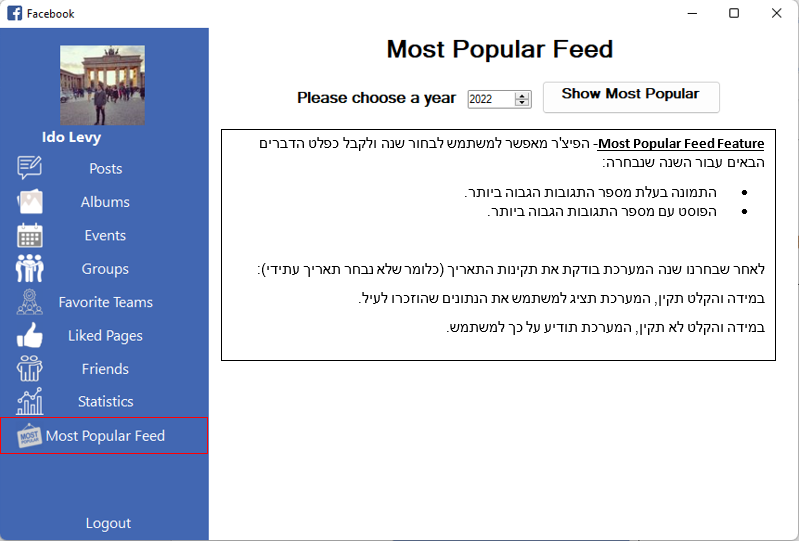
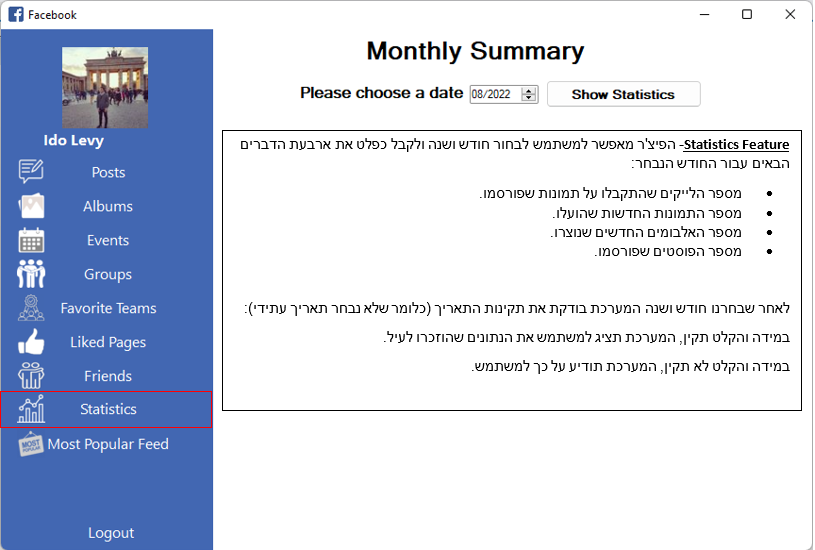
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:





### תבנית מס' 1 – Singleton

* **סיבת הבחירה / שימוש בתבנית:**

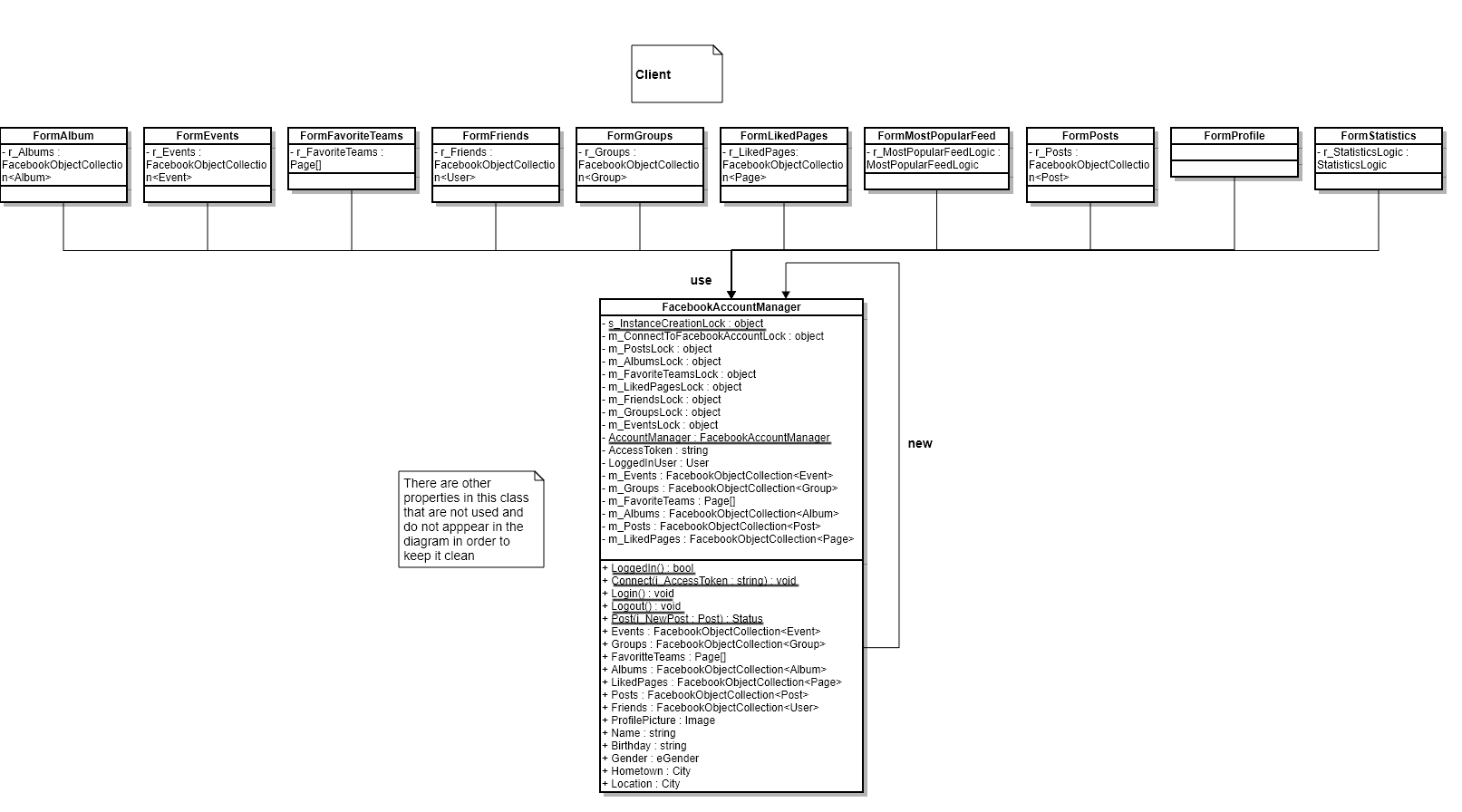
המידע אודות המשתמש נחוץ במספר רב של נקודות באפליקציה (בטפסים השונים, בשכבה הלוגית וכדומה), וכן נחוץ לדעת האם המשתמש מחובר או לא מחובר למערכת בעת הפעולה הרצויה, על מנת למנוע פעולות לא חוקיות בחשבון המשתמש ו/או באפליקציה עצמה.

על מנת שהמידע הנ״ל יהיה נגיש לכל נקודה ונקודה במערכת, מבלי שנצטרך לכפות תלויות על גופים אלו, ובכדי למנוע שכפול קוד ואף לאפשר סינכרוניזציה בין הגופים השונים, ננגיש את המידע הנ״ל לכל בכך שנגדירו כ-Singleton, ובכך כל אובייקט לאורך חיי התוכנית יוכל לגשת ביוזמתו אל המידע אודות המשתמש ולדעת מה מצבו.

* **אופן המימוש:**

המחלקה FacebookAccountManager משמשת כ-Singleton.  
ניתן לראות שימוש במחלקה זאת במחלקות הבאות:   
FormMain, FormAlbums, FormEvents, FormFavoriteTeams, FormFriends, FormGroups, FormLikedPages, FormPosts, FormProfile, MostPopularFeedLogic, StatisticsLogic

* **Diagram, schematic

  Description automatically generatedSequence Diagram**
* **Class Diagram**
* Client -> All subforms
* Singleton -> FacebookAccountManager

### תבנית מס' 2 – Factory Method

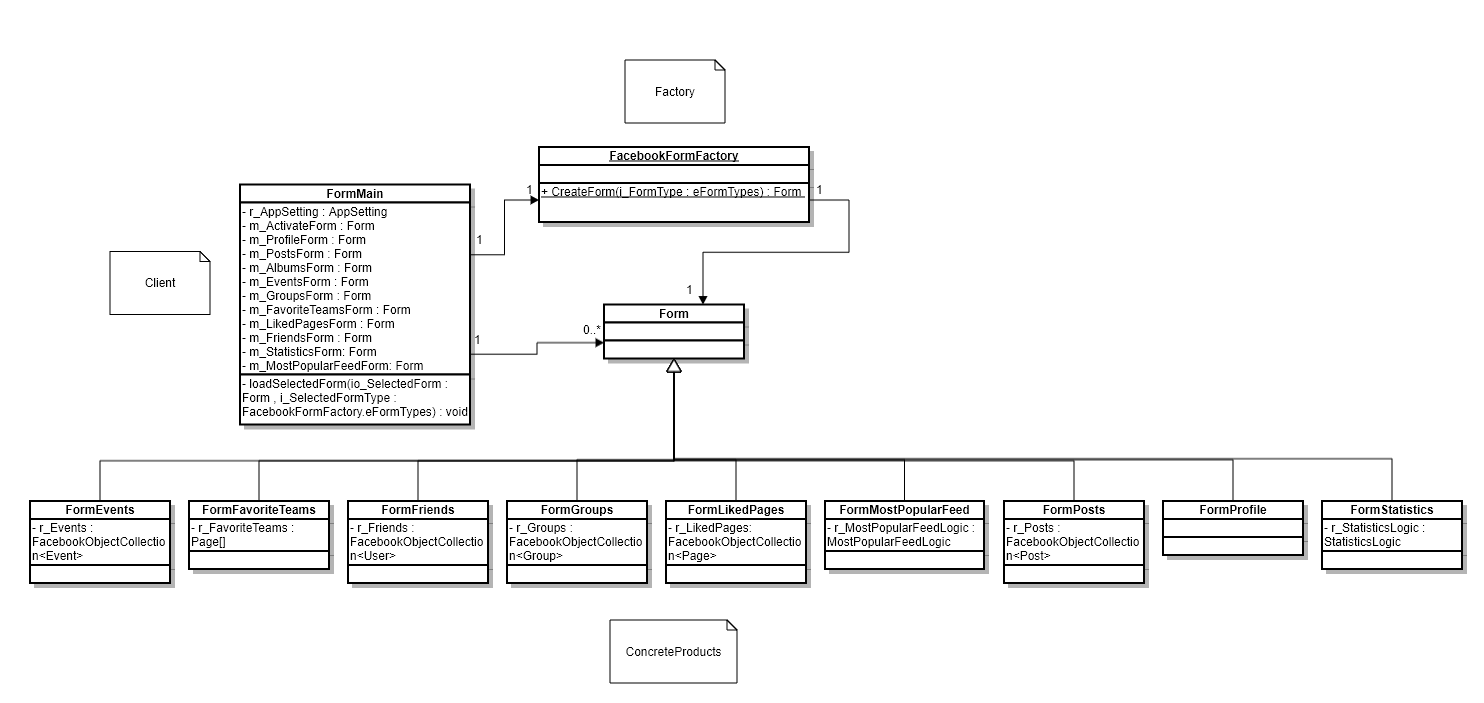
* **סיבת הבחירה / שימוש בתבנית:**

בעת השימוש באפליקציה, המשתמש ירצה לעבור ולהתבונן בנתונים שונים בחשבונו, ועל כך נכון יהיה להפריד בין קטגוריות נתוני המשתמש למספר טפסים שונים. על כן, נדרש לייצר את הטפסים השונים בזמן הרצת האפליקציה, כאשר כל טופס וטופס בעל דרישות שונות ומצריך לוגיקה שונה.  
על מנת להקל את המשתמש ביצירת הטפסים השונים, ובכדי ״להוריד ממנו״ את האחריות לשנות את מימוש יצירת הטפסים הנ״ל, אחריות יצירת הטפסים עוברת אל גורם המיועד לכך – FacebookFormFactory, כך שבכל פעם שהמשתמש יצטרך לייצר טופס כלשהו באפליקציה, המחלקה הנ״ל תדאג לכך הלוגיקה הנדרשת לכך, ובכך תפחית את כמות הבאגים ואת הצורך בתחזוק המערכת.  
בנוסף, אם נרצה להחליף את שרתי האפליקציה, כל שהיא תתמוך בשרתי Twitter למשל, כל שנצטרך לעשות הוא להחליף את סוג ה-Factory, ובכך המערכת תמשוך לתפקד כרגיל.

* **אופן המימוש:**

המחלקה FacebookFormFactory אחראית על יצירת הטפסים השונים באפליקציה, וזאת ע״י CreateForm, אשר מקבלת את סוג הטופס המבוקש מתוך קבוצה סופית של סוגי טפסים (Enum).  
המחלקה שמשתמשת במחלקה זו היא FormMain, המבקשת מ-FacebookFormFactory לייצר את הטפסים השונים בפעם הראשונה לפי בקשת המשתמש.

* **Diagram

  Description automatically generatedSequence Diagram**
* **Class Diagram**
* StaticFactoryClass -> FacebookFormFactory
* Client -> FormMain
* ConcreteProducts -> All forms in FormMain

### תבנית מס' 3 – Caching Proxy

* **סיבת הבחירה / שימוש בתבנית:**

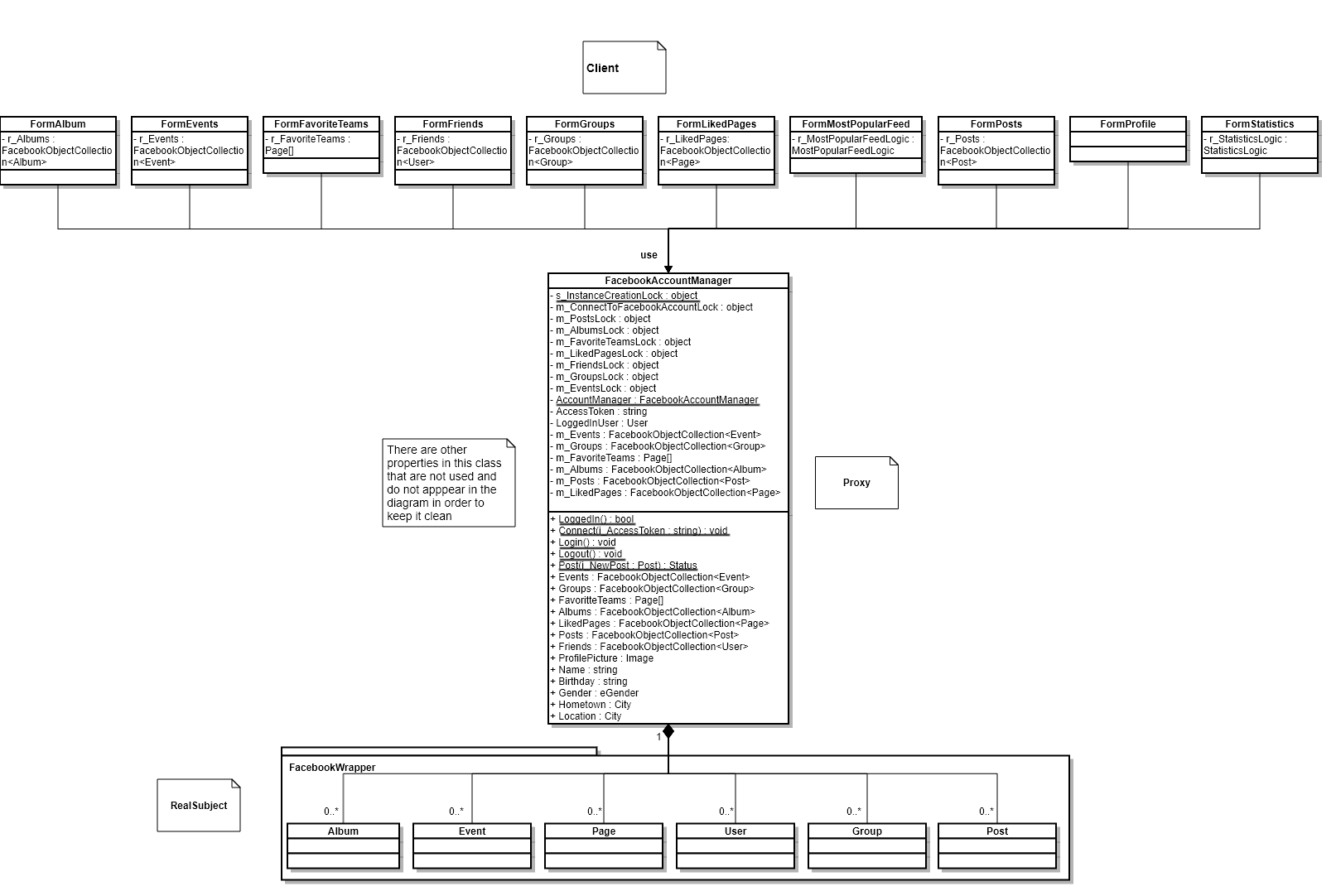
באפליקציה ישנן מספר נקודות בהן נעשה שימוש במידע שכבר ״נמשך״ משרתי פייסבוק ע״י גורם אחר קודם לכן, ואף ע״י אותו גורם עצמו שכעת מבקש להשתמש באותם משאבים שכבר היו בבעלותו בעבר.

על מנת לשפר את יעילות התוכנית, הן מבחינת ביצועים (מהירות) והן מבחינת זיכרון, נשמור את הנתונים שכבר נמשכו מהשרתים קודם לכן, ולאחר משיכתם לראשונה, לא יימשכו בשנית, ובכך נשפר את ביצועי התוכנית כפי שציינו לעיל.

* **אופן המימוש:**

המחלקה FacebookAccountManager משמשת כ-לCaching Proxy.  
ניתן לראות שימוש במחלקה זאת במחלקות הבאות:  
 FormMain, FormAlbums, FormEvents, FormFavoriteTeams, FormFriends, FormGroups, FormLikedPages, FormPosts, FormProfile, MostPopularFeedLogic, StatisticsLogic

* **Diagram, schematic

  Description automatically generatedSequence Diagram**
* **Class Diagram**
* Proxy -> FacebookAccountManager
* Client -> All sub forms
* RealSubject -> Classes in FacebookWrapper

### עבודה אסינכרונית

* נשתמש בעבודה אסינכרונית ...
* ניתן לראות שימוש בעבודה אסינכרונית במחלקות הבאות:FormMain, FormAlbums, FormEvents, FormFavoriteTeams, FormFriends ,FormGroups, FormLikedPages, FormPosts, FormProfile, FormMostPopularFeed, FormStatistics

### עבודה עם Data Binding

* נשתמש ב-Data Binding...
* ניתן לראות שימוש ב-Data Binding במחלקות הבאות: FormAlbums, FormEvents, FormFavoriteTeams, FormFriends ,FormGroups, FormLikedPages, FormPosts