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

1. אלבומים מיון והורדה:  
   המערכת מאפשרת למשתמש לצפות באלבומים שלו ולמיין אותם על בסיס הקטגוריות הבאות:

* החדש ביותר
* ישן ביותר
* גודל (מקטן לגדול)
* גודל (מגדול לקטן)

1. סטטיסטיקות פוסטים

המערכת מאפשר למשתמש לצפות בגרף סטטיסטיקות לפוסטים שהעלה בפילוח על פי זמן.

המשתמש יכול לבחור להציג גרף על בסיס חודשים או שנים ולצפות בכמות הפוסטים שהעלה בזמן שנבחר.

### תבנית מס' 1 – Singletone

* סיבת הבחירה / שימוש בתבנית:

השתמשנו בתבנית ה Singleton כדי לממש את מחלקת FacebookManager במטרה היא לוודא שיש רק מופע אחד של התחברות לפייסבוק בכל רגע נתון. פעולה זו מבטיחה שיהיה רק משתמש אחד מחובר לפייסבוק בכל הזדמנות, ומונעת את האפשרות שמספר משתמשים יתחברו בו זמנית. בנוסף, השימוש בתבנית Singleton מאפשרת השגת התחברות יחידה למערכת ושימוש בה במחלקות אחרות בפרויקט, ומפטיר מהן מהצורך לבצע התחברות מחדש.

* אופן המימוש:

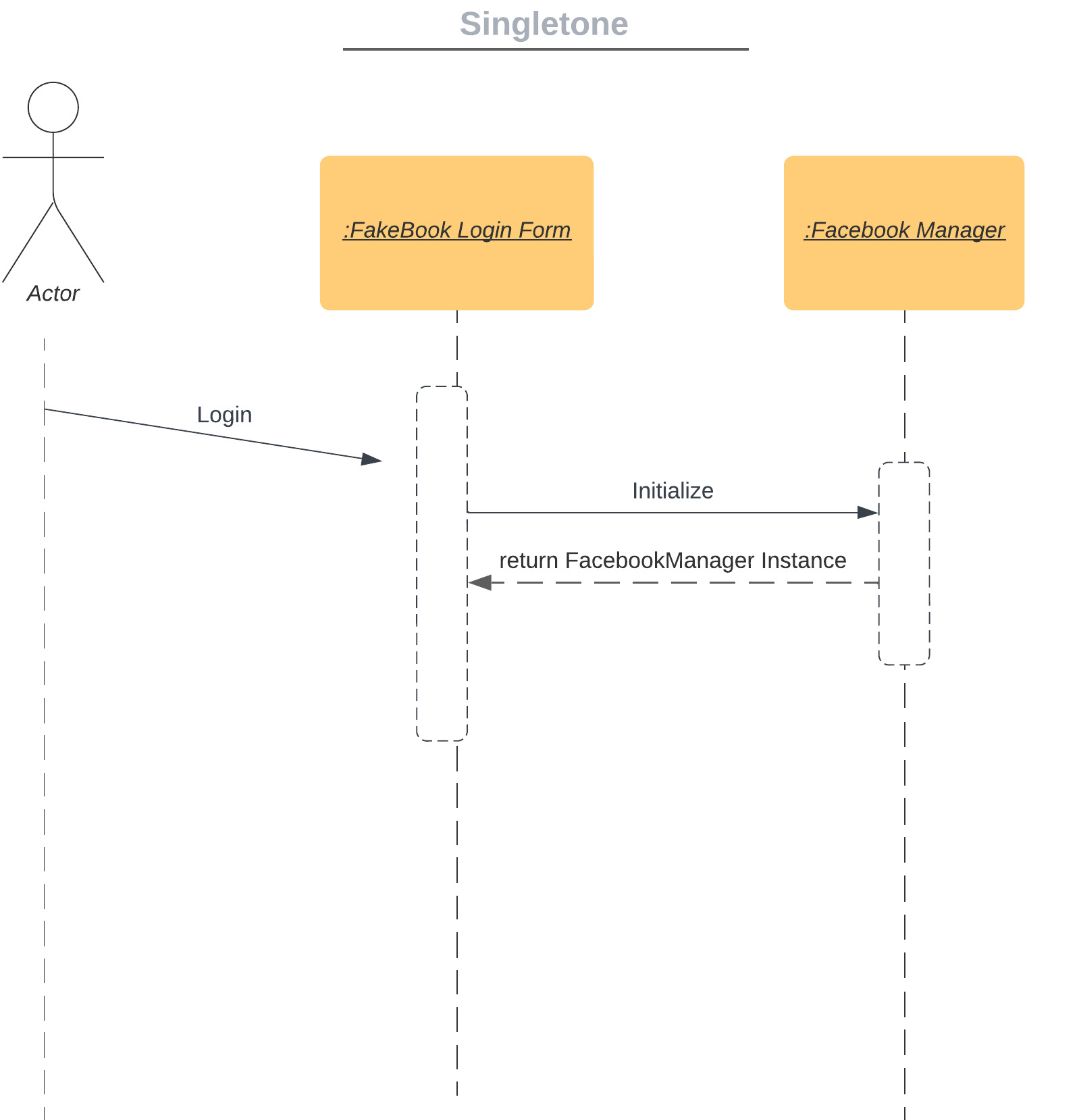
מחלקת FacebookManager הוגדרה כSingletone באופן הבא:  
1. Private constructor

2. public static void Initialize(FacebookWrapper.LoginResult loginResult) – מתודה אשר מאפשרת יצירת מופע ראשוני לאוביקט והתחברות של המשתמש לפייסבוק על ידי קריאה לprivate constructor. המתודה מאפשר ליצור אוביקט בפעם אחת וממומשת באופן thread-safe.

3. public static FacebookManager Instance – Property

מאפשר שימוש באוביקט לאחר יצירתו באופן thread-safe.

* Sequence Diagram



* Class Diagram

### תמונה שמכילה טקסט, תרשים, צילום מסך, מקביל התיאור נוצר באופן אוטומטי

### תבנית מס' 2 – Builder

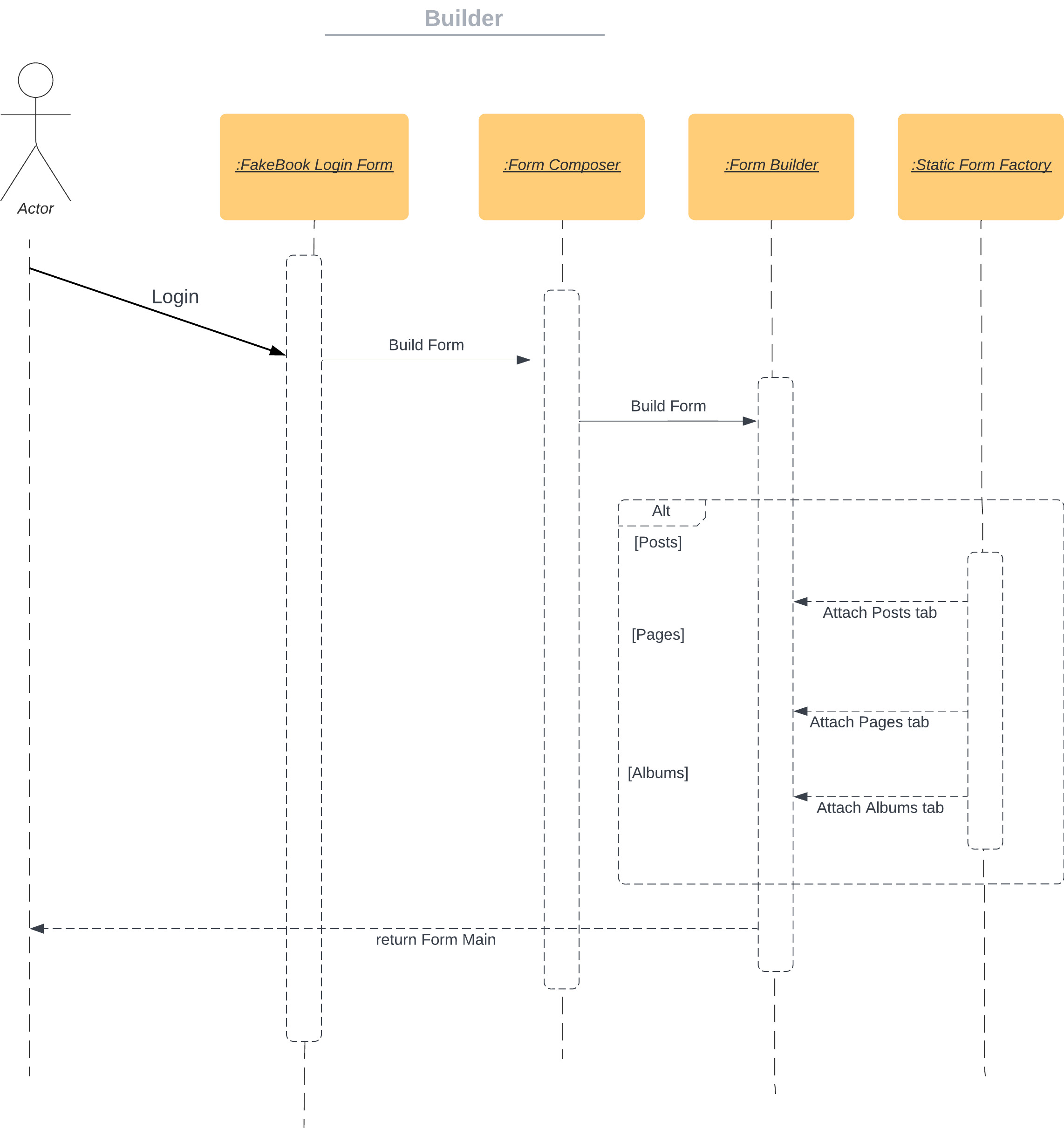
* סיבת הבחירה / שימוש בתבנית:

בחרנו לממש את התבנית הזו כדי לאפשר למשתמשים להגדיר את האובייקטים שהם רוצים לראות במערכת לאחר הרשמתם לאפליקציה. המשתמשים יהיו יכולים לבחור אילו פריטים מסוימים יהיו חשופים להם וממשק המשתמש ייוועץ בהתאם להעדפותיהם. כך יתווספו פחות הזדמנויות לעומס על השרת ונמנע יצירת אובייקטים שאין בהם באמת צורך.

* אופן המימוש:
* FakeBookLoginForm – הטופס הראשוני שנוצר למשתמש ושם הוא בוחר באילו אויבקטים ירצה לצפות מפייסבוק
* FormComposer – המחלקה מקבלת את בחירות המשתמש ומייצרת טופס באמצעותן ובאמצעות מחלקת FormBuilder.
* FormBuilder - המחלקה מייצרת את הטופס שיוצג למשתמש לאחר התחברותו.
* יצרנו 3 טפסים לכל אויביקט : PagesForm, AlbumsForm, PostsForm

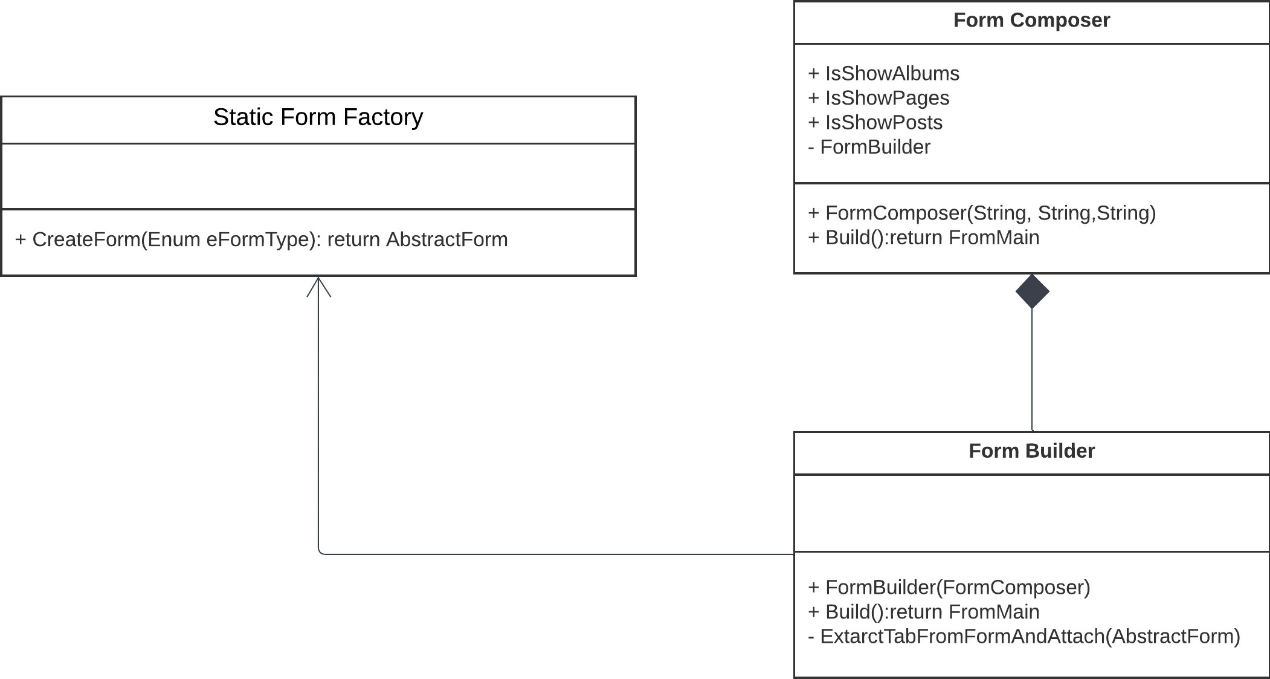
מכל טופס שהמשתמש בחר להציג הוספנו את הtab שלו לתוך הטופס הראשי (FormMain) ובכך נוצר הממשק של הפייסבוק למשתמש.

* Sequence Diagram



* Class Diagram

המחלקה Form Builder מייצרת את הטפסים ע"י שימוש במחלקת static Form Factory



### תבנית מס' 3 – Static Factory

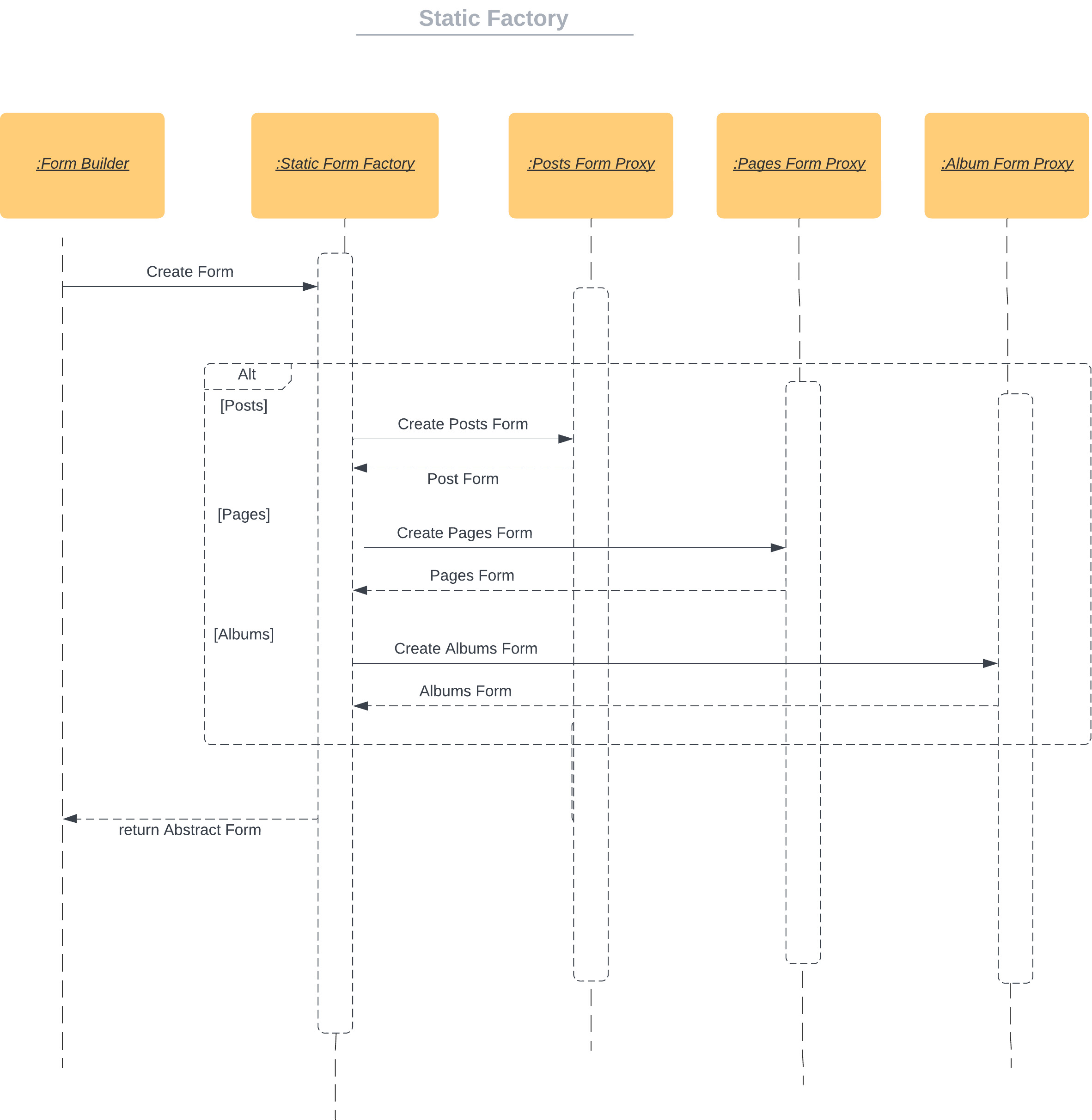
* סיבת הבחירה / שימוש בתבנית:

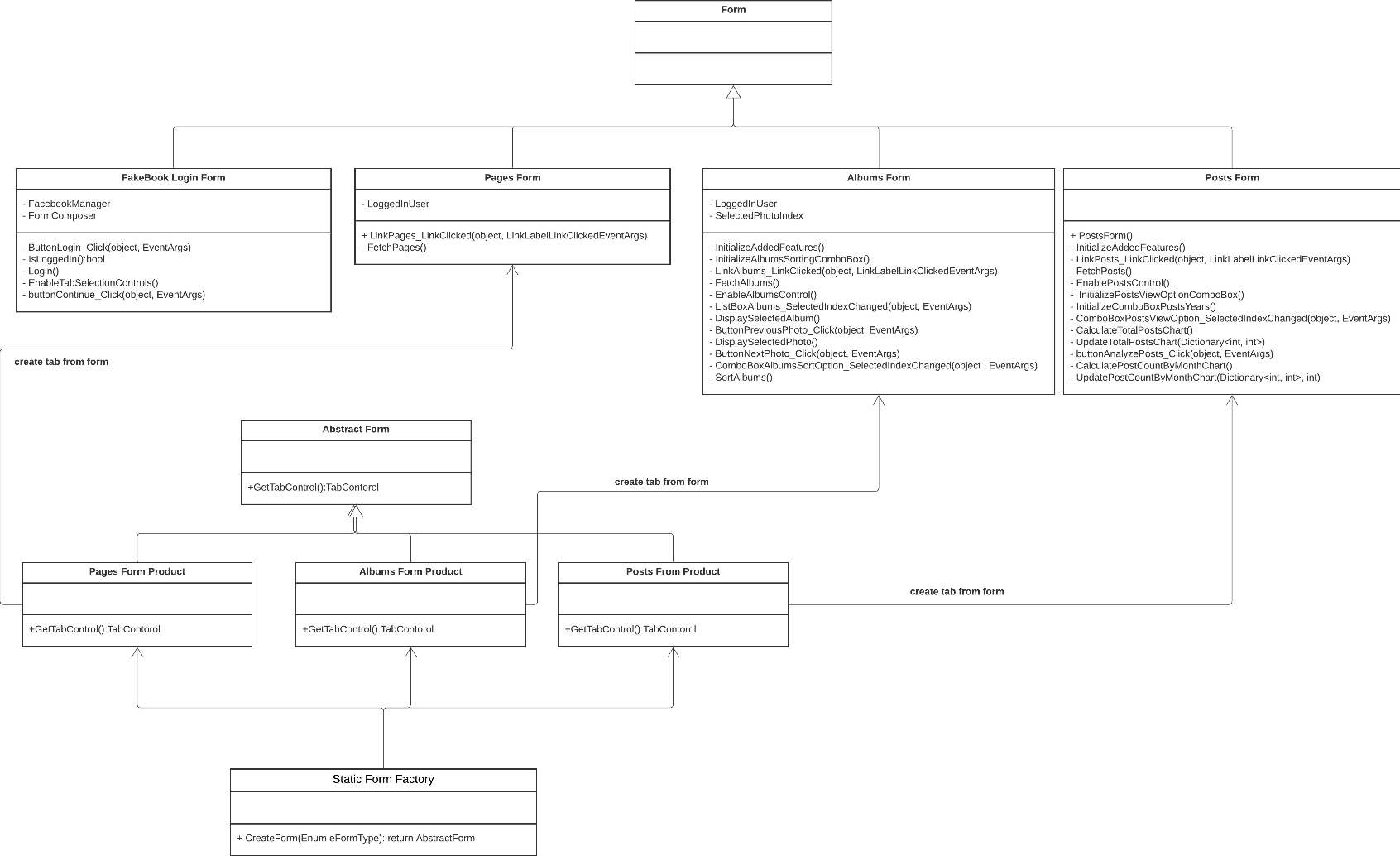
תבנית זו מאפשרת יצירת אוביקטים פולימורפים באמצעות מחלקה סטטית בשם StaticFormFactory. תבנית מונעת שכפול קוד ומאפשרת יצירת אוביקטים עתידיים בצורה נוחה יותר עם הפרדה לבין שכבת הלוגיקה לבין יצירת האוביקטים.

* אופן המימוש:
* StaticFormFactory – מחלקה זו מייצרת את האוביקט לפי סוג המוצר (אלבומים, פוסטים או עמודים)
* AbstractForm – מחלקה זו מהווה הבסיס הפולימורפי לטפסים שנוצרים באותו האופן מטיפוסים שונים
* AlbumsFormProduct – מחלקה זו מייצרת טופס מסוג אלבום ומחזירה טאב שלו לMainForm
* PagesFormProduct – מחלקה זו מייצרת טופס מסוג עמודים ומחזירה טאב שלו לMainForm
* PostsFormProduct - מחלקה זו מייצרת טופס מסוג פוסטים ומחזירה טאב שלו לMainForm
* יצרנו 3 טפסים לכל אויביקט : PagesForm, AlbumsForm, PostsForm

מכל טופס שהמשתמש בחר להציג.

* Sequence Diagram



* Class Diagram

המחלקה Static Form Factory יוצרת טופס לפי הקלט המבוקש ע"י שימוש במשפחה הפולימורפית Abstract Form

**Data bining**

בחרנו לממש זאת בכל גישה לשרת של פייסבוק לקבלת נתונים במתודות במחלקות הנ"ל:

1. פוסטים – PostsForm::FetchPosts
2. אלבומים – AlbumsForm::FetchAlbums
3. עמודים – PagesForm::FetchPages

**Threads**

בחרנו לממש threasds בכל גישה לשרת של פייסבוק על מנת לייעל את זמן הריצה בפנייה לשרת.

1. פוסטים – PostsForm::FetchPosts
2. אלבומים – AlbumsForm::FetchAlbums
3. עמודים – PagesForm::FetchPages