Assignment 4

Now that the client trusts your team with the direction, it's time to show off our coding skills in a larger project.

- Implement your design as a WPF app if your subgroup chose desktop, or MAUI app if your subgroup chose mobile. Yes, the entire app.
- In visual studio installer deselect Universal Windows Platform development (WPF is cooler and is better supported).
  - If you chose MAUI then in visual studio installer select .NET Multi-platform App UI development.
- When prompted, use .net 8 for the project, and merge (not rebase) for git. (I very strongly recommend creating separate branches for each task, when that task is done merge the main branch on the task branch, make sure everything is working then create a pull request to merge back into the main branch.)
- Create a GitHub repository with the name: UBB-SE-2024-[your team name]. Invite the client (imre.zsigmond@ubbcluj.ro) as a collaborator so they can take a look.
- Update your class diagram to encompass the entire application, not just your entities. Code and class diagram must be 1-to-1, if you deviated from the initial plan then update either the code or the diagram. Do not generate the class diagram. There are no requirements regarding the structure of your application aside from this.
- Your application should store its data long term, this may be either xml file(s) or SQL server. If you choose xml, you may use any .net framework element to work with it. If you choose SQL then the queries must be written by hand, no ORMs are accepted. If you haven't already, select Data storage and processing in visual studio installer.
- If you need the structure of entities developed by some other team, you can look at their documentation at the git repository for assignment 3. If you need some information that they would provide (e.g. user data), then just copy the info from there since they have committed to implementing it that way. You can create mock-up functions that return for example a single hard coded entity of that type (e.g. GetCurrentUser()).
- Unit or integration tests are not required.
- For now, there is no need for a server. Interactions should go through the data storage if you need one. Let's say you need to send a file to another user; you would save that file locally and store the location. If you logged in as that other user, you would display that they got a file from someone, and you would fetch it from the local disk. Anything direct like a voice call you can do either by sending files this way or connecting through static ip-s on the local network.

Deadline: lab 4