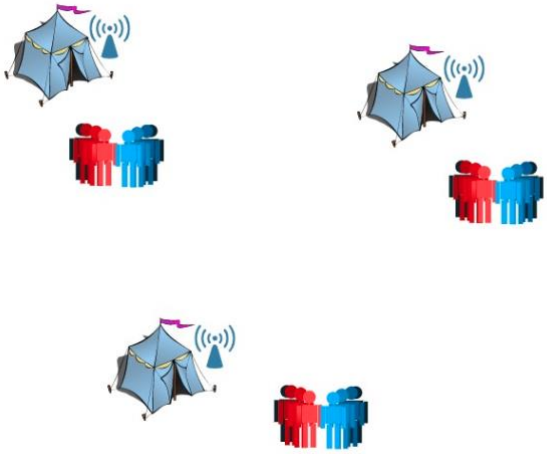
COMP0066 Introductory Programming Coursework
2021/22
Moodle submissions only

The coursework will assess your ability to demonstrate your programming knowledge, purely in python. You will work in groups on a programming project.

e-Adam; A humanitarian emergency management system

During a natural disaster, people flee their homes to more secure camps where they can receive medical help, food and shelter. To distribute resources equally, humanitarian agencies need to record the number of refugees and their needs at every available camp. The simplified architecture of the system is as follows:



e-Adam overall architecture

Task

A humanitarian group has appointed several developers (your group) in order to implement an e-Adam prototype. The system is mainly used to record survivors of natural disasters. A list of required features is described below but it shouldn't prevent you from adding new useful features.

Features

The system accepts two types of users. Both types should be able to login/logout using a username and password pair.

(a) The general application **admin**, who can create new emergency plans, with type of emergency, description, geographical area affected, and start date. The admin can also add closing date, then close emergency plans. At any time of the emergency plan life cycle, the administrator can display summary of all related details; including, number of refugees, their camp identification, and number of humanitarian volunteers involved in the plan at each camp.

(b) The admin can also deactivate/reactivate volunteers accounts or simply delete them completely from the system. Deactivating means, volunteers

can no longer use their accounts. If they try to login, they get a message "Your account has been deactivated, contact e-Adam administrator". But once reactivated they can login and use their account again as usual. If the account is rather deleted, the volunteer gets a message "Account doesn't exist"

(c) The humanitarian **volunteer** can edit their information (name, phone, etc), identification of their camp, their availability.

(d) The volunteer can also create an emergency profile for each refugee and his/her family, identification of the camp, medical condition, etc.
Keep it simple, information about the lead family member and number of relatives present is sufficient.

Persistence

All information entered into the system, such as refugee information, number of members of the family, medical conditions, date of creation of the record, etc, should be persistent across sessions; meaning that if you make a change and close your application, the last changes should remain when you restart your application again.

Implementation

You can use object-oriented programming if you wish. Its preferable but not mandatory.

Notes: The management system should be implemented as a command-line application, but you can have a GUI -based application instead if you wish. It is allowed but not mandatory.

The aim of the exercise is to practice your knowledge in core python. The use of ready-made frameworks such as ***Django is not allowed***, but it is possible to use third-party libraries as long as the source is mentioned.

Deliverables

The grade for your COMP0066 coursework will depend on the quality and correctness of your programming implementation, but also on the peer-evaluation of your group mates. You are required to submit two deliverables:

Deliverable (a) - one submission per group

1. Use moodle link to submit your single .zip file containing all your code source files (.py files).

2. Include a .txt file with the **link** to your UCL MediaCentral (https://mediacentral.ucl.ac.uk/) video demonstrating all the features of your application "not the code". You can add voice-over or text comments. The video length should be between 5 and 8 minutes maximum.

   Note: The video is not marked but used to promote your application to the examiners.

3. Name your zip file *groupNN.zip*, where *NN* is the group number. For example, if your group number is 7, your file should be named group07.zip

Notes: Before submission, make sure the maximum size of your zip file doesn't exceed 50MB.

Email submissions are equivalent to a non-submission. It's the responsibility of the group members to ensure submissions are made on moodle well before the deadline, to avoid last minute technical glitches.

Delieverable (b) **- individual submission**

4. A completed online ipac form that will be available on moodle when groups are finalised. The form takes 10 minutes to complete.

Note: The ipac form is mandatory and should be completed by the deadline on moodle. Not submitting the form will delay receiving your mark.

Marking scheme

You will be assessed clearly on the following, which must be shown in a useful context.

1. Implemented a reasonably complete application.
2. Showed that you can put in practice what we have covered in lectures and practicals.
3. Make sure your code is robust enough by testing it before submission, as you may lose marks if your software application raises errors or behaves strangely.
4. The submitted code should be self-contained. It should work on any machine with core python, without any extra configuration.
5. A detailed coursework marking criteria is attached at the end to the current document. Your final mark is devised by considering your individual contribution to the project (using ipac scores).

Group work

As a first task, you need to select a project leader that will organise the group. All members of the group should contribute to the implementation of the project. Poor engagement with the project will decrease your individual mark.

Useful links

Please note that the following are pointers to tools and libraries that might be useful. It is not mandatory to use them though.

1. **Command-line Applications**. https://docs.python-guide.org/scenarios/cli/

2. **Python Modules and Packages – An Introduction.** by John Sturtz. https://realpython.com/python-modules-packages/

3. **How to Write Beautiful Python Code with PEP 8** https://realpython.com/python-pep8/

Extenuating circumstances & late submissions

Please check out your academic manual available here
https://www.ucl.ac.uk/srs/academic-manual/c4/failure/late-submission

UCL Computer Science: COMP0066 Programming project marking criteria and grade descriptors.

Fail		Pass (2:2)		Merit (2:1)		Distinction (1st)		
Inadequate	Weak	Satisfactory		Good		Excellent	Outstanding	Exceptional
Below 40: Fail	40-49: Fail	50-54: Low pass	55-59: High pass	60-64: Low merit	65-69: High merit	70-79	80-89	90+
Either no solution or solution provided is inappropriate and irrelevant.	Rudimentary coding, significant omissions in list of implemented requirements.	A reasonable attempt at providing a software solution with limited features and code containing bugs limiting the use of the solution.	A sound solution with a reasonable list of implemented required features. Code containing several bugs and requiring improvements.	The solution implements all or the majority of required features, contains few bugs and is subject to some improvements and fixes.	The solution implements all required features, contains no or minor bugs and is subject to minor improvements.	The solution implements major extra features related to application domain in addition to the required ones. Research has been done and the resulting software is of the quality of commercial applications, competing in terms of features and robustness.	The solution is innovative, provides a major addition to the application domain. The solution can be published in a conference or journal.	The solution is exceptional in terms of algorithms, performance, and features. The solution provides a major contribution to the domain of software development.