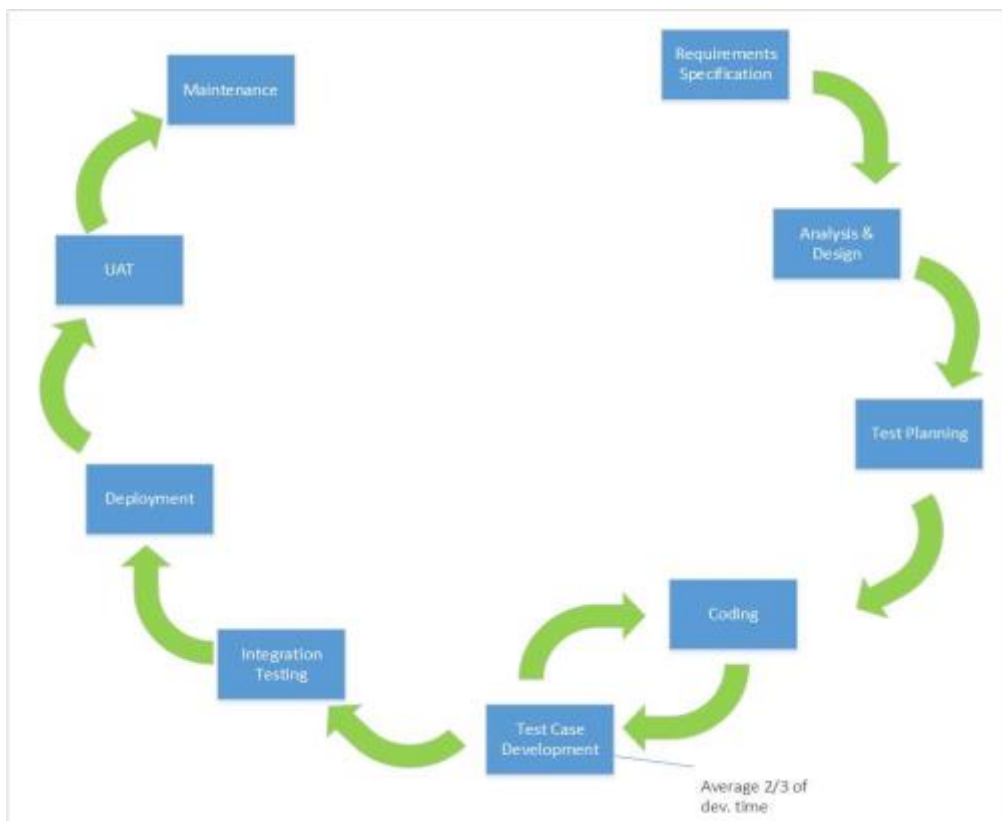


Figure 5 - Development life-cycle

The following sequence of steps are generally followed in a test-driven development projects[23]:

- Add a test
- Run all tests and see if the new one fails
- Write some code
- Run tests
- Refactor code
- Repeat

It is true that TDD slows down the development but once one get into the loop it becomes quite easy. It was important to produce better designs, allow easy and safe refactoring and slowly increase the test coverage in order to adapt to this methodology and take its benefits.

4.5. Plagiarism Check

Testrek incorporates a feature to run plagiarism check on answers from every test taker against the answers from every other test taker. For this purpose, *fuzzywuzzy* package is used which can provide similarity ratios processed for different types of string comparisons.[24]

Simple Ratio

```
>>> from fuzzywuzzy import fuzz
>>> from fuzzywuzzy import process

>>> fuzz.ratio("we are here, finally", "we are here,
finally!")
97
```

Partial Ratio

```
>>> fuzz.partial_ratio("this is a test", "this is a test!")
100
```

Token Sort Ratio

```
>>> fuzz.ratio("fuzzy wuzzy was a bear", "wuzzy fuzzy was a
bear")
91
>>> fuzz.token_sort_ratio("fuzzy wuzzy was a bear", "wuzzy
fuzzy was a bear")
100
```

Token Set Ratio

```
>>> from fuzzywuzzy import fuzz
>>> from fuzzywuzzy import process

>>> fuzz.token_sort_ratio("fuzzy was a bear", "fuzzy fuzzy
was a bear")
84
>>> fuzz.token_set_ratio("fuzzy was a bear", "fuzzy fuzzy was
a bear")
100
```

Fuzzywuzzy utilizes the *Levenshtein Distance*[25] to compute these ratios. Levenshtein distance (LD) is a measure of the similarity between two input strings. The distance is the number of deletions, insertions, or substitutions required to transform one string into another. The greater the Levenshtein distance, the more different the strings are.

Let's take two identical strings. If x is "test" and y is "test", then $LD(s,t) = 0$, as no transformations are needed.

If s is "rent" and t is "rant", then $LD(s,t) = 1$, because one substitution (change "s" to "n") is sufficient to transform s into t.

Levenshtein distance are used in the following fields:

- Spell checking
- Speech recognition
- DNA analysis
- Plagiarism detection

4.5.1. Available plagiarism check methods in Testrek

In *Testrek*, there are two available options to run plagiarism check between answer files from different students. One being the simple string check and other the hash check. These options can be toggled in the *config_file.py* which is explained in the next chapter.

When a simple string check is selected then the content of each file is converted into one long string and is compared to another long string created from the content of answer files for the same task from other students. While on the other hand, when hash comparison is selected then

whole of the file is run through a hash function to convert into a *md5* hash. The snippet below demonstrates how the conversion is done in *Testrek*.

```
import hashlib
import codecs

s_buf_raw = fp.read() # fp is an answer file
s_buf = s_buf_raw.encode('utf-8')
hasher = hashlib.md5()
hasher.update(s_buf)
s = hasher.digest()
```

Chapter 5. Technical walkthrough

This chapter describes the working of *Testrek* application, different standardized inputs that it requires for its operation and the output files it generates. As described in chapter 2, *Testrek* need standardization of certain parameters that it takes before starting the main process of downloading the files and running the plagiarism check.

5.1. Application runtime

Minimum requirements

Testrek is designed to run on Linux, Mac and Windows. Below are the minimum system requirements for *Testrek* to run:

Table 1 System Requirements

System Requirements	
Operating system	Windows 7 or higher, Mac OS X or higher, Linux
Hard drive	8 MBs for the application (<i>Testrek</i>) + (number of students X number of tasks X 3) MBs disk space for answer files from students
Python	Python 2.7 installed on the local machine

Standardized inputs

- The students are required to store the answer files in folders under `public_html`, which should be named in a certain way. An example for uploading the files for Task 1 should be done as follows:

student folder → *public_html* → *Task1* → (Task file here; of any format)

- The application takes an input a text file with names of all the test takers put on separate lines. These could be short user names as “tandoaks” or longer user name as “akshat.tandon”.

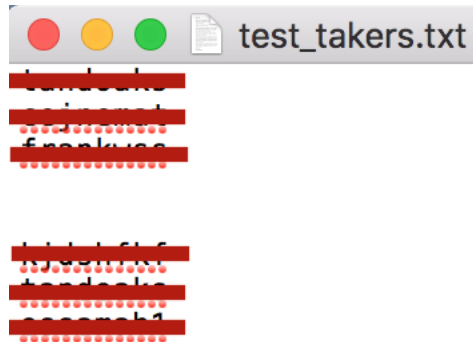


Figure 6 snapshot of file containing names of test takers

High Level overview

The working of the application is as follows:

- At first, test takers upload answers to every task into a task folder named for example as “Task1”, “Task2” etc., directly under the public_html folder. Test takers can name the file in any manner. Any file type can be downloaded using *Testrek*.
- When the test is over, the test supervisor can run the “RUN.py” file present in the application directory using “Python Launcher”.
- While the script is run, it will show a file picker dialog box where the instructor should select a text file with name of the test takers.
- A progress bar can be seen in the Python Launcher command line representing the progress of downloading of the files. After the files for every test taker is downloaded, a pop up window appears which asks if the application should run a plagiarism check or end itself.

- If yes is clicked, the plagiarism check is run and reports are generated.
- The download success reports in csv format is stored in “Reports” folder and answer files in the “Answers” folder. A report for the plagiarism check in html format is generated and stored in the application directory.

Outputs files

There are three main outputs of running the script.

- Answer files from test takers, which by default are downloaded in the application folder under folder called “Answers”. An example is shown below:

Testrek →Answers →Answers_current_date&time →tandoaks →Task1, Task2...

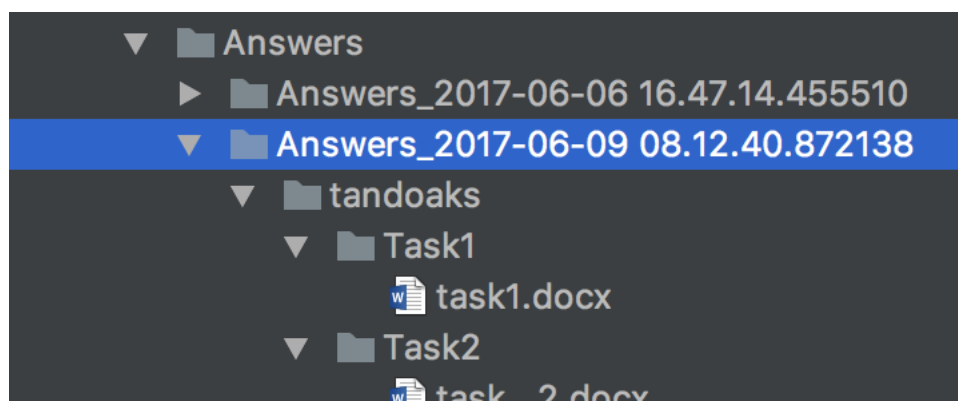


Figure 7 Screenshot of Answers Folder

- File download success report, which by default is downloaded in the application folder under folder called “Reports”. An example is shown below:

Testrek → *Reports* → *Report_current_date&time* → *report_date&time.csv*

NOTE: An example of report can be found attached in the appendix.

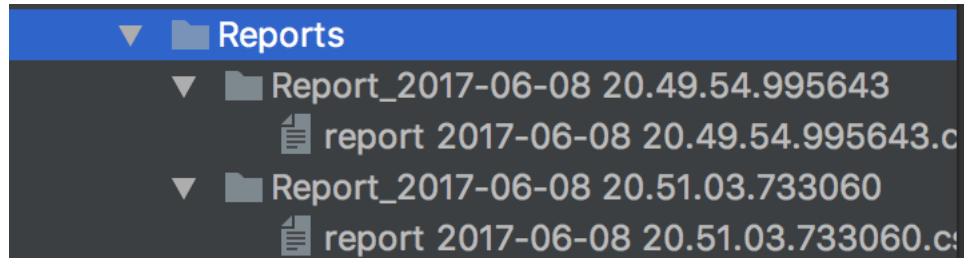


Figure 8 Screenshot of Reports folder

- Plagiarism report, which by default is downloaded in the application folder directly and is replaced each time the plagiarism check is run. An example is shown below:

Testrek → *plagiarism_check.html*

NOTE: An example of the plagiarism check can be found attached in the appendix.

5.2. Testrek components

The application *Testrek* is written in a few several modules to so that the code be more readable and modifiable if necessary. In this section, different modules of the application and their constituent are discussed.

“__init__.py” module

This is an empty file in the application directory and is used to mark directories on disk as Python package directories. If it were not present, then python cannot import the sub modules in other python files in the application[26].

“config_file.py” module

This is a configuration or setting file for the *Testrek*. It greatly contributes to the generality of the whole application by allowing users to change certain parameters. In the following snippet, it can be seen which all parameters are available to be modified and their meanings.

```
import os

#####
##### DOWNLOAD FILE SPECIFIC #####

# Do not modify this parameter
curr_dir = os.getcwd()

web_url = "http://users.fs.cvut.cz/"

answer_folder = os.path.join(curr_dir, "Answers")

tasks_folders = ["Task1", "Task2", "Task3"]

rep_dir = os.path.join(curr_dir, "Reports")

#####
##### PLAGIARISM CHECK SPECIFIC #####

hash_check = False

type_of_check = "Simple Ratio"
```

web_url: Specify the root url where the student folders are placed.

For e.g.: `web_url = http://users.fs.cvut.cz/~`

answer_folder: The directory where downloaded files are stored.

For e.g.: `answer_folder = os.path.join(curr_dir, "Answers")`

tasks_folders: Specify the folders on the web file-system where the answer files should be downloaded from. For e.g.: `tasks_folders = ["Task1", "Task2", "Task3"]`

type_of_check: Type of string comparison. Set to "Simple Ratio" for simple ratio. All available options: "Simple Ratio", "Partial Ratio", "Token Sort Ratio", "Token Set Ratio".

rep_dir: The directory where the reports generated regarding downloading of the files should be stored.

hash_check: Run plagiarism check on hash or whole file content. Set to True if check based on hash or False for check based on content. Hash check takes much less time as compared to the other option.

type_of_check: Type of string comparison. Set to "Simple Ratio" for simple ratio. All available options: "Simple Ratio", "Partial Ratio", "Token Sort Ratio", "Token Set Ratio".

“plagiarism.py” module

The plagiarism.py file consists of definition of the function `retrieve_folder_content`, which is used to retrieve contents of a folder. This function takes in two parameters *src_path* and *file_check*, where the first is the absolute path to the directory where contents are required to be checked and the latter is for checking if to check the sub-directories or files in the *src_path*. It was convenient to write a function like this as there was a need of subsequent retrieving of the contents in a directory for checking the files.

“scrapper.py” module

The scrapper.py file consists of definition of the functions that checks for the answer files in the student folder on the server filesystem

and then download the files. Other than that, there are some function definitions like `move_file` to move downloaded files under the right student and task folders and `test_takers` function to get the usernames of the test takers.

The `test_takers` function uses *easygui* file picker which restricts the selection to only text file containing the names of the test takers on separate lines. This function also incorporates several validation checks before a list of test takers is returned for the main program to loop through in order to download the files from the server file system i.e. `public_html` folder for each test taker. For an expected run and end of the application, *Exceptions* like no text file selected and other unexpected errors are taken care of in the function definition itself as shown below:

```
except Exception as e:
    if file_name is None:
        easygui.msgbox("No text file with usernames selected!"
                       + "\n" + "Script exited", "Error")
        raise SystemExit("No text file with usernames
                          selected!")
    else:
        easygui.msgbox(e.message, "Error")
        raise
```

The `file_fldr_exists` function uses *requests* library to check the status code that is returned while trying to get the file from the `public_html` folder. It was necessary to do to decrease the time complexity of the of *Testrek* during runtime. Below are some status codes that can be returned by the server[27]:

Table 2 HTTP/1.1 Response Codes

Code	Description
100	Continue
200	OK
201	Created
202	Accepted
400	Bad Request
404	Not Found
500	Internal Server Error

The `file_download` function defined below is used to download the files from the server. The `requests.get[28]` method returns the response from the URL that is provided as a parameter. The function proceeds if the URL can be reached by checking the status code of the response method.

```
def file_download(file_url):
    """Downloads file from the server

    :param file_url: url to the file to be downloaded
    :return: Either the downloaded file or False
    """
    local_filename = file_url.split('/')[-1]
    r = requests.get(file_url, stream=True)
```

```

if r.status_code == 200:
    with open(local_filename, 'wb') as f:
        count = 0
        for chunk in r.iter_content(chunk_size=1024):
            count += 1
            if count <= 3000:
                if chunk:
                    f.write(chunk)
                    f.flush()
            else:
                return 0
        return local_filename
    pass
return 0

```

“test.py” module

Test.py module imports `TestCase` from `unittest`[29] library that comes standard with Python. The function definitions found in `test.py` module runs the unit test on some major functions used in *Testrek*. Running these tests can guarantee that the functions related to downloading the files, found in the `scraper.py` module are running as expected and can connect to the server to download the answers files.

An example of how the unit tests are approached can be found in the snippet below. When run, the `TestDocxFileDownload` function mimics the normal working of application by downloading a test file from `public_html` folder of server file system which is publicly accessible and then `self.assertIs` is used to verify that the downloaded files exist in the *Testrek* root directory.

```

from unittest import TestCase
from .scraper import file_download, filenames_from_html
from .plagiarism import *
import os

class TestDocxFileDownload(TestCase):
    """

```

```

Modify the url parameters to the file_download to test.
Use url to the file for the user account to which you have
access.
"""

BASE_DIR = os.getcwd()

def test(self):
    base_dir = os.getcwd()
    if os.path.isfile("test.docx"):
        os.remove(os.path.join(base_dir, "test.docx"))
    else:

file_download("http://users.fs.cvut.cz/~tandoaks/test.docx")
    self.assertIs(True, os.path.isfile("test.docx"))

```

“RUN.py” module

RUN.py module serves as a main entry point to the *Testrek* application. It joins together all the functionality of *Testrek* defined in other modules described above. This module consists of `main` method under which most of the logic of *Testrek* lives.

The `main` method is called when *RUN.py* is exclusively run i.e. when `if __name__ == '__main__'` [30]. The *main* function creates the directories for Answers and Reports with correct date and time stamp wherever necessary and then it looks calls the `file_fldr_exists` and `file_download` functions to check and download the answer files from the `public_html` folder and generates the download success report at the same time. *tqdm* library is used to show the progress of the runtime to the user in the console.

The logic behind the plagiarism check is programmed right after the downloading of all the files are done. It was necessary to deal with different encodings because of the codec error that was thrown on different operation

systems. It was taken care of by encoding and decoding the strings using the following snippet.

```
import codecs
types_of_encoding = ["utf-8", "cp1252", "cp850", "utf8"]
for encoding_type in types_of_encoding:
    codecs.open(Ans_file, encoding=encoding_type,
                errors='replace') as fp:
        # Other logic of plagiarism
```

Chapter 6. Conclusion

The aim of this thesis was to design a feasible way to automate the examination process for the *Computer Support for Study* subject, taught at the Faculty of Mechanical Engineering at Czech Technical University. From the among the existing software technologies available in CVUT, *Moodle* qualified the most to handle such an implementation of automation. But it was concluded based on the reasons provided in the introduction that it will be not be efficient to design and implement this automation in *Moodle*. Based on the analysis of the current examination process, two solutions are proposed in Chapter 3, of which one is developed as a part of this thesis.

Testrek application written in Python has been designed in accordance to the scope set in the thesis requirements. It downloads the files answer files from each test takers' *public_html* folder to the local machine and runs a plagiarism check on each task file from test taker against every other. Additionally, the application is made very dynamic and can be used in other similar examinations too. *Testrek* also deals with several exceptions that may occur during the runtime, which gives the user an opportunity to easily debug the application if the need be. It also satisfies the need for platform independency as it can run on most major platforms as far as Python 2.7 is installed on that machine.

Testrek provides the user with a config file where they can modify several settings based on the requirements of the situation. After every run, it outputs: answer files from test takers, a download success report in csv format and a plagiarism check report in html format.

Examination process analysis and proposals, that can be found in chapter 2 and 3 respectively, can serve for further research on standardizing and automating the parts of the examination process which are not included in the scope of this

thesis. In fact, an even better and robust solution can be developed using Django but it will require much more resources, time and eventually maintenance.

Appendix

Testrek Application

The attached zip file below contains the *Testrek* Application. One can download and unzip the folder anywhere on a machine running Linux, Mac or Windows with Python 2.7 installed, and use RUN.py file in the directory to run the application. No installation of any kind is required as far as *Testrek* as an application is concerned.

The application directory has the following structure and these python modules should not be moved, deleted or modified without prior knowledge of the working of this application. A text file containing the usernames of all the test takers can be stored anywhere on the computer.

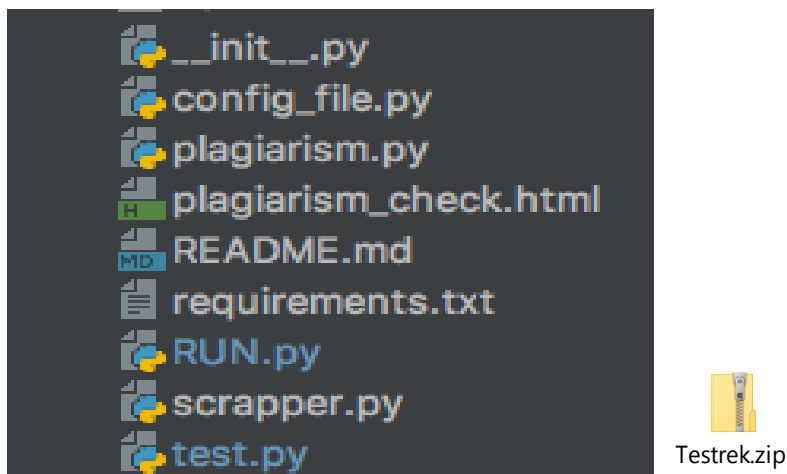


Figure 9 Testrek directory structure

Download Success Report

The “Report” folder is created after running *Testrek*, stores the download success report (in CSV format) under the Report folder with specific date and time stamp. One success report file is generated on every run. The report contains the following columns:

- **Test Taker:** Test takers name
- **Tasks:** Task number
- **Status:** Status of file download
- **File Name:** Name of the downloaded file for that particular task

There can be three possible statuses:

- **Files successfully downloaded:** File has been successfully downloaded and stored under respective *Answers* folder.
- **Can't access url or user folder not found:** The file download has failed either because the user does not access for access to the server file location (i.e. `public_html`) is denied.
- **Folder named Task# not found:** Student folder can be accessed but a particular task folder cannot be found under `public_html`.

An example of download report file is attached.



report 2017-06-07
13.12.54.436867.csv

Plagiarism Check Report

Plagiarism check report is stored in the application root directory and is overwritten each time the *Testrek* is run. The plagiarism check report is generated in the html file format. The cells with red color in the table highlight that there is that similarity between files are over 80%. A screenshot of the plagiarism check report can be seen below.

Plagiarism Check Results

Task1

Test Takers	danedane	dickdick	cejnemat
shipship	100	100	10
tandoaks	100	100	10
toiletote	10	10	100
corncorn	0	0	100
rickrick	10	10	100

Task2

Test Takers	danedane	dickdick	corncorn	cejnemat
shipship	100	100	18	18
tandoaks	100	100	18	18
toiletote	18	18	100	100
rickrick	18	18	100	100

Figure 10 Plagiarism check Report



plagiarism_check.html

References

- [1] PEP 8 -- Style Guide for Python Code. *Python.org* [online]. [vid. 2017-05-28]. Dostupné z: <https://www.python.org/dev/peps/pep-0008/>
- [2] 2. *Using the Python Interpreter — Python 2.7.13 documentation* [online]. [vid. 2017-06-17]. Dostupné z: <https://docs.python.org/2/tutorial/interpreter.html>
- [3] *Hash function* [online]. 2017. Dostupné z: https://en.wikipedia.org/w/index.php?title=Hash_function&oldid=784087124
- [4] BUKOVIVO. Computer Support for Study (E372041). *Department of Instrumentation and Control Engineering* [online]. 3. duben 2013 [vid. 2017-06-17]. Dostupné z: <http://control.fs.cvut.cz/en/courses/computer-support-study-e372041>
- [5] *Moodle - Open-source learning platform | Moodle.org* [online]. [vid. 2017-06-17]. Dostupné z: <https://moodle.org/>
- [6] FARCIC, Viktor. Black-box vs White-box Testing. *Technology Conversations* [online]. 11. prosinec 2013 [vid. 2017-06-17]. Dostupné z: <https://technologyconversations.com/2013/12/11/black-box-vs-white-box-testing/>
- [7] *About Moodle - MoodleDocs* [online]. [vid. 2017-06-17]. Dostupné z: https://docs.moodle.org/33/en/About_Moodle
- [8] *Features of our online examination system | Onlineexambuilder.com* [online]. [vid. 2017-06-17]. Dostupné z: <https://www.onlineexambuilder.com/features/item10065>
- [9] *EasyTestMaker* [online]. [vid. 2017-06-17]. Dostupné z: http://www.easytestmaker.com/?utm_campaign=elearningindustry.com&utm_source=%2Ffree-testing-tools-for-online-education&utm_medium=link
- [10] VEYRAT, Pierre. Business Process Standardization: All you need to know. *HEFLO EN* [online]. 25. leden 2016 [vid. 2017-06-17]. Dostupné z: <https://www.heflo.com/blog/bpm/business-process-standardization/>
- [11] *Flask - Full Stack Python* [online]. [vid. 2017-06-17]. Dostupné z: <https://www.fullstackpython.com/flask.html>
- [12] *The Web framework for perfectionists with deadlines | Django* [online]. [vid. 2017-06-17]. Dostupné z: <https://www.djangoproject.com/>
- [13] PyCharm. *JetBrains* [online]. [vid. 2017-05-28]. Dostupné z: <https://www.jetbrains.com/pycharm/>
- [14] NOWELL STRITE. Introduction to Python. In: [online]. Technology. B.m. 20:50:44 UTC [vid. 2017-05-29]. Dostupné z: <https://www.slideshare.net/nowells/introduction-to-python-5182313>

- [15] TUTORIALSPPOINT.COM. Python Dictionary. *www.tutorialspoint.com* [online]. [vid. 2017-05-29]. Dostupné z: https://www.tutorialspoint.com/python/python_dictionary.htm
- [16] *Learn Python the Hard Way* [online]. [vid. 2017-05-29]. Dostupné z: <https://learnpythonthehardway.org/book/ex40.html>
- [17] *PyPI - the Python Package Index : Python Package Index* [online]. [vid. 2017-05-29]. Dostupné z: <https://pypi.python.org/pypi>
- [18] *Requests: HTTP for Humans — Requests 2.17.3 documentation* [online]. [vid. 2017-05-30]. Dostupné z: <http://docs.python-requests.org/en/master/>
- [19] *fuzzywuzzy: Fuzzy String Matching in Python* [online]. Python. B.m.: SeatGeek, 2017. Dostupné z: <https://github.com/seatgeek/fuzzywuzzy>
- [20] DEVELOPERS, tqdm. *tqdm: Fast, Extensible Progress Meter* [online]. Python. nedatováno. Dostupné z: <https://github.com/tqdm/tqdm>
- [21] RICHARDSON, Leonard. *beautifulsoup4: Screen-scraping library* [online]. Python. nedatováno. Dostupné z: <http://www.crummy.com/software/BeautifulSoup/bs4/>
- [22] FARCIC, Viktor. Test Driven Development (TDD): Example Walkthrough. *Technology Conversations* [online]. 20. prosinec 2013 [vid. 2017-06-08]. Dostupné z: <https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/>
- [23] *Introduction to Test Driven Development (TDD)* [online]. [vid. 2017-06-17]. Dostupné z: <http://agiledata.org/essays/tdd.html>
- [24] MARCO. Fuzzy String Matching in Python. *Marco Bonzanini* [online]. 25. únor 2015 [vid. 2017-06-08]. Dostupné z: <https://marcobonzanini.com/2015/02/25/fuzzy-string-matching-in-python/>
- [25] *Levenshtein Distance* [online]. [vid. 2017-06-08]. Dostupné z: <https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm>
- [26] *python - What is __init__.py for? - Stack Overflow* [online]. [vid. 2017-06-11]. Dostupné z: <https://stackoverflow.com/questions/448271/what-is-init-py-for>
- [27] *HTTP/1.1: Status Code Definitions* [online]. [vid. 2017-06-17]. Dostupné z: <https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
- [28] *Quickstart — Requests 2.18.1 documentation* [online]. [vid. 2017-06-17]. Dostupné z: <http://docs.python-requests.org/en/master/user/quickstart/#make-a-request>
- [29] *25.3. unittest — Unit testing framework — Python 2.7.13 documentation* [online]. [vid. 2017-06-17]. Dostupné z: <https://docs.python.org/2/library/unittest.html>

- [30] 29.4. `__main__` — *Top-level script environment* — *Python 3.6.1 documentation* [online]. [vid. 2017-06-17]. Dostupné z: https://docs.python.org/3/library/__main__.html