Milestone 4

DB

General

After the success of the second and third milestones, you are now requested to use a database and further expand the functionality of your kanban application. Your clients still request the graphical user interface (GUI) of the Kanban board. You and your team are tasked to upgrade your current kanban app and generate a new version that supports these new demands.

The current version of the kanban board will have all three tiers. Thus, your kanban software will have: a data tier, a logic layer, and a presentation tier.

Goals

The goals of this milestone are:

- Practice the use of databases.
- Practice the N-tier architecture.
- Practice team work & version control.
- Experience right use of OOP with C#.
- Practice designing a GUI and implementing it in WPF.
- Practice C#.

Changes from Milestone 2 are marked in yellow and from milestone 3 in blue.

The GUI will support all of the requirements below and will have at least the following windows:

1. Login / registration window.
2. The user window, for managing boards (add and remove boards) and for selecting a board to present.
3. The board window. No changes here, yet, notice that the GridView of this window will containing the data from the selected board ONLY.
4. The Task window containing all the task info.

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 1 or more ordered columns, each identified by a name (e.g., backlog).

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

1. By default, a board has three columns: 'backlog', 'in progress' and 'done'.
2. Columns can be added to the board and be removed.
3. Columns presentation order can be changed.
4. Each column should support limiting the maximum number of its tasks.
5. By default, there will be no limit on the number of the tasks.
6. The board will support adding new tasks to its ~~backlog~~ leftmost column only.
7. Tasks can be moved from ~~'backlog' to 'in progress' or from 'in progress' to 'done' columns.~~ one column to the next column in the presentation order. No other movements are allowed.
8. A user can add and remove boards.

Tasks

1. A task ~~that is not done~~ can be changed by the user, unless it arrived to the final column in the representation order.
2. All of the task data can be changed, except for the creation time.

3. Tasks can be sorted by date.
4. Tasks can be filtered by its text fields (i.e. the title and/or the description). For example: filtering the tasks by the search term “login” will return tasks that their title/description include the word login, as:
    a. Implement login screen.
    b. To add captcha to the login screen.

Non-functional Requirements

1. All the non-functional requirements from Milestone 2 and 3.
2. The DAL (persistence layer) should use a DBMS (SQLite) and **not files**.
3. Unit testing:
    a. implement unit testing for two functions, the functions should be fully covered by the test. At least three tests per function.
    b. Use Microsoft unit test framework for managed code.
4. Usability. The GUI must comply with the following requirements:
    - Buttons in the interface should contain only short text descriptions (no more than two words) or a descriptive icon (preferably both).
    - Text fields and text boxes should be large enough to comfortably accommodate user input, but not so large that they are unwieldy.
    - Data input by users should be validated, and input of invalid characters should be blocked (e.g. empty messages, nicknames, etc.).
    - For additional guidelines, read this page on Good User Interface Design Tips.
    - For the sake of simplicity you can use DataGrid to design your Kanban (like we practiced in the practice session).

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. (Continue using your existing repo)
    - 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.
- Use bindings when writing the GUI. **You are not allowed to read properties directly** (for example a function should not use the text of a textbox directly).

- Pay attention to "Magic numbers".
- 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 (Presentation layer\Data access layer\Business logic layer) will be placed in a separate directory. In total you should have three folders with the names of the layers (four if the interface layer is separated from the business layer).
3. An **updated** class diagram (which acts as your LLD).
4. A **updated HLD** relevant for the project.
5. 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 - 13.06 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 **<u>opened in different tabs</u>** on a computer at your time slot:

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