Milestone 2

Design and OOP

General

In this milestone, you are requested to implement the logic behind the Kanban board and to make sure the data is persistent for a case of failure.

The current version of the kanban board will have no UI (user interface), yet it will have an interface layer, that you (and we) will use for testing your software. Thus, your kanban software will have three layers constructed by you: an interface layer, a business layer, and a persistent layer.

Goals

The goals of this milestone are:

- Practice the N-tier architecture.
- Practice team work & version control.
- Practice C#.
- Experience right use of OOP with C#.
- Understand and write a Low Level Design (LLD) by creating a class diagram.

Business and Integrity Rules

1. A user is identified by an email, and is authenticated by a password.
2. A task has the following attributes:
    a. The creation time.
    b. A due date.
    c. A title (max. 50 characters, not empty).
    d. A description (max. 300 characters, optional).
3. A board has three columns: ‘backlog’, ‘in progress’ and ‘done’.

Functional Requirements

Users

1. A user password must be in length of 4 to 20 characters and must include at least one capital character, one small character and a number.
2. Each email address must be unique in the system.
3. The program will allow registering of new users.
4. The program will allow login and logout of existing users.
5. The login will succeed if, and only if, the email and the password matches the user's email and password.

Boards

6. Each column should support limiting the maximum number of its tasks.
7. By default, there will be no limit on the number of the tasks.
8. The board will support adding new tasks to its backlog column only.
9. Tasks can be moved from 'backlog' to 'in progress' or from 'in progress' to 'done' columns. No other movements are allowed.

Tasks

1. A task that is not done can be changed by the user.
2. All of the task data can be changed, except for the creation time.

Non-functional Requirements

1. Persistency:
    a. In this assignment, persistent data is stored in local files by serialization.
    b. The following data should be persistent:
        i. The users
        ii. The tasks
        iii. The columns
    c. Persistent data should be restored once the program starts.
2. Logging:
   You must maintain a log which will track all errors in the system. Note that "error" does not necessarily mean an exception (see next item) that is "thrown" or "raised" by your runtime environment, but any situation which counts as invalid in our domain. Some guidelines:
    a. Tag entries with their severity / priority
    b. Provide enough information to understand what went wrong, and where
    c. Avoid storing entire stack traces directly in the log (they are more verbose than useful)

    d. Use a shelf Logger. Some examples: log4net, Observer logger, Composition Logger, end there are many more.
3. Exception Handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. Your program is expected to operate even when error occurs:
    a. Handle any malformed input.
    b. Handle logic errors (e.g., login for non-existing user, etc.).

General Guidelines

- Before creating a repo, make sure your group is registered. Don't change your group id unless it's really needed.
    - Version Control: You are expected to work as a team and use source control (GitHub specifically).
    - We will use GitHub classroom. One team member should create a team and the others should join the team using this link (don't use it until your team is registered).
    - A repository will be created for the team automatically. You should use only this repo during the development of this milestone. The team's name should be: "team" + your team number. For example, for team number 1: "team1".
- Document your code thoroughly.
- Pay attention to "Magic numbers".

Design instructions

- Before starting your design (and of course writing your code), consider the following:
    - Are there any operations that a column can/should do?
    - Are there any operations that a task can/should do?
    - Are there any operations that a user can/should do?
    - Where do each method belongs? (login & logout/create & move a task, etc.)
    - Can you think of new requirements that the client (the course staff) will **probably** request? Can you make the design and the code versatile enough to support such requests (of course without too much work on your side because we might not ask for it)?
- N-Tier structure: pay attention to right use of the N-tier model.
- OOP principles: remember OOP principles and use them (Encapsulation, Abstraction, Interface, and Inheritance).
- The aforementioned definitions for a user, a board and a task, contain the required information for the functional requirements. You are allowed to add or change the data fields as long as the requirements are met.

- Having a persistent layer (and backuping data in files) does not mean that the data objects (i.e., users, tasks, and columns) are stored in files only. In fact, these data object will probably have some logic and methods (e.g., the user object). Moreover, your data should be stored in the RAM for fast retrieval. The persistent layer should be called only upon system startup (for restoring the persistent data) and upon data update (e.g., registration of a new user).

Submission Dates, Deliverables and Grading

Deliverables

1. One solution named *KanbanSolution*, containing one project only named *KanbanProject*.
2. The code for each Tier (Interface layer\Data access layer\Business logic) will be placed in a separate directory. In Total you should have three folders with the names of the layers.
3. To be clear: you do not develop a presentation layer.
4. You should test your project using the interface layer. You should include a main file that will execute these tests. Name it *Program.cs*. Place it in the root of the project.
5. A class diagram (which acts as your LLD).
6. A **new HLD** relevant for the project.
7. The HLD and LLD will be placed in a folder named 'design' in the root of the solution. The HLD will be named 'HLD.pdf' and the LLD will be named 'LLD.pdf'.

General Submission Notes

1. You need to work with git. All team members must commit. Your submission should be in the master branch.
2. Tag the commit with the submission with the proper tag as described in the project information.
3. Additionally, one team member should zip all of the submission files and submit them using Moodle.
4. Every hour late in the submission will cause a reduction of 5 points from your grade, up to 30 pointes.
5. Your design documents (HLD & LLD) should match and reflect your actual program design. Please update them and re-push them to the repository before the milestone deadline.

Milestone Deadline - 04.04 23:59

Milestone Presentation

The program functionality will be examined during one of the lab sessions, while the rest (e.g., design documents, documentation, etc.) will be examined offline.

All team members must arrive at least 10 minutes before your time slot (a schedule will be published) and present your work. The following documents should be ready and **opened in different tabs** on a computer at your time slot:

a. Your GIT repo webpage.
b. Your project on visual studio.
c. Your design documents.


Good Luck