A20 EX01 Alex Girshfeld Tal Goldstein

# Introduction

“Facebook Influencers Toolkit" is a Facebook application implemented by us as part of Design Pattern course.

The application supplies a basic Facebook experience alongside supplying dashboards to allow the user to understand and analyze the influence of his "posts".

# Basic Facebook features:

1. Watch your posts - you can also preview their content in the post preview box. (the post you select to preview is the one that the influence preservation\expansion grades will be calculated to)

2.Watch your albums.

3.Watch your friends.

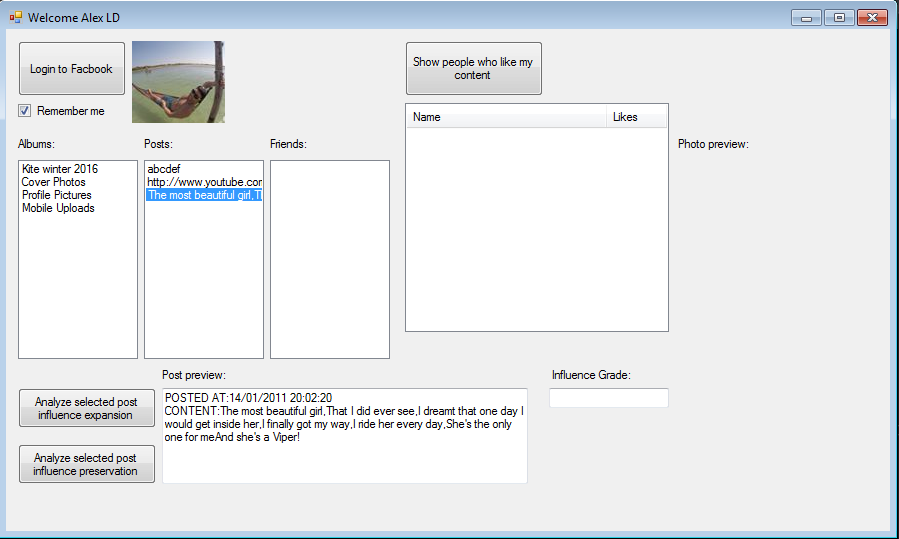
4.You can ask the application to remember your settings (window size) and to keep you logged in. It'll save your settings in your "C:\Users\Public" directory.

# Influence features:

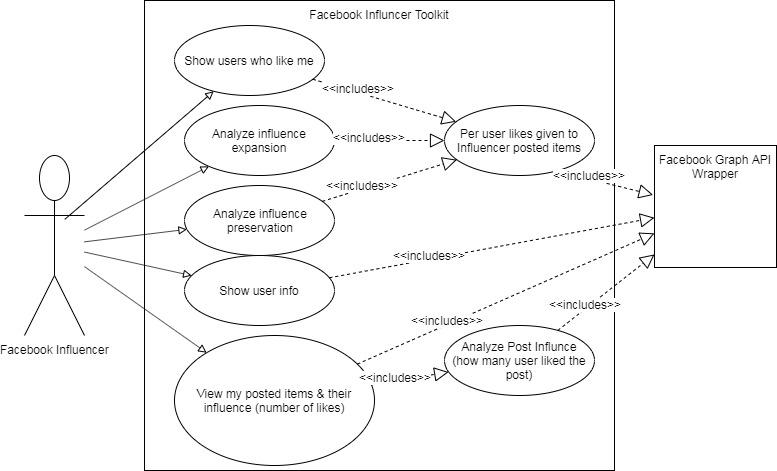
1. "Who likes me most?" - A sorted list of you friends who like you the most by the number of total likes given to your post
2. "influence preservation grade per post" - 0-100 grade of you post in order to preserve your current "fan base" of "likers" that means that if the people who always like you post didn't like this one it's "preservation grade" will be significantly lower.
3. "Influence expansion grade per post" - 0-100 grade of your post in order to expand your influence. meaning that if you want that people who before gave you less likes will now give you more likes.

\*In the Influence preservation\expansion feature there are two factors: A quantitative & a qualitative factor. The first factors the number of like compared to the avarage amount of likes you post receives and the latter takes into account **who** liked this. Example: If he never liked any of your posts he'll improve the influence expansion grade.

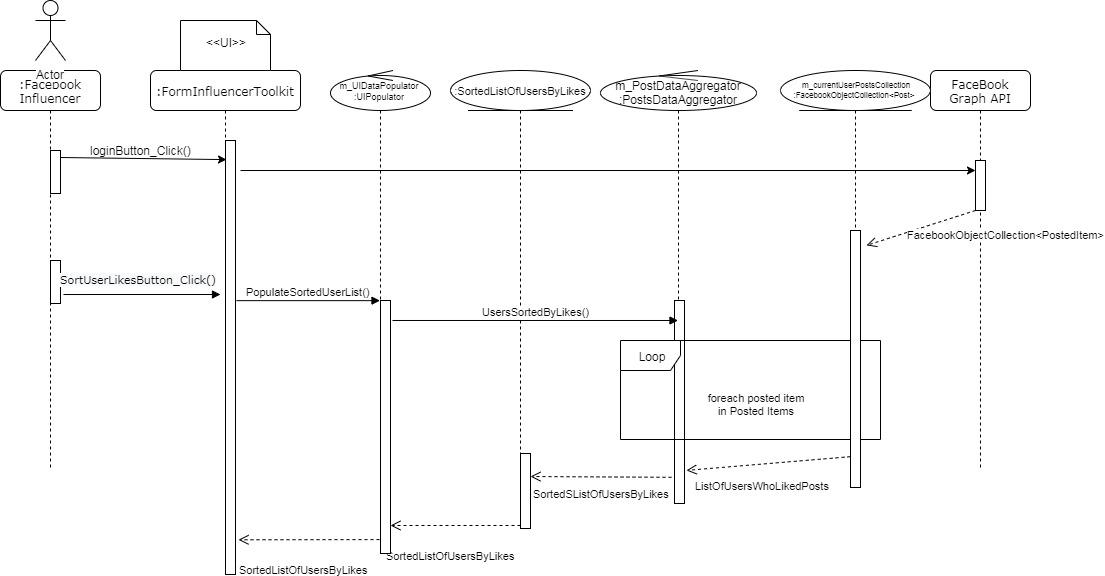
\*For features 2+3 you have to select the post you want to analyze from the list of posts and click on the corresponding button in the bottom left corner of the window. The selected post was is highlighted in blue:



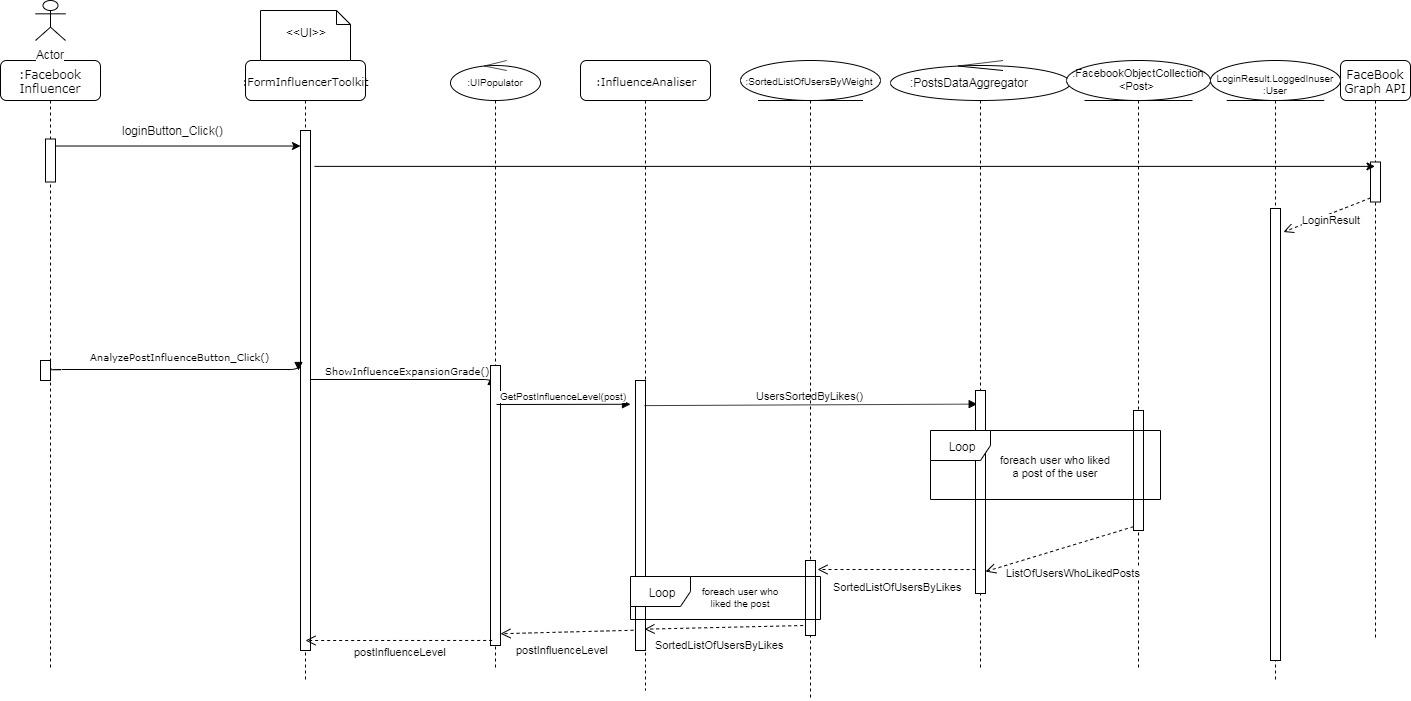
# Sequence Diagram of "Who likes me most?"



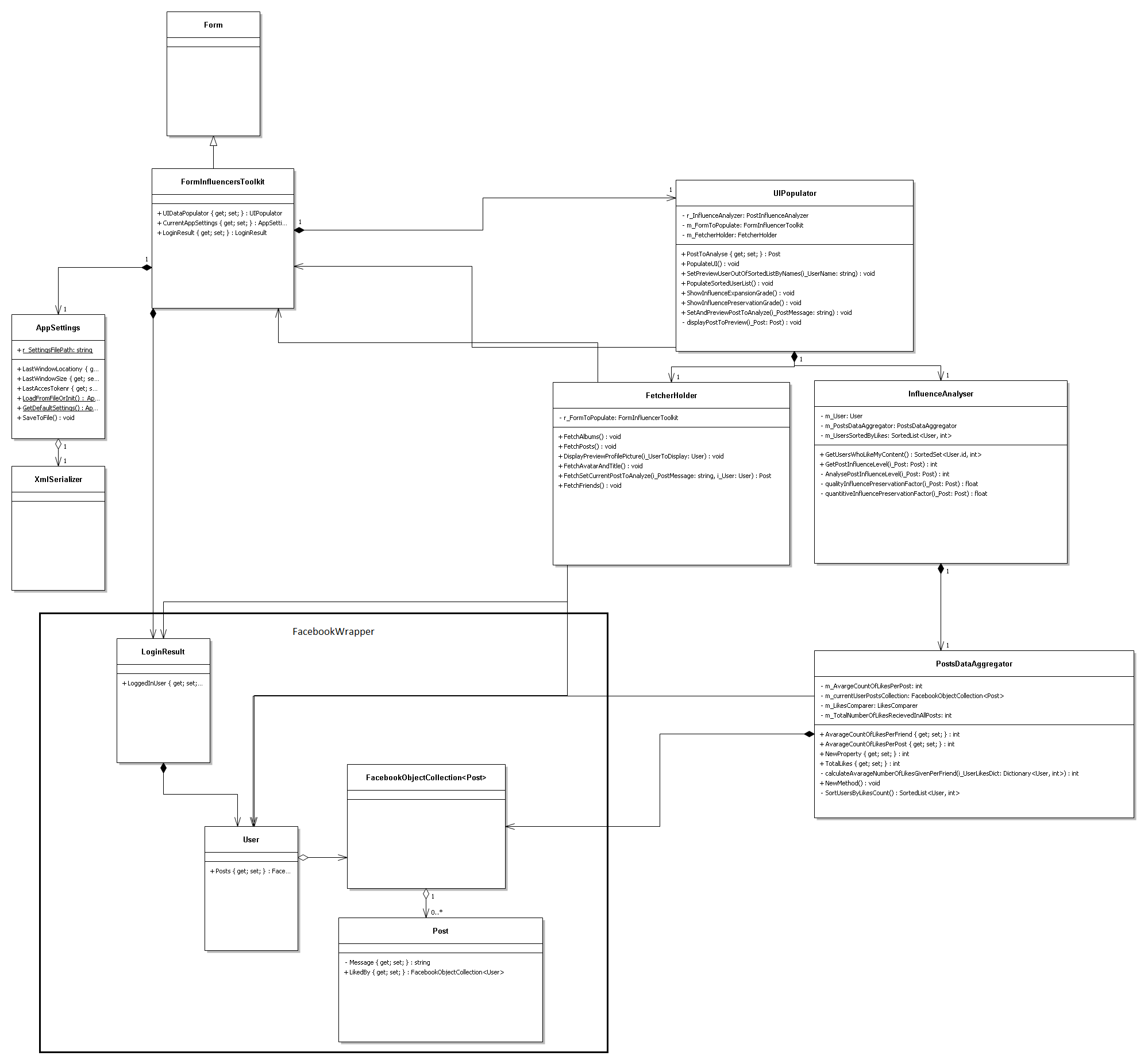
# Sequence Diagram of "Who likes me most?"



# Sequence Diagram of "Influence preservation grade per post?"



# Class diagram



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

* [תיאור קצר של הפיצ'ר הראשון]
* [תיאור קצר של הפיצ'ר הראשון]

### **תבנית מס' 1 – Facade**

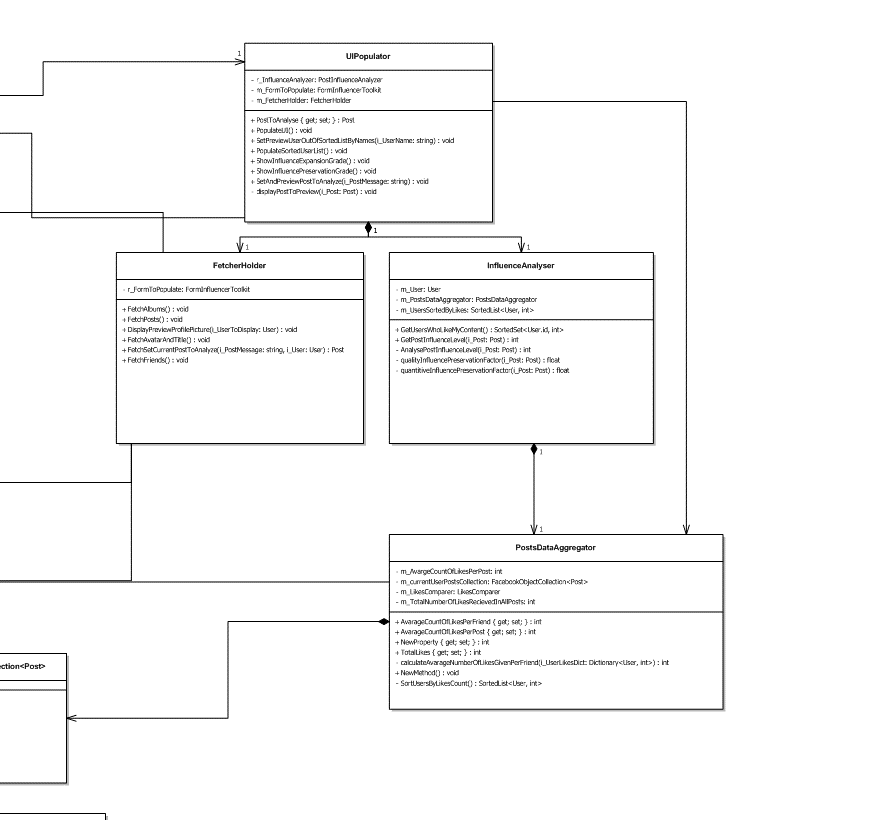
* סיבת הבחירה / שימוש בתבנית: מאפשר לבודד את החלק הלוגי המורכב של חישוב "רמת ההשפעה" של פוסטים. הרכיב העונה לשאלות מתממשק עם רכיבים שרוצים לקבל את "רמת ההשפעה" של post מסויים. במצב שלפני ה-refactor הנ"ל יש לנו שני רכיבים לוגיים   
  שה-UIpopulator משתמש בהם ויש תלות ביניהם: PostsDataAggregator ו-InfluenceAnalyser
  + השני משתמש בראשון (COMPOSITION)
  + גם ה-UIPOPULATOR – משתמש בראשון וזה מייצר משולש בהיררכיות שונות עם תלויות ש"לא קל ללמוד ולנתח" – ייתכן והמצב שנוצר הוא לא מיינטיניבילי.
  + ע"י שימוש ב-FACADE כל התלויות ירוכזו במקום אחד.
* אופן המימוש: את ה-Facade הכנסנו כרמה נוספת בין ה-UIPOPULATOR לבין שתי המחלקות איתן הוא עובד, PostsDataAggregator ו-InfluenceAnalyser. כ-Facade שקוף.  
  למעשה, ה-UIPOPULATOR אינו מכיר כעת את המחלקות הנוספות, ועובד עימן דרך ה-Facade.  
  בתוך ה-Facade ישנן כל המתודות אשר היו בשימוש על ידי ה-UIPOPULATOR.   
  בעזרת שימוש בקומפוזיציה, ה-Facade מנגיש למשתמש בו את הפונקציונליות והלוגיקה אשר בשכבה הלוגית באמצעות שיטוח למתודות פשוטות, ללא צורך להשתמש בשכבות הלוגיות השונות וללא תלות כלשהי בעומקן, או במשתנים מסוג מסויים, אלא רק ב-Facade עצמו המרכז את היכולות הקיימות ומפעיל את המתודות הרלוונטיות בהתאמה.  
  את ה-Facade מימשנו בצורה פשוטה, כך שאינו מכיל הרבה קוד ואינו משנה או מוסיף יכולות כלשהן למערכת, אלא רק משטח את המערכת על מנת להקל על ה-Client.  
  דוגמת קוד מה-Facade, ניתן לראות שימוש בקומפוזיציה ב-Constructor:

Façade

* Sequence Diagram

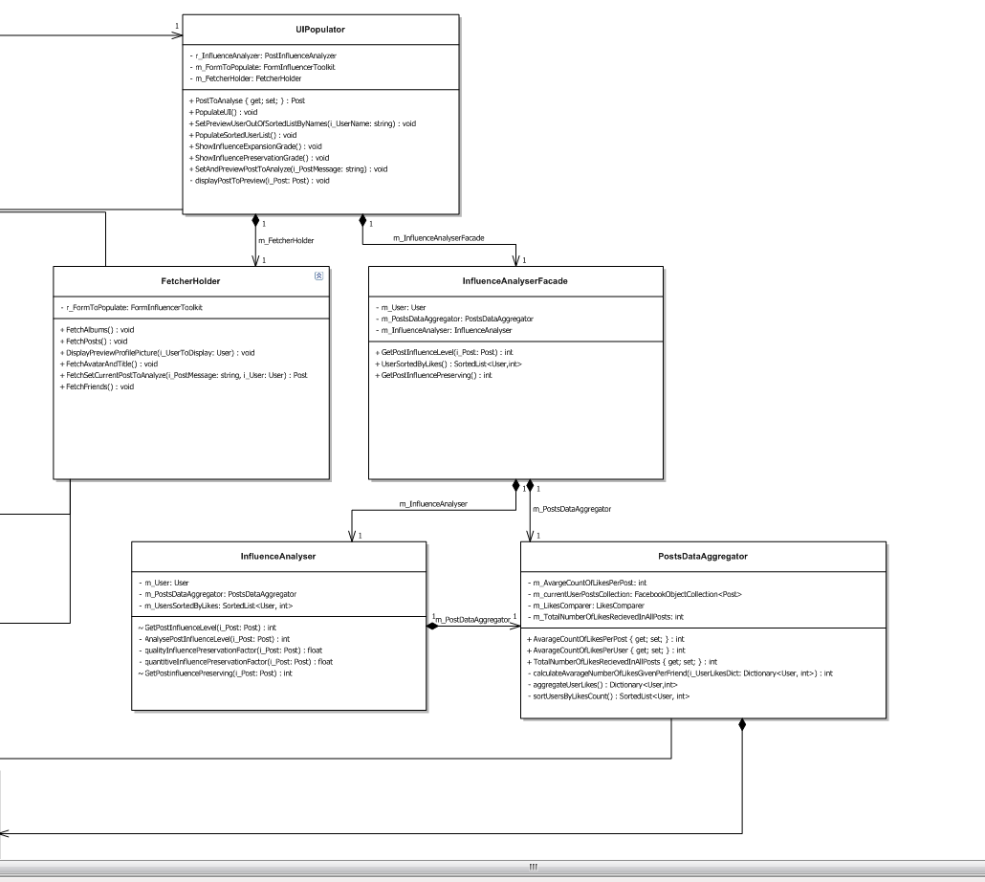
[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram
* לפני השינוי:



**לאחר השינוי:**

השטחה של הגישה של UIpopulator לשני הרכיבים: PostDataAggregator ו InfluenceAnalyser ההפעלה והגישה לשני הרכיבים עכשיו ממחלקה שנקראת InfluenceAnalyserFacade

**Façade:** InfluenceAnalyserFacade

**Class 1:** PostDataAggregator

**Class 2:** InfluenceAnalyzer

### **תבנית מס' 2 – Proxy**

* סיבת הבחירה / שימוש בתבנית:

מאפשר לנו שימוש offline באפליקציה ע"י שמירת כלל המידע הנדרש עבור רכיבי המערכת על דיסק.

* אופן המימוש:

הוספה של רכיב חדש ((FacebookWrappperCacheProxy שנותן "שירות" לכלל הרכיבים שמכירים אותו עד כה: הFacebookInfluencerForm, PostDataAggregator,, HolderFetcher

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### **תבנית מס' 3 – Singelton**

* סיבת הבחירה / שימוש בתבנית:

במהלך מימוש הCaching Proxy שמוזכר לעיל, נתקלנו בצורך לתעל את תהליך כתיבה וקריאת המידע ל\מ-הדיסק לרכיב נגיש וזמין, שקיים רק במופע יחיד במערכת וזאת על מנת:

1. למנוע גישה לא מתואמת ומסונכרנת לcache שנשמר על הדיסק.
2. להמנע מטעויות בשמירת המידע ולהפוך את התהליך ליותר קל לתחזוקה ושינויים בהמשך:
   1. לדוגמא אם יווצר רכיב CACHING נוסף עם הגדרה לשמירה בDIR – יווצרו שני מופעים של CACHE. אם הרכיב שיזין חזרה את המערכת מהדיסק הוא לא העדכני ביותר - במקרה כזה קיים סיכון שמה שיגיע למשתמש יהיה שונה מהותית מהמצופה (לא המידע הכי עדכני שהמערכת "מצפה" לו)

* אופן המימוש:

[תיאור המימוש והיכן ניתן למצוא אותו בקוד]

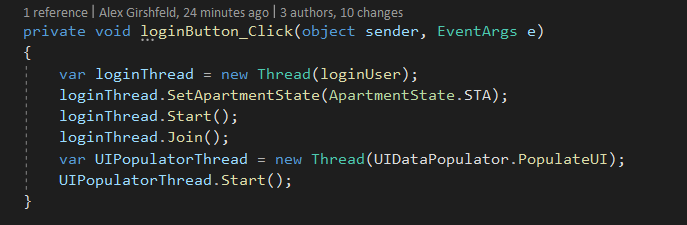
* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

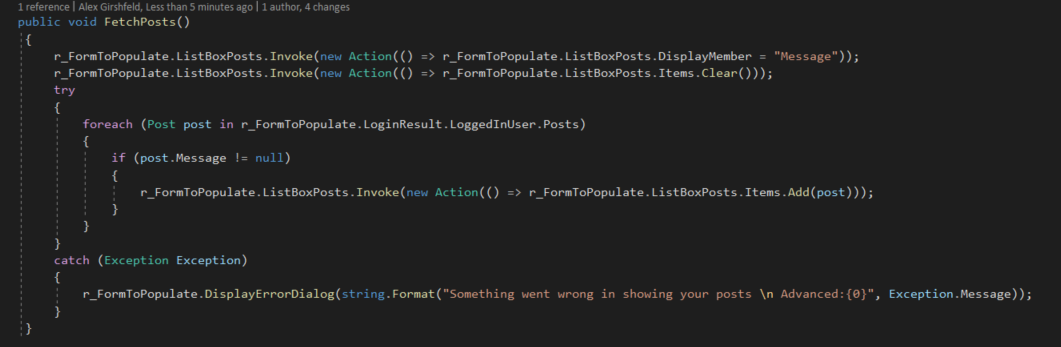
[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### **עבודה אסינכרונית –**

* אנחנו משתמשים THREADים חדשים במהלך תהליך הLOGIN:
  + הרצת מתודת הloginUser מהDLL
  + הרצת מתודת טעינת המידע לUI (PopulateUI)
* הסיבה לביצוע שתי הפעולות הנ"ל מTHREADים נוספים היא חווית משתמש יותר "חיה" ותגובתית. כלומר משתמש שביצע פעולה שלוקחת זמן (במקרה הזה עקב ממשק רשת וטעינת נתונים) לא יחווה חלון "קפוא" עד סיום הפעולות.
* הפעלת הTHREADים: 
* טעינה והפעלת הפקדים מTHREAD נפרד (UIPopulator):

אין לנו אפשרות לבצע את פעולות טעינת המידע לפקדים דרך הUIPopulatorThread (כפי שראינו בסרטון ההדרכה – פעולות כאלו נדרש לבצע מTHREAD הUI ש"מריץ" את הפקדים עצמם)

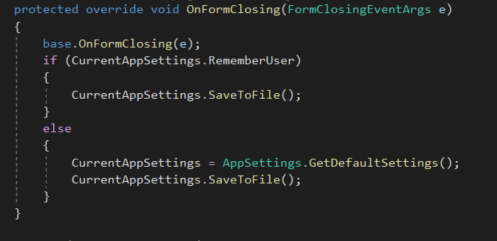
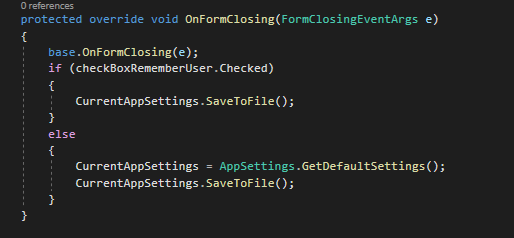
ולכן אנחנו נבצע פעולות אלו ע"י INVOKE מהפקדים עצמם ונבצע את הפעולות האלה בTHREAD הUI עם LAMBDA EXPRESSION)



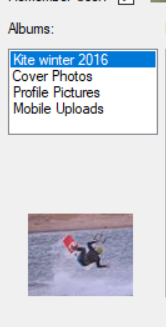
### **עבודה עם dataBinding**

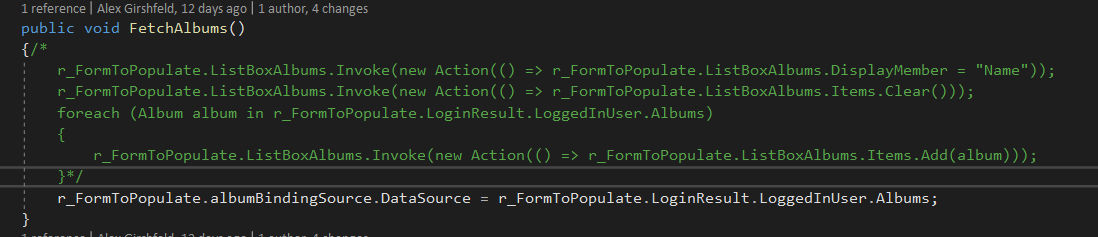
* שינינו את העבודה עם הפקד Remember user לעבודה עם two-way data binding ול CurrentAppSettings. ועכשיו אפשר שהCurrentAppSettings הופך להיות הרפרנס הבלעדי ללוגיקה בעת כיבוי החלון (הרבה יותר ברור, מיינטנבילי ונוח לשינוי)

לפני השינוי התייחסות לפקד: אחרי (התייחסות להגדרות הנוכחיות):



* שינינו את העבודה האלבומים של היוזר לtwo-way data binding. כמו שמצויין בסרטון ההדגמה הבאת ה"תוכן" (אלבומים) הוחלפה בחיבור של האלבומים שנמצאים לאובייקט albumBindingSource שבתורו דואג לאכלס pictureBox שמציג את תמונת האלבום לפי בחירת המשתמש.





TODO:

* General:
  + Databinding – Alex possible binding via fetch methods in UIpopulator
  + AsyncWorking with multithreading
  + Updating Class diagram - Alex
* Façade:
  + Code - TAL
    - Extracting the two public methods from influence analyzer
    - Moving the usage of post aggregator by UIpopulator (now uses postAgregator by himself)
    - The new façade class holds the post aggregator, Influence analyzer and the interactions between them
  + Sequence diagram - Tal
* Caching Proxy (UserCachedProxy)
  + Code
  + Features:
    - Display of the last updated time
    - Maximum update interval
    - Each time the data being accessed – check if update needed (max update interval)
    - Uses the FacebookObjectCacher to save the data
  + Sequence Diagram - Tal
* FacebookObjectCacher
  + Code
  + Sequence Diagram - Tal