6. PRACTICAL EXAMINATION

Below you will find a problem statement similar to what you can expect to receive during the practical examination. The problem statement will in general follow the requirements set out between A45 and A10, will require a graphical user interface (using Qt) and writing specifications, tests and the implementation of layered architecture.

Observations:

1. Solving the following problem statement completely should be possible for you in a time span of 3 hours.
2. You are free to use your preferred IDE. Make sure your IDE is set up correctly and it works! Make sure that Qt works!
3. You are allowed to use Qt Designer, if you want to.

6.1. Problem statement

Write an application which simulates the development and testing of a software application, as follows:

1. The information about the development team is in a text file. Each member of the team **- *User*** has **a name** (string) and **a type** (string), which indicates whether the user is *a tester* or *a programmer*. This file is manually created and it is read when the application starts.
2. Another file contains information about the issues reported by the testers. Each ***Issue*** has **a description** (string), **a status** (can be *open* or *closed)*, **the reporter** – the name of the person who reported it and **the solver** – the name of the person who solved it. These are read when the application starts and are also stored in the file by the program.
3. When the application is launched, a new window is created for each user, having as title the user's name and type (tester or programmer). **(0.5p)**
4. Each window will show all the issues, with their description, status, reporter and solver, sorted by status and by description. **(1p)**
5. Only testers can report issues, by inputting the issue's description and pressing a button "Add". The issue's reporter will automatically be set – this will be the name of the tester who added it. This operation fails if the description is empty or if there is another issue with the same description. The user will be informed if the operation fails. **(1.25p)**
6. Both programmers and users can remove issues. An issue can only be removed if its status is *closed*. **(1p)**
7. Only programmers can resolve issues, by selecting the issue and pressing a button "Resolve". This button is activated only if the status of selected issue is *open*. When an issue is resolved, the name of the issue's solver is automatically updated to the name of the programmer who solved it. **(1.25p)**
8. When a modification is made by any user, all the other users will see the modified list of issues. **(2p)**
9. When the application is finished, the issues file will be updated. **(0.5p)**

Observations

1. **1p** default
2. Specify and test the following functions (repository / controller):
    a. Function which adds an issue. **(0.5p)**
    b. Function which updates an issue's status and programmer. **(0.5p)**

3. Use a layered architecture. If you do not use a layered architecture, you will receive 50% of each functionality.
4. If you do not read the data from file, you will receive 50% of functionalities 3, 4, 5 and 6.

6.2. Advice for the practical examination

1. Implement a problem similar to the one in the example. Time yourself while solving it, make sure you can implement at least some of the functionalities in the allotted time (3 hours). This way you can detect where your difficulties are and you can improve yourself.
2. Build your application incrementally: one step at a time, **compile frequently**.
3. Only add one function in one step, so that you can easily revert to a functional version, in case something doesn't work.
4. **Do not ignore errors**, solve them before continuing with writing source code. **Read the error text!**
5. Do not ignore warnings, sometimes these can indicate errors in the program.
6. **Do not implement functionalities that are not required**. By doing this, you might waste valuable time and **the source code will not be graded**!
7. If there are issues that you cannot solve, try finding alternative implementations such that you can still test your code.
8. For the practical examination, build a Qt empty project and make sure that it compiles and that you can execute it.