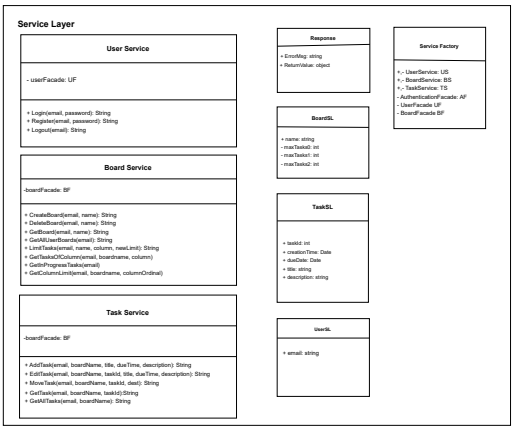

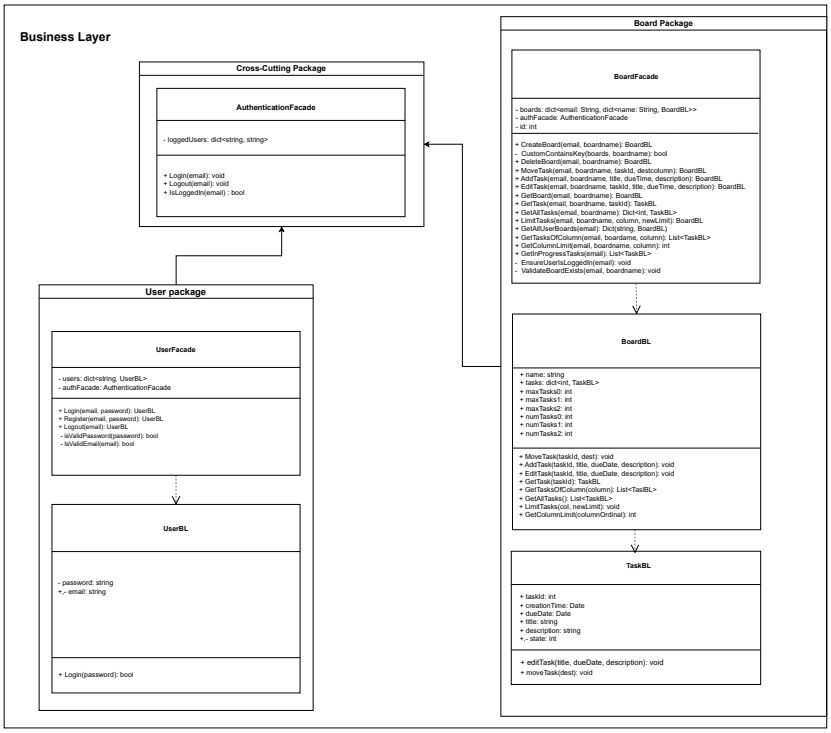

Changes we made from the initial diagram

1. BoardBL:

- Added fields: `maxTasks0`, `maxTasks1`, `maxTasks2`, `numTasks0`, `numTasks1`, `numTasks2`.

Justification: To enforce column-specific task limits and track task counts, the system needed to support per- column constraints for correct task flow control.

- Added methods: GetTasksOfColumn(column), GetColumnLimit(columnOrdinal).

Justification: These methods were required in order to support the functions in grading service.

2. BoardFacade:

- All methods now take `email` and `boardName` as parameters.

Justification: Since each user can have multiple boards, identifying both the user and the board became essential.

- Added public methods: GetInProgressTasks(email), GetAllUserBoards(email), GetColumnLimit(email, boardname, columnOrdinal).

Justification: These methods were required in order to support the functions in grading service.

- Added private methods: EnsureUserIsLoggedIn(email), ValidateBoardExists(email, boardname).

Justification: These private methods reduce duplications in the code.

3. UserBL:

- We removed `userName` and kept only `email` as a unique primary field.

Justification: We made the email the unique identifier for each user in the system, instead of userName.

4. UserFacade:

- Replaced all uses of `userName` with `email`.

Justification: We made the email the unique identifier for each user in the system, instead of userName.

Added private methods: IsValidEmail(email), IsValidPassword(password).

Justification: These private methods reduce duplications in the code.

5. Service Layer (UserService, BoardService, TaskService):

- Clearly separated into independent classes.

- All methods return strings.

- Parameters consistently include `email` and `boardName`.

Justification: This structure improves modularity and testability, returning strings simplifies interaction with the external grading service, which expects serialized outputs. Consistent use of `email` and `boardName` ensures precise identification of resources and prevents ambiguity when handling multiple users and boards.

6. ServiceFactory:

- Stores references to facades: UserFacade (UF), BoardFacade (BF), AuthenticationFacade (AF)

Justification: Keeping all facades in one place makes it easier to manage and use them across the service layer. It avoids duplicate code and helps keep the system organized.

7. Added TaskSL, BoardSL, UserSL.

Justification: These classes exposing only necessary fields from the business layer while hiding internal logic. This separation is required for clean serialization and to support the service layer interface expected by the grading service.