Interview Case v1.9

Task: Implement an Issue Tracking Engine

Ilias Merentitis – 12/10/2022

App Description

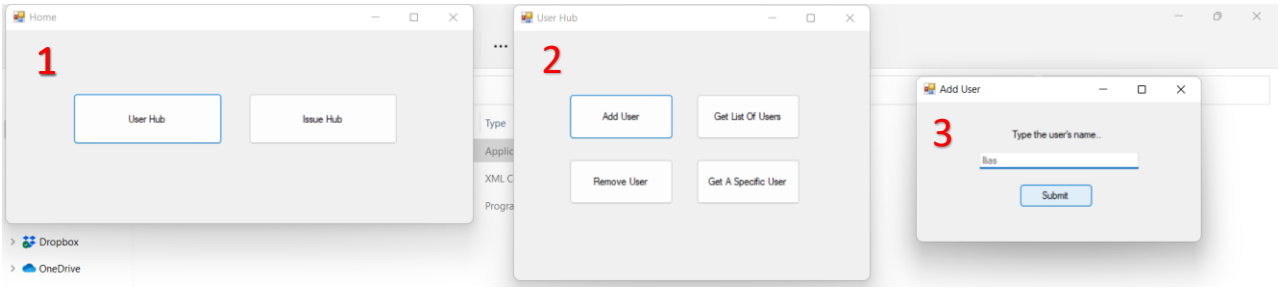
For the purposes of developing a solution on the topic of issue tracking, I decided to work with C# and specifically the .Net framework. The reason for this is the simplicity of designing very simple interfaces. Although in the description of this case it was very clear that an interface of any kind is not mandatory, as it is well out of scope for the interview, I found that using Windows Forms actually makes it easier and simpler to develop such an application than using simple test functions and running them on console. The advantage of having a full visual representation of the architecture while not sacrificing development time made it much easier for me to design, develop and test the application. Therefore, all use cases of the solution can be directly tested by opening the appropriate form, filling the necessary information, if any, and then visually observing the results.

An initial windows form class under the name “Home” is where the application begins. It is there that all global variables and objects are kept, for example the list of users and the list of issues. Also, all functions required, like adding a user, adding an issue, assigning a parent issue to another issue etc, are written here and called from other windows forms classes. These latter classes get a reference to Home granting them access to the data and functions of Home. For this, an intermediate Windows Form class called either “User Hub” or “Issue Hub”, for users and issues respectively, is used as a connector.

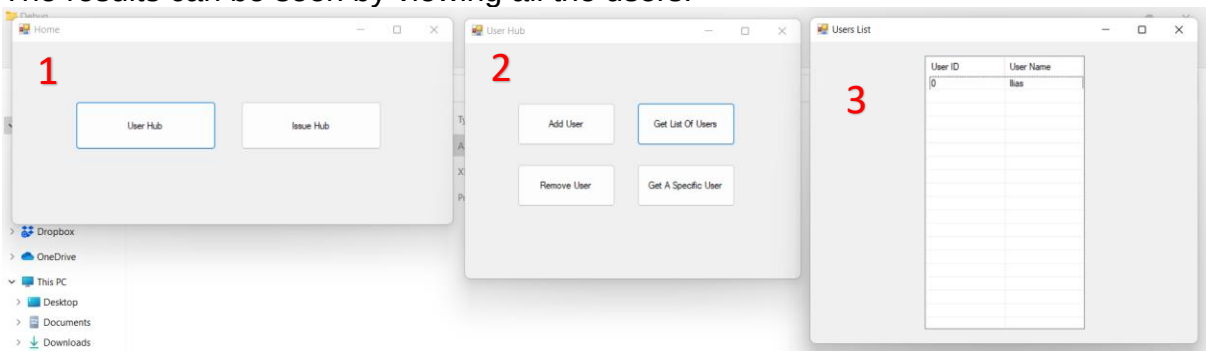
As for the data, a user is an object with a <int ID, string Name> structure. The ID is a unique identifier with no specific meaning, unlike the Swedish personal number, and it is produced by incrementing a global integer. An issue has a <int issueID, DateTime timestamp, string state, string type, int parentID, string title, int userID> structure. As with the userID, the isssueID is also produced by an incrementing integer. Furthermore, all variables that should never have a null value get a -1 instead. For example, an issue that has just been created has no parent issue or user assigner. That means that for said issue, userID = -1 and parentIssueID = -1 too.

App Description

Since the drag and drop functionality of Visual Studio made it so easy to use Windows Forms, each function, like addUser, can be tested by clicking to open the appropriate form. For example, the addUser function can be tested with various inputs by just writing in the field in step 3. Inputs of non-integers as well as negative numbers are not allowed, and the application does not proceed.



The results can be seen by viewing all the users.



The same holds true for all functions that require an input, including removeUser, removeIssue etc.

Others have an extra layer of input restriction. For example, in the form that is used to match a user to an issue, I have used dropdown lists populated by the available data in the users' and issues' list only. Therefore, the user cannot input a wrong value as the userID or the issueID. This sort of safety net has been implemented in other functions as well, making testing and debugging easier, as well as reducing errors.