### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

* **Youtube Posts Feature** - A feature that allows the user picking a specific facebook friend and search all his posts containing youtube clips, play it, add clips to favourite list.
* **Laetest Pages Posts** – A feature that allows the user getting a quick view on the latest liked-pages posts of all of his friends. Only the latest posts are picked and sorted right after.

### שכלולים של הפיצ'רים והרחבת המערכת:

* **Youtube Posts Feature** - Adding the option to download the chosen youtube clip, adding the option to drag and drop it into the desktop. Adding the options to Serialize to Json format as well.
* **Laetest Pages Posts** – By using a remote server (rented from DigitalOcean and configured in linux ubuntu server) the users are now able to complain on a specific page post – so that when the number of users that complain on this post reach 10, an automatic mail is sended from the application email to the page owner mail. (information about user sessions, page posts and so on is saved on the database).
* **Application Configuration** – when there is not config.json file the config.json file is automatically created from default parameters. However, if there exsists a config.json file, the application configurations is fetched from this file (for example, number of maximum pages posts downloaded from facebook service). Implementation: ApplicationConfiguraion.cs,JsonConfiguraionApplication.cs
* **Basic Logger** – writes a log file mentioning db connections.

### תבנית מס' 1 – Strategy

* סיבת הבחירה / שימוש בתבנית:

We implemented the strategy pattern in the part of the application where the user can "save as" his favourite list of youtube clips. As a result, choosing a type: Json, Xml, is now implemented with the strategy pattern. This helps easily adding new serializer and lessen the code bugs.

**Advantages:**

**Extensibility:** Easily add new strategies – aka add new file formats to the Save As operation.Without the strategy pattern If you ever have to go in and add or change a serializer, you're modifying this code. If you expect to have more conditions, just adding a new strategy is simpler/cleaner/complete. The application can switch serialization strategies at run-time, enabling clients to choose the required algorithm. So to conclude: Serializing Strategy allows you to extend a logic in some parts of your code ("open for extension") without rewriting those parts ("closed for modification").

**Encapsulation:** Data structures used for implementing the algorithm are completely encapsulated in Strategy classes. Therefore, the implementation of an algorithm can be changed without affecting the Context class.The class that uses the strategy, the client class, is decoupled from the algorithm implementation. You can change the algorithms implementation, or add new algorithm without having to modify the client. This can also be done dynamically: the client can choose the algorithm he will use.

* אופן המימוש:

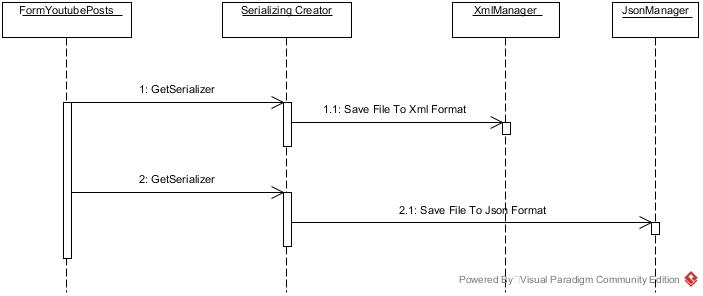
Relevant Classes:

**ISerializingStrategy**

**SerializingStrategy**

**XmlStrategy, JsonStrategy**

Sequence Diagram



### Class Diagram

### C:\Users\nadav\Desktop\Strategy Class Diagram.jpg

### תבנית מס' 2 – Template Method

* סיבת הבחירה / שימוש בתבנית:

**Description:** We implemented the template method because we wanted to dictate the steps required to commit any kind of database manipulation. The steps must be to first **open a connection**, then do the **db query** and then **close the connection**. The manipulation itself is for the inheriters to implement and thus abstract in the abstract class.

**Advantages:**

**Encapsulation & Safety:** Encapsulating the database manipulation happens when subclassesextend the base algorithm without altering its structure**.** In this way we force individual steps of querying the database of the application, providing a template database manipulation which allows defining only the db querying itself.

Readability & Code Reuse: Code reuse happens with our template method because it uses inheritance and only one method need to be overriddren.The Template Method here allows you to define the “skeleton” of an database managing responsibility while leaving concrete implementation up to inheriting classes.(database management classes). The template (in this case, the parent class) serves as an outline of the steps needed to complete the task.

אופן המימוש:

Relevant Classes:

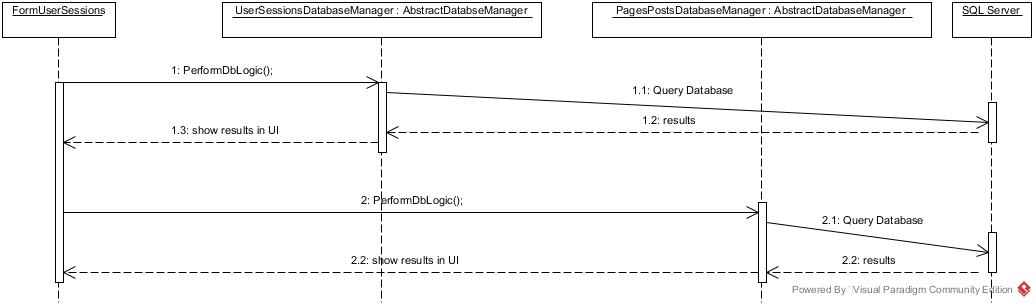
**FormUserSession (Client)**

**AbstractDatabaseManager**

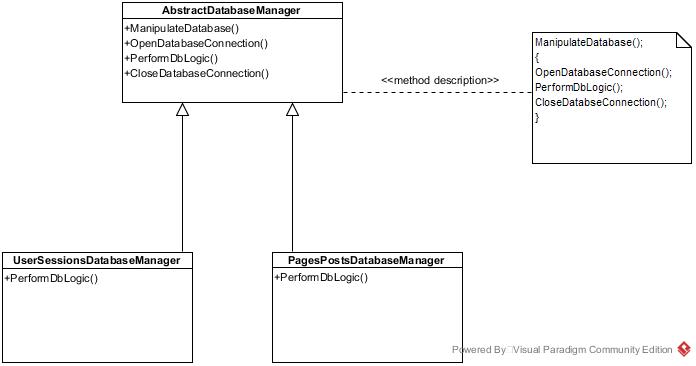
**UserSessionsDatabaseManager**

**PagesPostsDatabaseManager**

Sequence Diagram

****

Class Diagram



### תבנית מס' 3 – Decorator

* סיבת הבחירה / שימוש בתבנית:

Description: We've chosen to implement A Decorator for the database validator of our application which is responsible to validate that the sql server connection is valid. The decorator of the db validator decorates it with Logging abilities which makes sure any validation is followed by updating the log file of the application.

**Advantages:**

Extensibility and Reusability:

Out decorator provides a way to add a logger ability to classes instead of the alternative which is to use subclassing. Moreover, It allows behavior modification at runtime rather than going back into existing code and making changes, moreover,the developer can wrap a database validator with with any number of decorators, allowing adding more decorators in the future.

Less Object Creation: The benefits are that you can change behaviour of database validators at runtime and you are not tied down to having many many different objects a more flexible way to add responsibilities to a class than by using inheritance, since it can add these responsibilities to selected instances of the class..

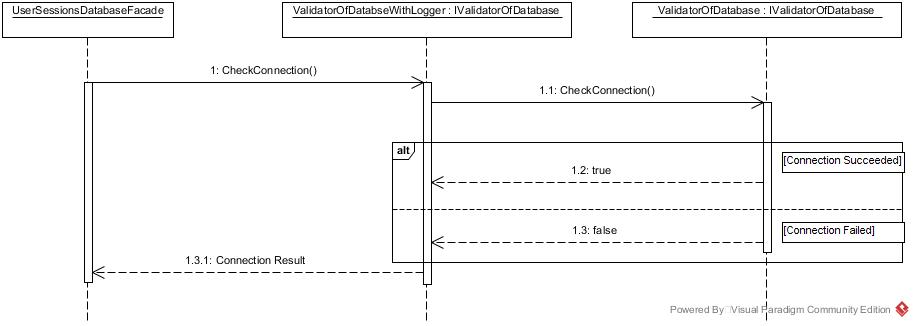
אופן המימוש:

Relevant Classes:

**UserSessionsDatabaseFacade**

**FormUserSessions**

Sequence Diagram



Class diagram

