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

# Top 4 Popular Photos

This feature shows the user his “Top popular” photos.

We scan all the user photos, in all of the users’ albums, and give each photo a score. The score is calculated 60% from the likes on that photo, and 40% others from the number of comments on that photo.

After the scan is complete, we show the user his top 4 photos.

Later, the user can select a photo, from the photos shown under the feature panel, and like it and/or comment the photo.

This ranking feature does not exist on Facebook, getting the “Top liked” or “Top commented” photos is done manually.

# News Feed Text Filtering

The feature gives you the ability to filter posts in your news feed.

The user has the option to create different filter groups.

For each group the user can add text items which will be removed from the news feed.

Furthermore, the user has the option to remove unnecessary item and groups and can decide whether to enable or disable the feature on the news feed.

For Example, if the user doesn’t want to see any series spoilers, the user will create a “spoilers” group and will add item such as: “Game of Thrones”, “Marvel”, etc.

### תבנית מס' 1 – Iterator

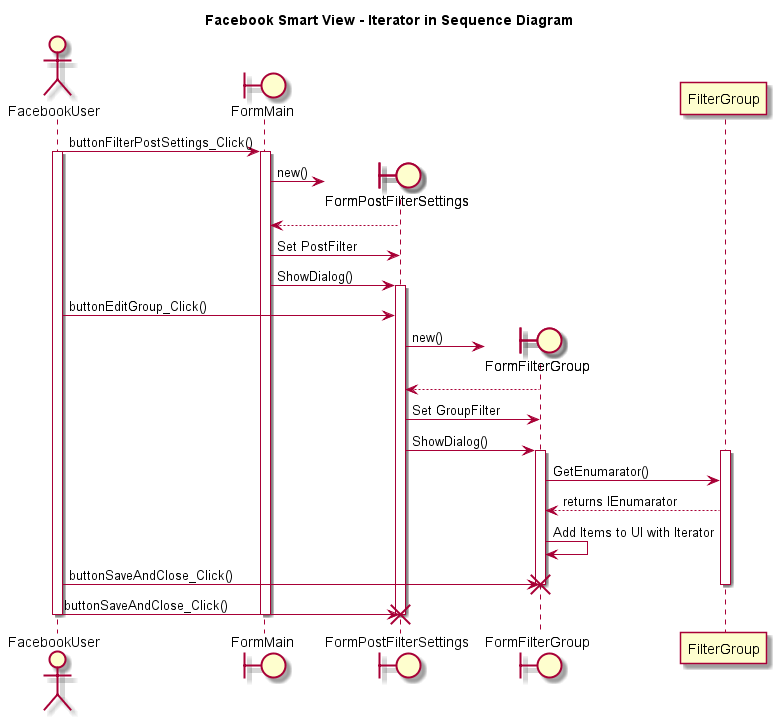
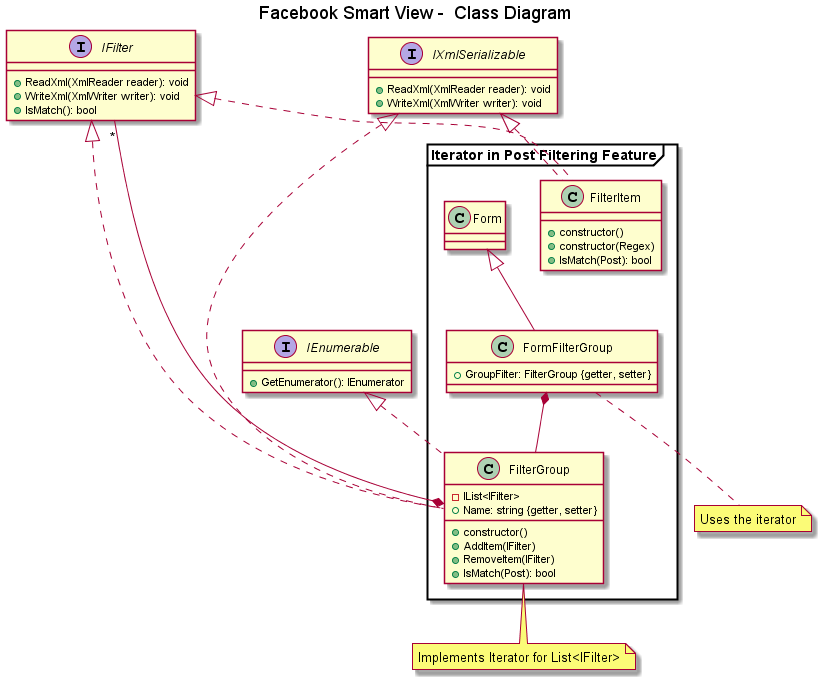
* סיבת הבחירה / שימוש בתבנית:

בfeature השני קיים לנו FormFilterGroup המציג את הItems שאנחנו מעוניינים לפלטר. הItems האלו נמצאים תחת FilterGroup Class במבנה נתונים של List.

מימשנו איטרטור בFilterGroup על מנת שהForm יוכל לעבור על כלל הItems בFilterGroup ולהוסיף אותם לCheckListBox ללא ידיעה באיזה מבנה נתונים נשמרים הItems בFilterGroup.

* אופן המימוש:

בFilterGroup Class קיים מבנה נתונים List שבו שמורים IFilter Class.

FilterGroup מממש את IEnumarable ולכן קיימת מתודה GetEnumarator שעוברת על List<IFilter> בעזרת foreach ומחזירה yield של הitem.(ראה FilterGroup.cs שורה 45) 

### תבנית מס' 2 – Strategy

* סיבת הבחירה / שימוש בתבנית:

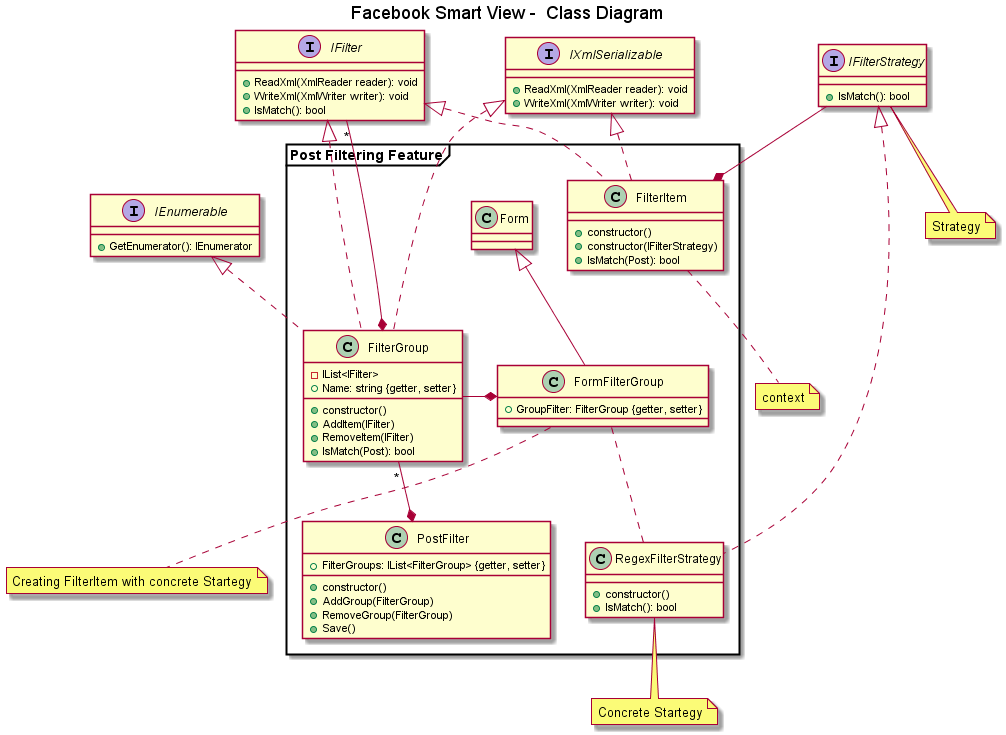
בfeature השני אנחנו מפלטרים את הtext בnews feed לפי שיטת Regex. מימשנו את Stretegy DP בFilterItem על מנת שנוכל בעתיד לשנות את אסטרטגיית הפילטור שלנו לשיטה אחרת ללא ביצוע שום שינוי בקוד בFilterItem class.

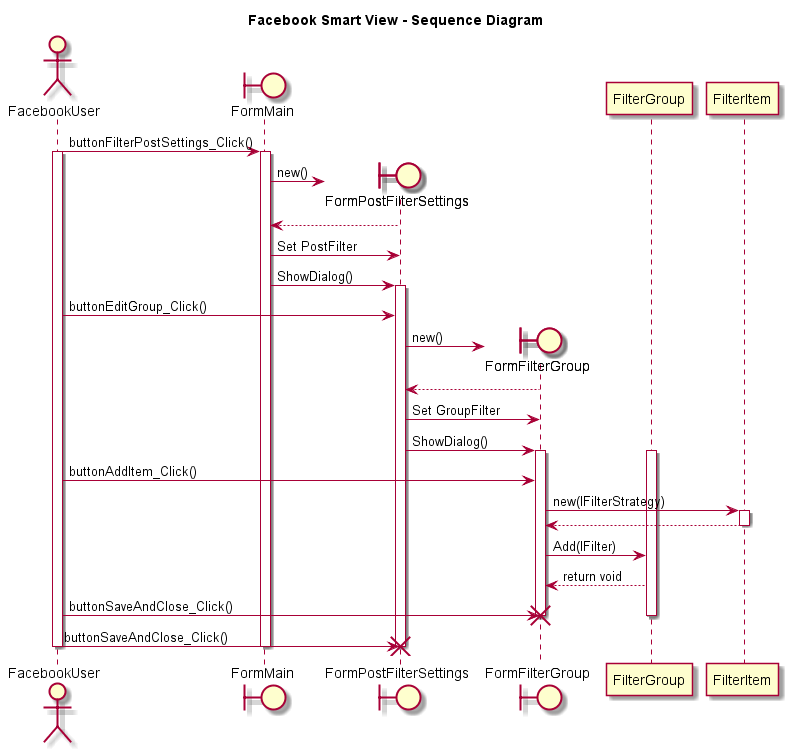
* אופן המימוש:

יצרנו IFilterStrategy Interface שקיימת בו מתודה אחת IsMatch. זהו הStrategy (ראה IFilterStrategy.cs)

יצרנו מחלקה חדשה RegexFilterStrategy שמממשת את הIFilterStrategy. זהו הConcreteStrategy (ראה RegexFilterStrategy.cs)

FilterItem מחזיק reference לIFilterStrategy אותו הוא מקבל בc’tor. וכאשר אנחנו בודקים האם יש צורך בפילטור הטקסט אנחנו מפעילים את מתודת הIsMatch של הinterface. זהו הcontext. (ראה FilterItem.cs, שורה 22 ושורה 38).

כאשר הFormFilterGroup יוצר FilterItem חדש הוא מעביר לו בc’tor את הConcreteStrategy, במקרה שלנו זה הRegexFilterStrategy. (ראה FormFilterGroup שורה 37).



### תבנית מס' 3–

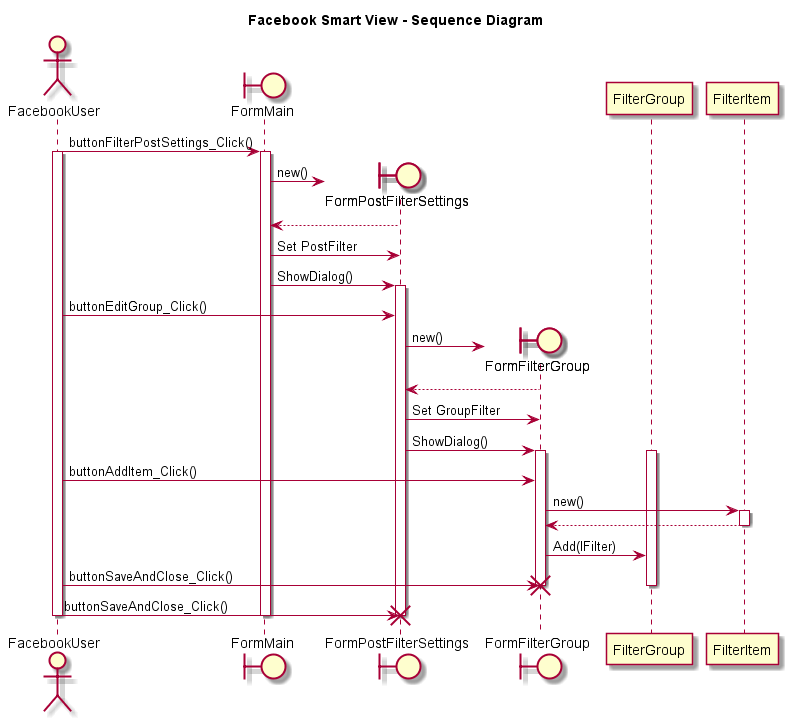
* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר מספר 2 אנחנו מבצעים סינון של הnews feed עבור מילים שנבחרו על ידי המשתמש. הוספנו את היכולת להוסיף תתי קבוצות. לדוגמא, ניצור קבוצת חגים. תחת חגים ניצור item רגיל שמכיל את המילה "שבועות" וניצור item קבוצה בשם "חגים אהובים" שמכילה את המילה "חנוכה". הפיצ'ר יסנן את המילה "חנוכה" וגם את המילה "שבועות". כלומר, עלינו להתייחס לitem ולgroup בצורה דומה ובכך שאנחנו מפעילים את המתודה שמבצעת את המיון אנחנו לא צריכים לבדוק את הtype שנמצא ברשימת המילים לסינון של המשתמש.

* אופן המימוש:

1. יצרנו interface IFilter שמחייב מימוש של IXmlSerializable ושל IsMatch Method(ראה IFilter.cs). משמש בתור הComponent
2. FilterGroup class מממש את IFilter ומכיל IList<IFilter>. בנוסף, מימוש הIsMatch שלו מבצע IsMatch על כל item ברשימה. (ראה FilterGroup.cs). משמש בתור הComposite
3. FilterItem class מממש את IFilter ומכיל את המימוש שמכיל את ביצוע הסינון המילים בnews feed(ראה FilterItem.cs). משמש בתור הLeaf.

* Sequence Diagram



* Class Diagram

