Technical Task for Software Developer (Python) Position (TU/e – PITC)

- Please implement the project in a git repository, and do so the same way you would do it in a real team project.
- If any information in the tasks is missing (e.g. restrictions, fields, etc.), feel free to define it yourself.
- Send us the link to the git repository with the code: d.pustakhod@tue.nl, e.panina@tue.nl, b.e.hasenson@tue.nl.

Part 1

Intro: Our Digital Platform provides customers access to services of several companies (service providers). On the platform, there are also account managers (users with a specific role), each of them manages her own set of service providers. A service provider may be managed by several account managers.

Workflow: When a new customer comes for JePPIX services, the workflow is organized in the following way:

- An account manager registers the new user account for this customer.
- The customer creates an order. The complete order is visible to the account manager.
- The customer fills the order with services from the service providers. However, there's a limitation: she can add services only from those service providers which are managed by her account manager.
- Later, the same customer may create more orders.
- Later, another account manager may also work with the same customer, and this account manager must not see the previous orders of this customer which were managed by the first account manager.

Note: confidentiality of orders and customer relations is extremely important!

Tasks:

1. Create Django models for the required entities for the digital platform.
2. Fill the models with any information you think may be relevant.
3. Add required relations between the models to ensure DB-level enforcement of the constraints mentioned in the workflow.

Part 2

Intro: On the digital platform, there is functionality to create analysis reports on jobs and orders. The goal is to provide detailed metrics on performance, which can be used for quarterly reviews,

performance tracking, and reporting. A typical analysis is performed for a range of dates (from … to …), and the results are stored in the corresponding models. *The functionality is implemented in the attached code, see comments in the code for details of implementation.*

Tasks: We want to enhance the analysis capabilities. The following changes are to be implemented:

- Enable statistics calculation on creating or saving *Report* model.
- Broaden the scope of the statistics related to *Jobs*. In addition to the total number of jobs in the specified time range (implemented), we need to calculate and store the following:
    - average job completion time per job type;
    - number of jobs per status.
- Implement analysis script(s) for statistics related to Orders. Put the scripts in the `stat_utils.py`.
- Add model and analysis for User statistics.
- Add django admin pages displaying the statistical report data. Focus on the convenience of the data representation, filtering, etc.
- Add a possibility to link a PDF file to reports.
- Add unit tests to check that the statistical calculation operations are correct.

Notes:

- Use links to the models created in Part 1 of this task as necessary.
- Update existing model fields and relations if necessary.