***Design Changes***

*And Justifications*

*This document presents the changes made to group 34’s design, with elaboration and a small reasoning to each of our changes.*

*List of Changes and their explanations:*

1. *Added Log Attributes to each Class:*

Our group has added log attributes as fields in each of our classes in both the service layer and business layer, in order to utilize the given .NET log4net in each of our classes. We used log4net in order to log important events such as creating an object successfully, as well as tracking method calling for debugging and when errors are received.

1. *Adding “BoardManagement” Class:*

Differently to our initial design, we decided to add a BoardManagement in order to store a list of boards, managing them such as adding boards, removing boards, and other methods that are deemed relevant to fulfill the clients functional requirements. A board manager makes it easier by having everything centralized in one place regarding boards and their tasks.

1. *Adding string email to Board:*

We added the email of the user who created the board to the board object itself in order to give our project an easy way to link between the users and the boards they create.

1. *Adding “Response”, “ResponseInt”, and “JsonController” classes to our ServiceLayer:*

Our group decided to add these three classes in order to effectively handle and return the proper jsons for their appropriate cases. Response and ResponseInt handle the responses we are supposed to return and “JsonController” manages all json functions.

1. *“Addboard”,“Removeboard” and “listprogresstasks” in Boardmanagement:*

Our group decided to move these three functions from UserManagement to BoardManagement since it is only reasonable and logical for the Board Manager to handle board functionalities, such as adding boards, removing boards, and listing their respective progresstasks.

1. *Added “isloggedin” and “passwordmatch” in user:*

Used for many other functionalities throughout our project.

* “IsLoggedIn” checks whether a user is logged in, used in many functions that require the user to be logged in first of all in order to perform an action.
* “PasswordMatch” verifies that the password of the user who is trying to login matches the password that was created. Ensures security.

1. *editTask motivation (using “string?”, dateTime+ dateTime, numAct):*

Our group performed the functional requirement editTask with the use of many different ideas and methods:

* we utilized the function “#nullable enabled” with “string?” in our inputs for values in order to allow the insertion of null values, which allows us to check which value we want to change. Our motivation is that we check Title, Description, and DateTime whether they are null or not. The values that aren’t null, are then validated if they are legal values and then edit them accordingly.
* We utilized the dateTime object of visual studio C# in order to register the duedate, and creation time of tasks, with year, month, day, and hour: minutes. Making it easier to track time using an already built system. One main functionality of DateTime is. Now, which we used in order to assign creationTime of a task to the exact time it was created.
* Finally, we included an int “numAct” in our inputs to editTask. This integer is responsible for representing the edit we want to do, to elaborate:
  + 0 -> editTitle
  + 1 -> editDescription
  + 2 -> editDueDate

This document elaborated on some of the many functions of our design changes.