Course: **NIT-CE-08 Python programming**

Final project

Using the Python programming language, implement the user register according to the given specification below.

- The input data for the program is in the JSON files with user data.
    - The fields in the JSON file are:
        - Name and surname, type: string,
        - e-mail address, type: string,
        - IPv4 address of users with decimal parts, type: string,
        - A list of mapped devices, tip svakog elementa liste: string.
    - Make at least 4 JSON files with the different user data, each containing at least 8 users. At least one file must have a user that is already in another file (a duplicate).
- The program should support loading the JSON input file.
- The program should combine the data from the JSON input files in a single user register that should be printed as an output of the program.
- Users and data can be found in multiple JSON files, so it is important to detect and delete duplicate data. If the user has different devices listed in different files, information should be merged, i.e. the union of devices should be saved in the register.
- The key in the register is the user's e-mail address.
- A class representing the user register should have at least the following functionality:
    - A constructor that accepts the list of input files as parameters. The constructor should create a user register and fill it with data.
        - During the parse of the user data, it is necessary to check if the e-mail address and IPv4 address of the users are in the correct format. If the format is not correct, such data should be discarded (with the appropriate message).
    - Overloaded methods for data collections: allow the usage of the len operator to find out the size of the register and indexing with the key (e-mail address), for reading and for writing.
    - Getter methods for each piece of user data, except the e-mail address. For example, get the IPv4 address of the user with the key abc@def.com.
    - Setter methods for each piece of user data, except the e-mail address. If IP address is changed, check the format.
    - Overloaded operators for: addition (merging, i.e., union of registers) and multiplication (intersection of two registers, i.e., a new register containing only the users that exist in both registers).
- All implementation parts must be *unit* tested.

You should implement your solution as team and create the presentation of your solution (PPT) that presents what you have implemented. During the presentation session, each team will have 20 minutes for the presentation during which each of you should contribute equally and present your part of the solution. The presentation should have main information about the solution (PPT) and the demo of the execution of the implemented program.

Submission requirements:

- Source files of all modules in the project solution.
- PPT presentation describing the main concepts of the solution.
- Zip everything in a zip file: Project_groupN.zip (N is the number of your group: 1, 2, 3, 4).
- One submission in the team is enough, as it will be visible to all group members.

Project consultations: Monday, 20 Oct 2025 at 8:15 CEST.

Submission deadline: Wednesday, 22 Oct 2025 at 7:00 CEST via Canvas.

Project presentations: Wednesday, 22 Oct 2025 at 8:00 CEST.

Grading:

The project is worth 40 points.

- The implementation and testing (30 points).
    - Handling input files (5 points).
    - Working with data collections (5 points).
    - Format checking (5 points).
    - Classes and its methods, other than operator overloading (5 points).
    - Operator overloading (5 points).
    - Testing (5 points).
- Presentation and answers during the presentation session (10 points).