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

* הצגת פרטי חמשת החברים שעשו הכי הרבה לייקים.
* הצגת אזורים ומקומות שהמשתמש ביקר והעלה מהם תמונות, פוסטים וכו'

### שימוש באסינכרוני -Threads

הפעלת המתודות fetchFriends,fetchPlaces,fetchTopLikes לוקחות זמן רב בשל גישה למאגר הנתונים של Facebook.

בשל כן השתמשנו במתודות אלה בthread חדש על מנת שהמשתמש יוכל לבצע פעולות נוספות בזמן שמתבצעת גישה לשרתי פייסבוק

לדוגמא :

if (topLikesListBox.Items.Count == 0)

{

Thread thread = new Thread(new ThreadStart(fetchTopLikes));

thread.Start();

}

topLikesButton.Invoke(new Action(() => topLikesButton.Enabled = false));

mostLikesList = m\_LoggedInUser.Friends.FetchTopLikeFriends(5);

topLikesButton.Invoke(new Action(() =>topLikesListBox.DisplayMember = "name"));

foreach (Friend friend in mostLikesList)

{

topLikesButton.Invoke(new Action(()=> topLikesListBox.Items.Add(friend)));

}

topLikesButton.Invoke(new Action(() => topLikesButton.Enabled = true));

### שימוש ב - Data Binding

TwoWayBinding :

השתמשנו ב basicEventDataBindingSource וקישרנו אותו לרשימת האירועים בטופס FormFacebookWindow.

כאשר המשתמש בוחר אירוע, מוצגים לו בין היתר שם האירוע ותיאור נוסף על האירוע.

כאשר המשתמש משנה את תיאור האירוע, התיאור נשמר בזיכרון התכנית.

### תבנית מס' 1 –Singleton

* סיבת הבחירה / שימוש בתבנית:

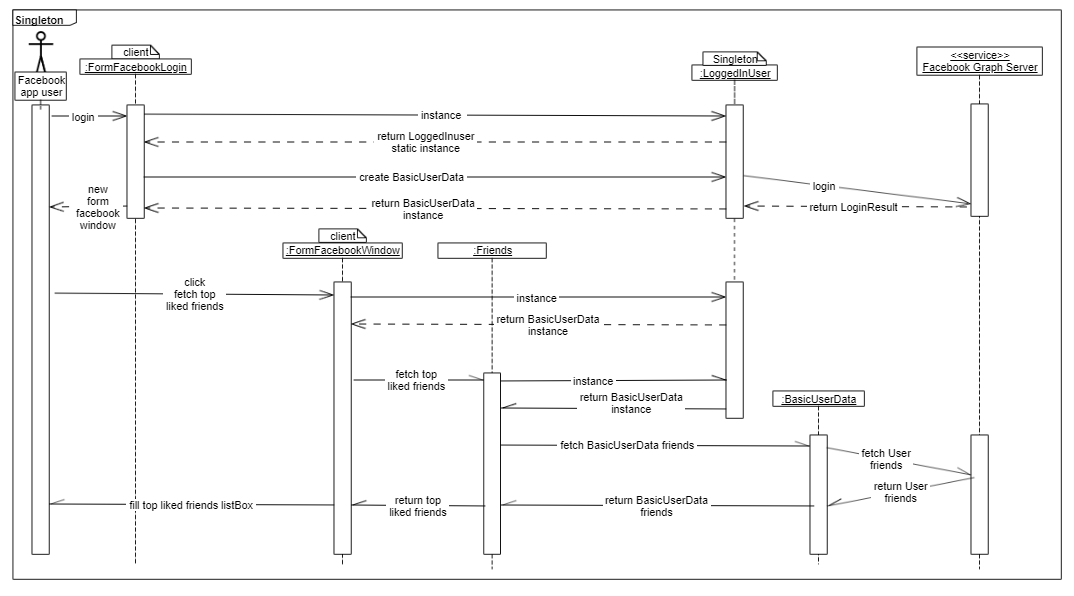
1. הצורך בגישה למחלקה **LoggedInUser** ממקומות שונים במערכת הנוכחית + גישה ממקומות עתידיים במערכת (מחלקות שונות עושות שימוש במידע הקיים במחלקה **LoggedInUser**).
2. אין צורך ואין משמעות ליצירה של יותר ממופע אחד של המחלקה (מאחר ואנו לא רוצים לאפשר כיום ובעתיד התחברות בו זמנית מ-2 משתמשים שונים)
3. בטיחות שימוש - אנו מונעים יצירה נוספת ממקומות שונים במערכת, כך שאם אובייקטים עתידיים ירצו לגשת למחלקה זאת לא יוכלו לעשות זאת ביוזמתם.

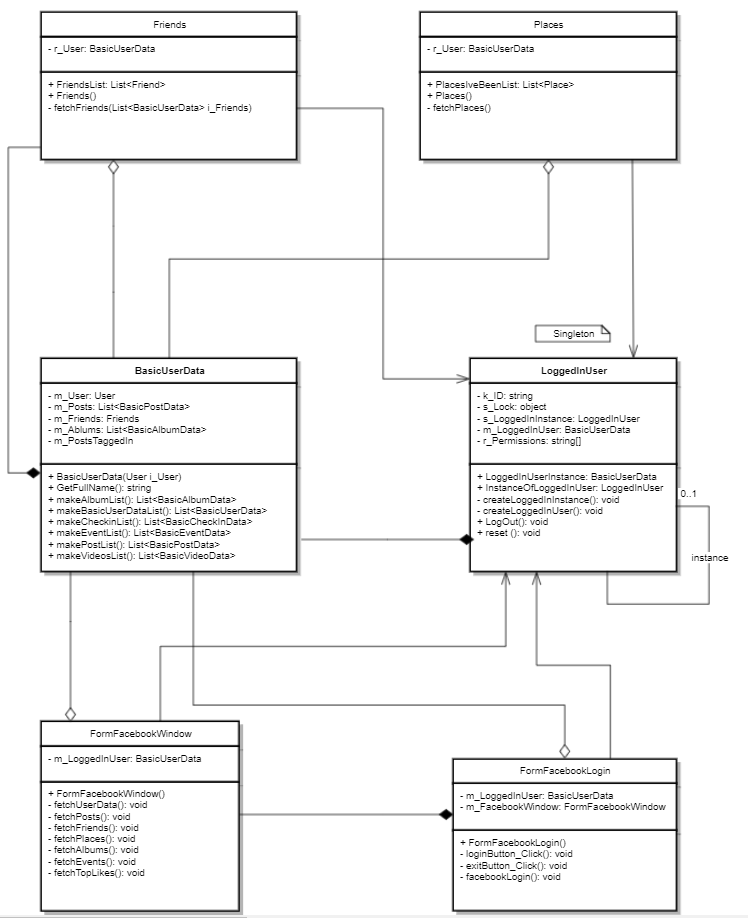
* אופן המימוש:

המימוש נמצא בפרויקט הלוגי Engineבמחלקה LoggedInUser ((singleton. המימוש נעשה ע"י שימוש במופע סטטי של המחלקה עצמה, מימוש של, private ctor ע"מ למנוע יצירה של האובייקט ממקומות שונים ולא רצויים במערכת. ומימשנו פונקציה:

public static LoggedInUser InstanceOfLoggedInUser שניתן לקבל את הinstance , ואם הוא טרם נוצר היא גם יוצרת אותו.

* Sequence Diagram



* Class Diagram

### תבנית מס' 2 –Façade

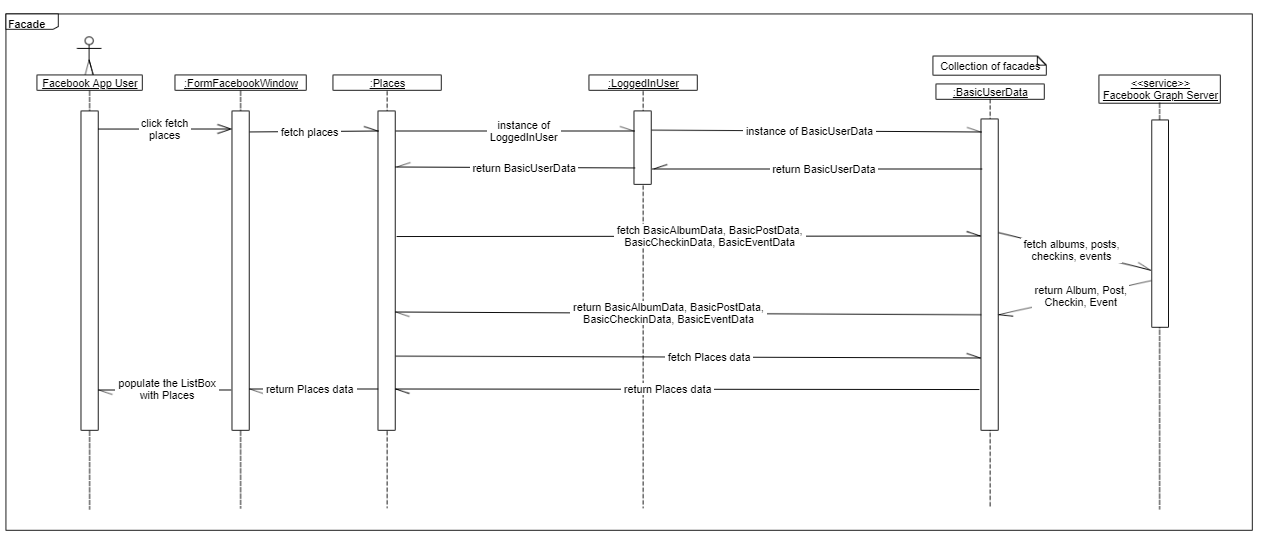
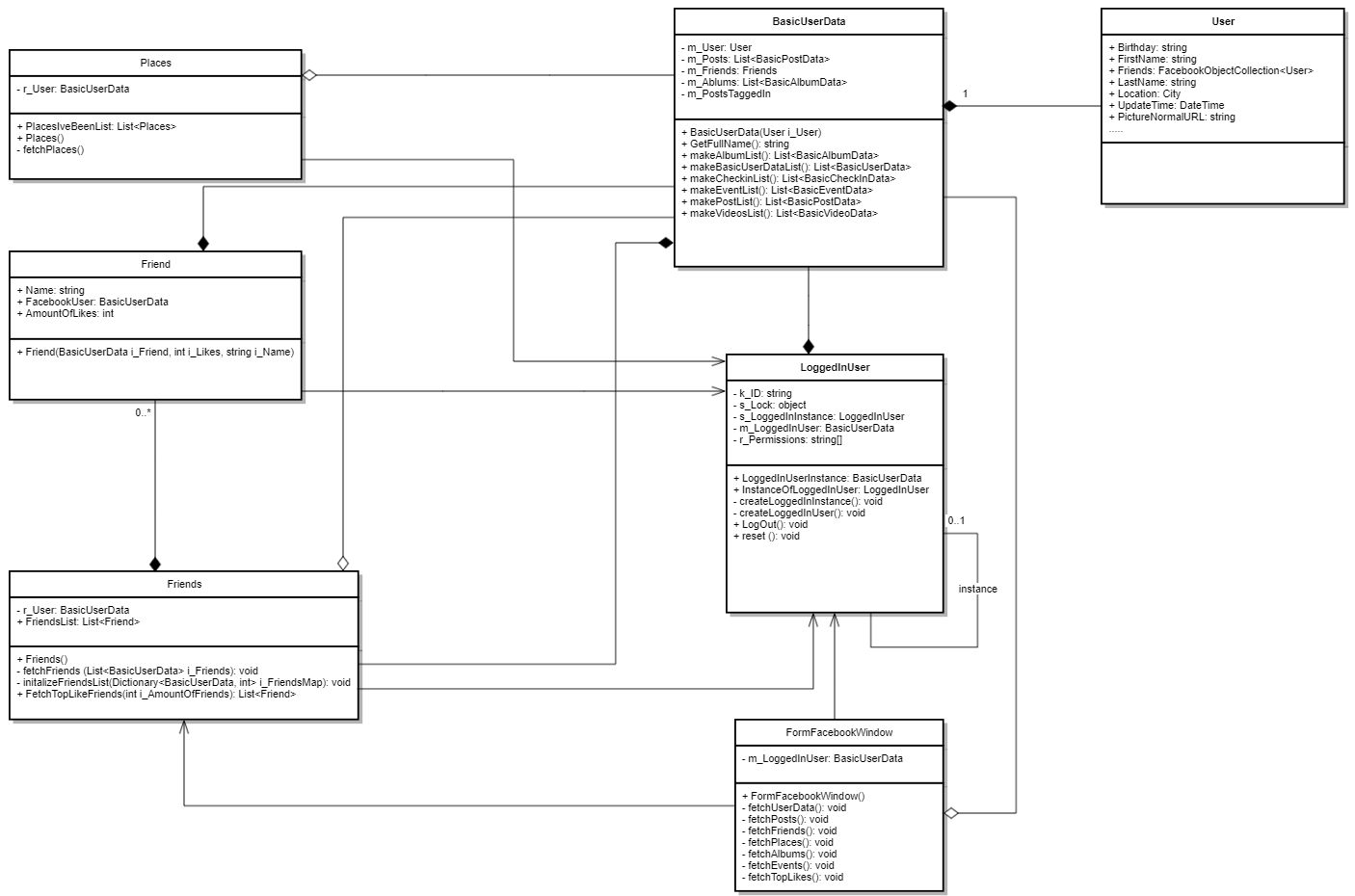
* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית זו כדי לפשט עבור השימוש בUI מול המידע הרלוונטי של האובייקט FacebookWrapper.ObjectModel.User (המשמש כתת מערכת) שמכיל הרבה נתונים ומתודות שלא רלוונטיות למשתמש. לכן, יצרנו מחלקה שבה אנחנו משתמשים כדי לאפשר שימוש קל בתת מערכת זו. כמו כן, קיים יחס תלותי בין האובייקט User לטופס הממשק (מתבצע עדכונים לחברי המשתמש). ולכן התקשורת בין האובייקא לטופס מתבצע ע"י המחלקה שיצרנו שמשמש כ- Façade.

* אופן המימוש:

יצרנו מחלקה בפרויקט הלוגי Engineבשם BasicUserData שהיא משמשת כ- Façade. היא מכילה מופע של User וחושפת למשתמש רק את המתודות והנתונים הרלוונטים למשתמש (פרטי הUser המבוקש)

בעזרת תבנית זו הצלחנו להכמיס את תהליך המעבר על הפרטי המשתמש/החברים, וקבלת הפרטים של המשתמש והחברים ועדכונם לפי הצורך.

* Sequence Diagram
* ****Class Diagram

### תבנית מס' 3 – Adapter

* סיבת הבחירה / שימוש בתבנית:

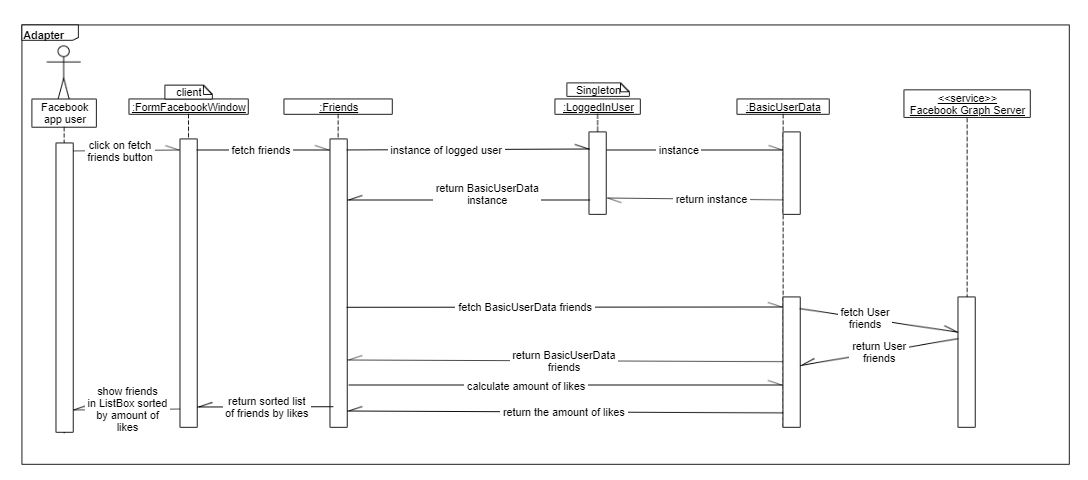
בכדי להציג את רשימת החברים ממויינים בסדר יורד לפי סך כל הלייקים שכל חבר נתן למשתמש. לשם כך, יצרנו מחלקה המשתמשת בFacebookWrapper, מחשבת את מס' הלייקים שכל חבר עשה וממיינת את הרשימה בהתאם לזאת. המשתמש עובד עם מחלקה זו ולא ישירות עם הFacebookWrapper.

המשתמש מכיר את הAdapter ועובד מולו באופן זהה, והAdapter ניגש לFacebookWrapper.

* אופן המימוש:

מימשנו מחלקה, Friends שאחראית על יצירת מאגר החברים שהם מטיפוס Friend אשר מכילה את המידע הרלוונטי (את שם המשתמש וסך כל הלייקים). יצרנו רשימה מטיפוס Friend המכילה את כל החברים ומיינו אותה לפי מס' הלייקים שכל חבר נתן בסדר יורד.

* Sequence Diagram



* Class Diagram

