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

* [תיאור קצר של הפיצ'ר הראשון]

**Friends Checks In** - the feature displays a check in list of place name of all the user’s friends.

The user can select a particular check in using two different ways:

1. Manual search by name of particular place.
2. Selection from given checking list.

Then, the list of friends who have checked in at this place before will show up.

The user able to select a particular member and get more details about that check in such a picture of the checkin

* [תיאור קצר של הפיצ'ר השני]

**Activity filter** - The feature is input of the user that the user selects:

1. Year of start and end year in which the user was active.

2. Option to select photos or posts (there is an option to select both).

3. if you choose photos you have the option to filter them by minimum comments.

After pressing the Start button, the selected options will appear.

It will be possible to browse through the images by clicking buttons.

### תבנית מס' 1 – Strategy

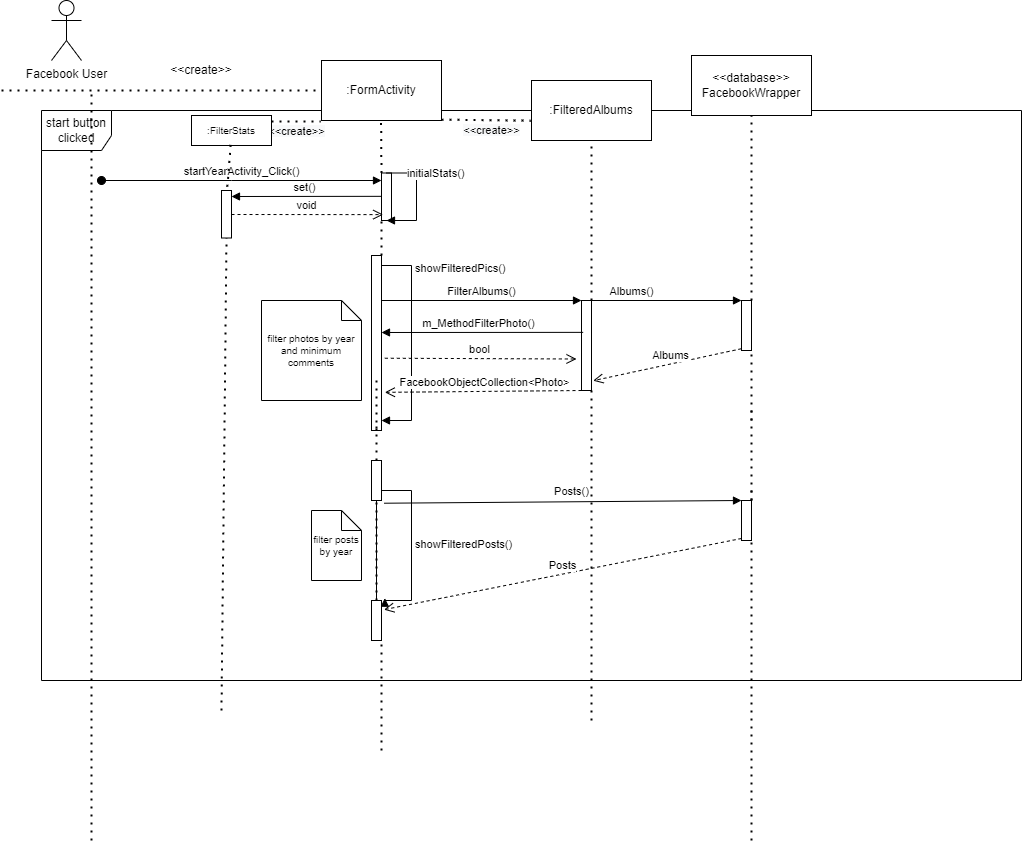
* סיבת הבחירה / שימוש בתבנית:

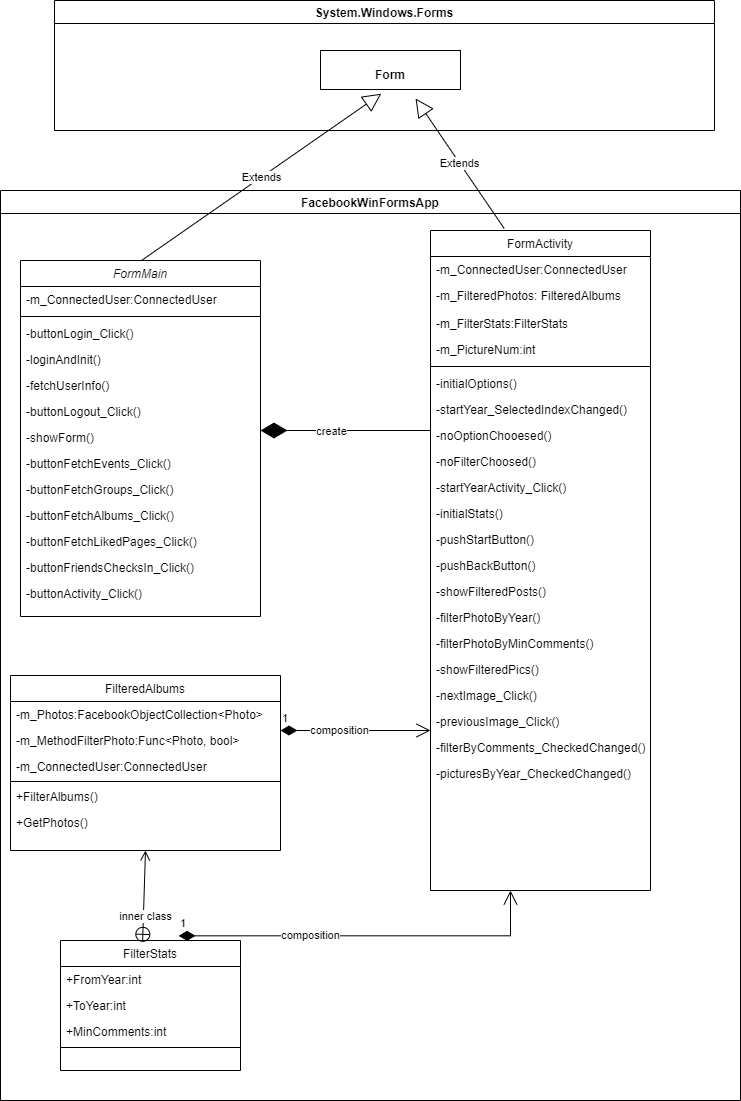
בפיצ'ר Activity filter המשתמש יכול לפלטר תמונות עפ"י מספר תגובות מינימאלי ובין אילו שנים הועלתה התמונה. אנו רוצים לאפשר בעתיד לבצע את פעולת הפילטור על ידי פרמטרים אחרים מבלי לשנות את קוד המחלקה(את אלגוריתם הפילטור ונתוני הפילטור ניתן לשנות).

* אופן המימוש:

במחלקה FilteredAlbums ישנו משתנה מחלקה מסוג Func<Photo,bool> אשר מחזיק את מתודת הפילטור על פיה ממיינים תמונה.

בנאי המחלקה מקבל את המתודה שבעזרתה יתבצע הפילטור ומזין אותה בmember הנ"ל.

* Sequence Diagram
* Class Diagram



### תבנית מס' 2 – Chain of responsibility

* סיבת הבחירה / שימוש בתבנית:

כחלק מפיצ'ר חדש שנוסף לאפליקציה, המאפשר למשתמש לשנות תיאור של תמונה במהלך מעבר על התמונות באלבומים שלו, הוספנו מחלקה חדשה שתבדוק האם התיאור החדש הולם ובלי שגיאות כתיב.

השתמשנו בתבנית על מנת לבצע שרשרת בדיקות ואף לתת מקום כך שבעתיד יהיה ניתן להוסיף עוד בדיקות או פעולות במקרה הצורך.

המערכת שלנו מבצעת מס' בדיקות אשר פועלות כשרשרת של אחריות מאחת לשנייה.

המשתמש מכניס תיאור חדש לתמונה, אשר עוברת לבדיקה האם קיימות מילים גסות, בעזרת הספרייה החיצונית ProfanityFilter, לאחר מכן התיאור החדש מופנה לטיפול על ידי השלב השני בשרשרת שם מתבצעת בדיקת דקדוק בעזרת הספרייה החיצונית NHuspell.

באם התיאור החדש יעבור את שתי הבדיקות הנ"ל התיאור ייקלט במערכת, אחרת המערכת תציג הודעה מתאימה למשתמש.

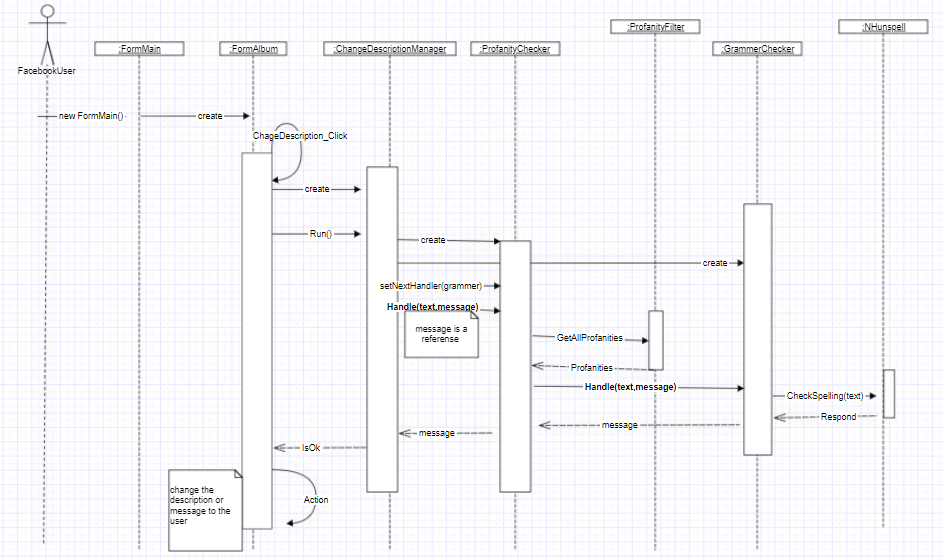
בעזרת התבנית נוכל להוסיף בדיקה נוספת לשרשרת בצורה נוחה.

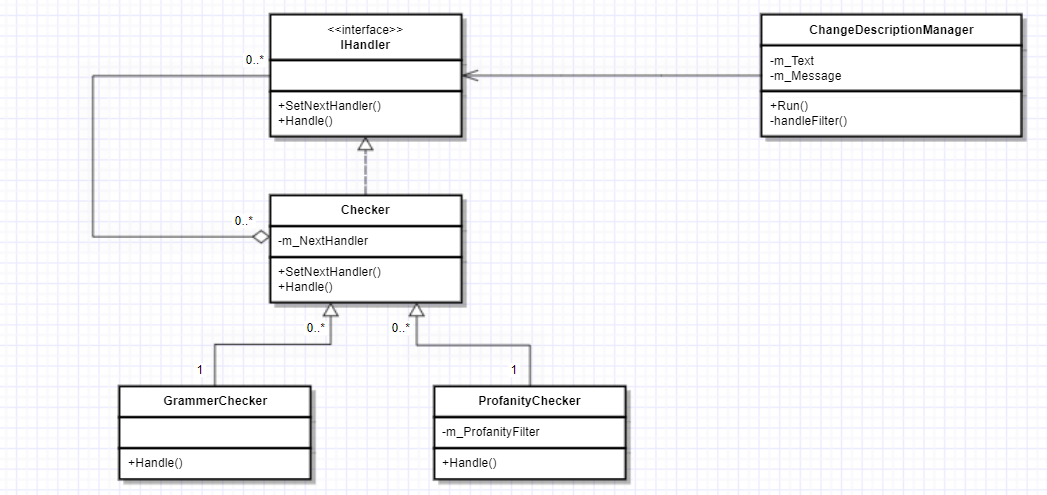
* אופן המימוש:

בלחיצה על כפתור ButtonDescription שמוצג בFormAlbum מופעלת המתודה Run שנמצאת במחלקה ChangeDescriptionManager.

המחלקה יוצרת את שני השלבים בשרשרת הבדיקות ProfanityChecker וGrammerChecker מחלקות אשר יורשות מChecker שמממשת את הinterface IHandler ובה המתודות Handle ו SetNextHandler. לאחר מכן מחברים את מחלקות הבדיקה לכדי שרשרת הגיונית וקוראים לפונקציה Handle של המחלקה הראשונה בשרשרת הבדיקות.

שרשרת הבדיקות מחזירה הודעה) string) המשמש כשדה במחלקה ChangeDescriptionManager ובה השגיאות שנמצאו במהלך הבדיקות, ואם אין שגיאות, הודעה ריקה.

* Sequence Diagram
* Class Diagram



### תבנית מס' 3 – Observer

* סיבת הבחירה / שימוש בתבנית:

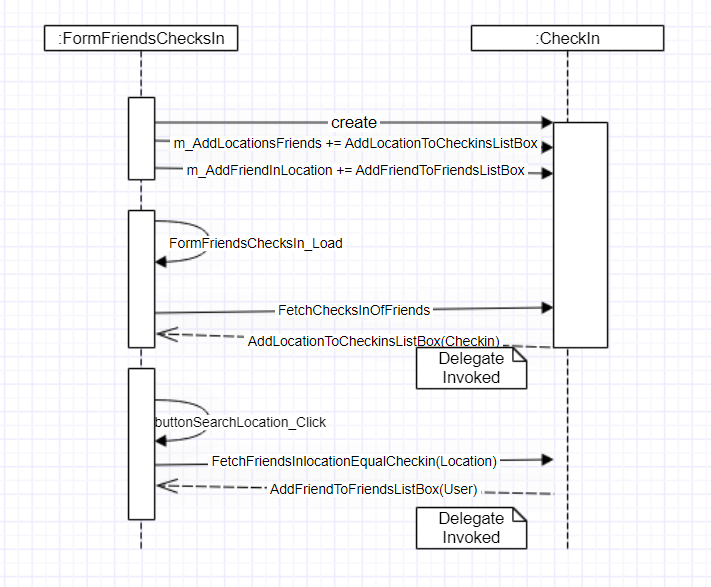
בפיצ'ר friends Check In המחלקה CheckIn היא זאת שמבצעת את הלוגיקה של חיפושי המקומות שבהם החברים של המשתמש ביקרו ואת חיפושי החברים שהיו במקומות האלה.

הנתונים אמורים להתבסף לListBox המתאימים בטופס FormFriendsCheckins ועל מנת לא ליצור תלות בין הטופס לבין הלוגיקה מימשנו בכל פעם ש CheckIn מוצאת נתון שצריך להיכנס לListBox המתאים, היא מפעילה מתודה אצל FormFriendsCheckins .

* אופן המימוש:

מימשנו במחלקה CheckIn שני public event המציינים אירועים של מציאת Checkin של חברים ומציאת חברים שהיו בlocation מתאים . לאירועים נרשם כמאזין ה FormFriendsCheckins והוא מפעיל את המתודות הרלוונטיות לכל Event .

* Sequence Diagram



* Class Diagram

תמונה שמכילה שולחן

התיאור נוצר באופן אוטומטי

### תבנית מס' 4 – Singelton

* סיבת הבחירה / שימוש בתבנית:

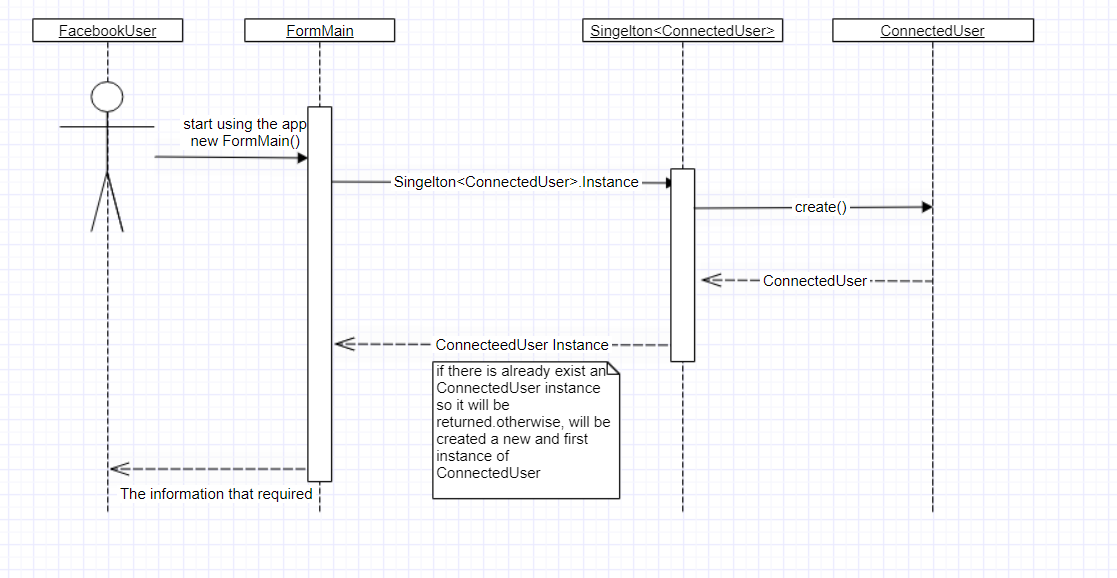
בחרנו לממש את המחלקה ConnectedUser כסינגלטון כי זו היא המחלקה המחזיקה את כל המידע על הUser שמשתמש באפליקציה. במידה ויהיו כמה מופעים של המחלקה, אז יהיו מספר משתמשים המחוברים לאפליקציה בו זמנית דבר שעלול לגרום לחוסר אמינות בנתונים ו\או להתנהגות לא צפויה.

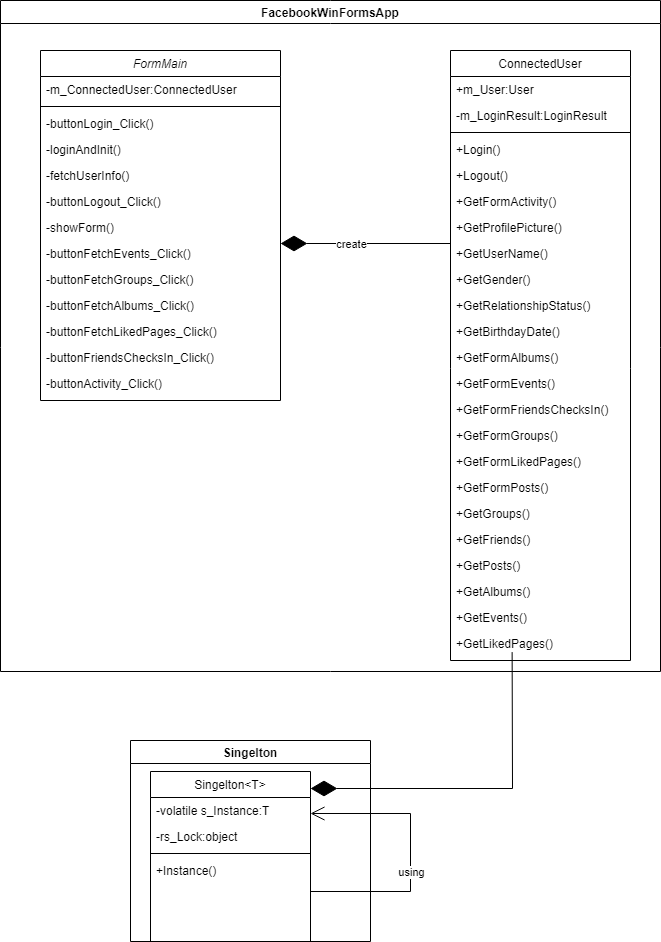
* אופן המימוש:

מימשנו את המחלקה הגנרית Singelton<T> והחזקנו ומופע של Singelton<ConnectedUser> במחלקה MainForm שהיא הנקודת כניסה לאפליקציה.

למחלקה ConnectedUser יש private constructor ובכך נמנע מליצור מופע נוסף שלה מבלי לבקש Instance מהמחלקה Singelton הגנרית.

המחלקה FormMain מבקשת מופע סינגלטוני של המחלקה ConnectedUser .

* Sequence Diagram
* Class Diagram



* המקביל ל"Client" הוא "FormMain".
* המקביל ל"Singleton" הוא "Singleton<ConnectedUser>".

### 

### תבנית מס' 5 – Facade

* סיבת הבחירה / שימוש בתבנית:

1. המערכת בui הממשמשת את אפליקציה זו מורכבת מכמה מחלקות. בכדי לפשט את העבודה בה יצרנו ממשק יחיד העובד עם כמה ממשקים שונים.
2. בכדי למנוע שימוש לא נכון במחלקות השונות הגדרנו את ממשק זה המונע מהמשתמש בו סרבול וטעויות.
3. שימוש בfacade תורם לשימוש חוזר ותחזוקתיות בכך שהfacade חושף ממשק משתמש ומאפשר שינוי/הוספה של רכיבים לממשקים השונים.

* אופן המימוש:

המחלקה ConnectedUser מהווה את הfacade באפליקציה שלנו.

המחלקה FormMian מהווה את הclient, בנוסף לשאר הForms באפליקציה שהם גם מהווים את הclient.

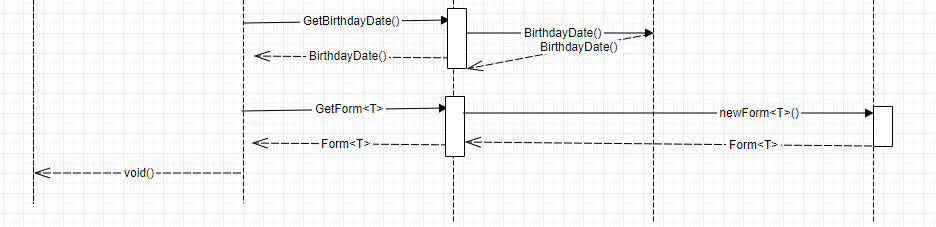
החלטנו לממש את ConnectedUser כfacade לא אטום מכיוון שרצינו לאפשר למשתמש גישה למידעים ספציפיים. בנוסף יש מתודת GetForm לכל שדה של המשתמש.

* Sequence Diagram

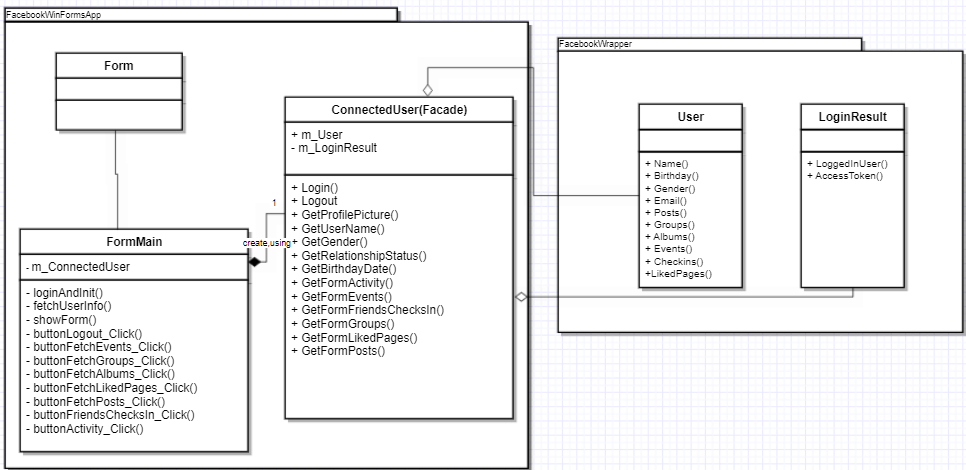
נציג את הדיאגרמה במצב שבו הclient הוא FormMain. וForm<T> הוא אחד מהForms הבאים: FormFriendsChecksIn, FormEvents, FormAlbums, FormLikedPages, FormGroups, FormActivity, FormPosts.

תמונה שמכילה טקסט, מפה, מקורה, לבן

התיאור נוצר באופן אוטומטי



* Class Diagram



המקביל ל”Client” הוא ה”FormMain”.

המקביל ל”Façade” הוא “Connected User”.

המקביל ל”subSystemA” הוא “User”.

המקביל ל”subsystemB” הוא “LoginResult”.

### תבנית מס' 6 – Static Factory Class

* סיבת הבחירה / שימוש בתבנית:

קיימת לנו בתוכנית משפחה פולימורפית היורשת מForm, במקום שהclient ייצר בעצמו את האובייקטים, יצרנו מחלקה סטטית נפרדת אשר מקבלת אובייקט ולפי סוגו יודעת להחליט איזה אובייקט מסוג Form לייצר, ובכך אנו מאפשרים אופציה לשינויים עתידיים/הוספת רכיבים במערכת מבלי לבצע שינויים בclient כלל, אלא רק ע"י שינוי הFormFacrtory.

* אופן המימוש:

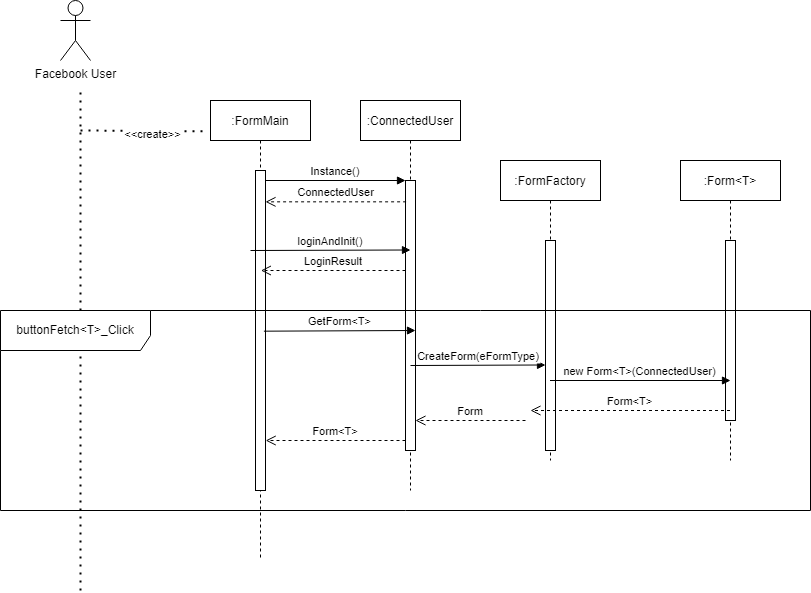
Client – ConnectedUser.cs - שימוש בfactory במטודות GetForm

Base interface – Form

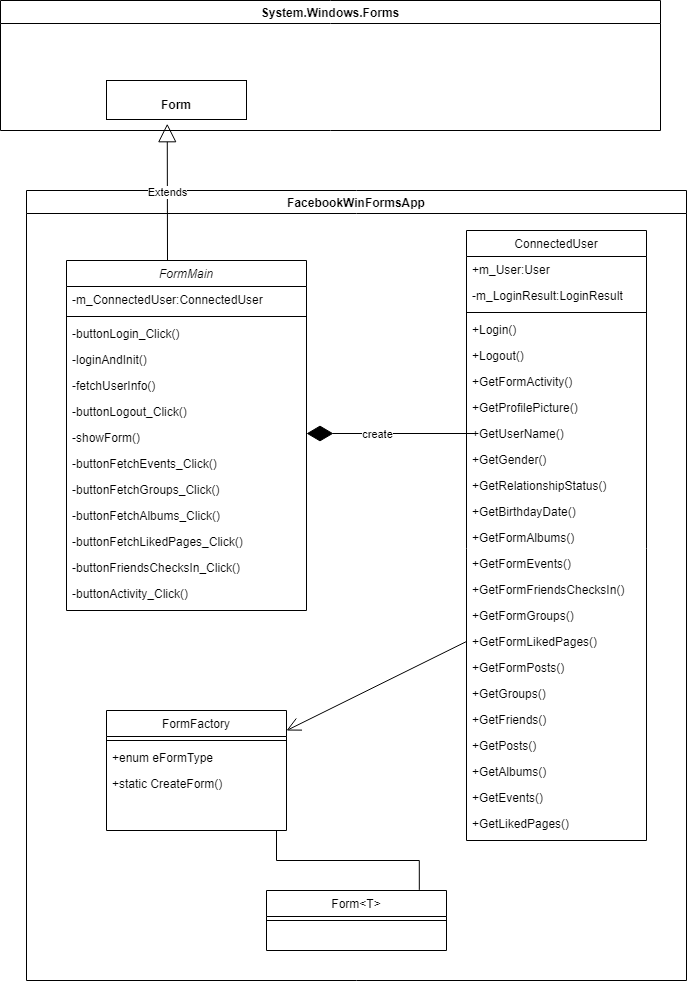
Factory - FormFactory.csבמטודה CreateForm

Products – FormGroups, FormFriendsChecksIn, FormEvents, FormAlbums, FormActivity, FormLikedPages, FormPosts.

* Sequence Diagram



* Class Diagram



**עבודה אסינכרונית:**

ממשנו עבודה אסינכונית במתודה Load של הטפסים:

FormPosts, FormLikedPages,FormGroups,FormFriendsCheksIn,FormEvents

כל פעולות הfetch של הנתונים הרלוונטים בטפסים אלה מתבצעות בThread נפרד ובכך טעינת המידע מתבצעת מהר יותר וה-UI נשאר אינטרקטיבי למשתמש.

**עבודה עם Data bilding:**

ביצענו עבודה עם Data bilding בטופס FormAlbums כך שהאלבומים של המשתמש

מוזנים לתוך ה albumBindingSource, כאשר הlistBox של הFormAlbums ,הTextbox שמציג את שם האלבום והpictureBox שמציג את תמונת האלבום מאזינים לו, כך ניתן לשנות את שם האלבום דרך ה- textBox.