Milestone3: Design Pattern

1. Our data layer implementation is based on the **Singelton** pattern.
   The data layer logic is as follows:
   1. DTO: Represent single record at the DB. Each business layer object hold reference to it compatible DTO which keep it state similar to the business layer object.
      So, each mutation in business layer object will reflect immediately to the compatible record at the DB.
      In this manner we achieved data integrity.
   2. DAO: Implemented as a singleton. Shared between all DTO of specific table. For example, all DTask hold a reference to the same DAO: DTasks.
      These objects contain methods for retrieving, storing, and updating records of a single table.
      DAO will hold a reference to its derived component (if exists), so it can update or retrieve data regarding its components.
      For example, DBoards will hold a reference to DColumns, so when loading the data from the DB, each BL board will receive its relevant columns.
      **Note: BackendController instance at the Frontend package is used as a singleton (all PL model object shares a reference to it, so they can communicate with lower layers), but it does not implemented as one.
      Instead, it is transferred to each PL model object constructor.

2. Our Password object is implemented based on **Decorator** pattern.
   We engage with this pattern to easily add additional restrictions in the future.
   For example, if our customer will ask to forbid more passwords, in addition to the top 20 NCSC, we will simply create a new IPasswordChecker which will enforce this restriction(in its implementation for the Check(string) method) and wrap our old Password object.