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

1. **תזמון פוסטים לפי קבוצה מסויימת:**

פיצ'ר זה מאפשר למשתמש לכתוב פוסט עתידי בקבוצה מסויימת.

המשתמש בוחר את תוכן הפוסט ובעוד כמה זמן הוא רוצה שהפוסט יתפרסם.

כאשר הפעולה מתבצעת המשתמש מקבל התראה של הצלחה או כישלון (במידה ומשהו השתבש).

הערה: כיום פייסבוק מאפשרת לפרסם פוסטים רק בקבוצות אשר המשתמש הוא המנהל שלהם.

ניתן למצוא בקוד תחת המחלקות :

FormSchedulePost.cs – (מחלקת טופס)

ScheduledPost.cs – (מחלקה לוגית)

1. **קבלת פרטים על תמונות:**

פיצ'ר זה מאפשר למשתמש לקבל מידע אודות תמונות המידע כולל:

* מי החבר שהגיב לו הכי הרבה פעמים בכל תמונותיו.
* מי החבר שעשה לו הכי הרבה לייקים על תמונותיו.
* כמה סה"כ לייקים יש למשתמש על כל התמונות שלו.
* כמה סה"כ תגובות יש למשתמש על כל התמונות שלו.
* איזה מהתמונות של המשתמש קיבלה הכי הרבה לייקים.
* איזה מהתמונות של המשתמש קיבלה הכי תגובות לייקים.

ניתן למצוא בקוד תחת המחלקות :

FormPhotosDetails.cs – (מחלקת טופס)

UserPhotoDetails.cs – (מחלקה לוגית)

### תבנית מס' 1 – Static factory class

* **סיבת הבחירה / שימוש בתבנית:**

המערכת שפיתחנו כוללת מספר מסכים ( מסך לכל פיצ'ר).

הסיבה שלימוש בתבנית זו היא שאנו רוצים למנוע מהקליינט את הצורך לאתחל את המסכים שברצונו להפעיל.

בנוסף לכך במידנה ויוספו פיצארים חדשים הכוללים מסכים נלווים חדשים כל אשר ידרש לבצע הוא הוספה של המסך החדש ל"מפעל".

* **"שחקנים" בתבנית:**

קליינט – FormMain.

המפעל – ScreenFactory

הממשק - IScreen

המסכים שממשים את הממשק – FormPhotosDetails & FormSchedulePost

* **אופן המימוש:**

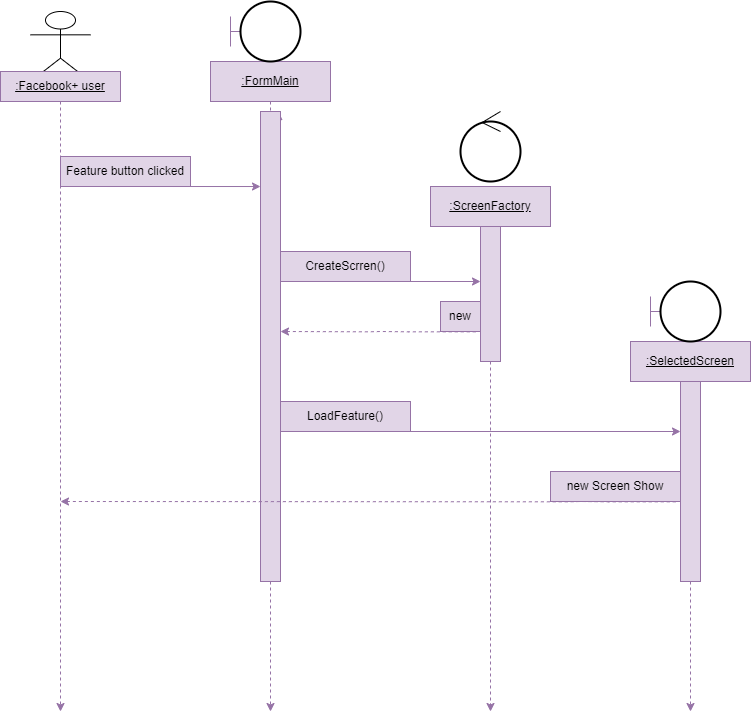
יצרנו מחלקה סטטית בשם ScreenFactory בעלת מתודה אחת שמטרתה להחזיר מסך חדש לפי בקשת המשתמש. ( המתודה מחזירה אובייקט מסוג IScreen ).

יצרנו ממשק בשם IScreen בעל מתודה אחת הנקראית StartFeature כאשר כל מסך שממש את הממשק מבצע בעזרתה טעינה לאובקייטים הנחוצים לאותו פיצאר שנבחר בנוסף.

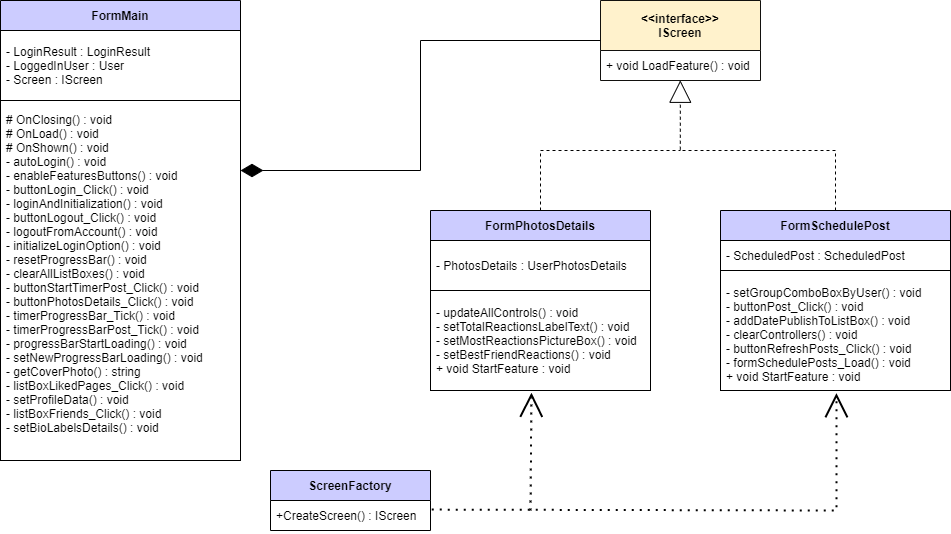
ניתן למצוא את המחלקות החדשות ב:

1. B22 Ex01 MatanBarEliyahu 318324258 DanielGelfand 209380773\FacebookWinFormsApp\IScreen.cs.
2. B22 Ex01 MatanBarEliyahu 318324258 DanielGelfand 209380773\FacebookWinFormsApp\ScreenFactory.cs

* **Sequence Diagram:**



* **Class Diagram:**



### תבנית מס' 2 – Facade (שקוף)

* **סיבת הבחירה / שימוש בתבנית:**

לאפשר לקליינט לבצע שימוש פשוט ונוח בחלק הלוגי של המערכת ולמנוע ממנו לפנות ישירות למחלוקות הלוגיות. כלומר איפשרנו לקליינט ללבצע קריאה פשוטה על מנת להפעיל את הפיצ'רים המורכבים.

* **"שחקנים" בתבנית:**

קליינט – , FormSchedularPost, FormPhotoDetails FormMain.

ה-Facade – FacadeLogicManager

ה"ספרייה" – SchedularPost, PhotoDetails

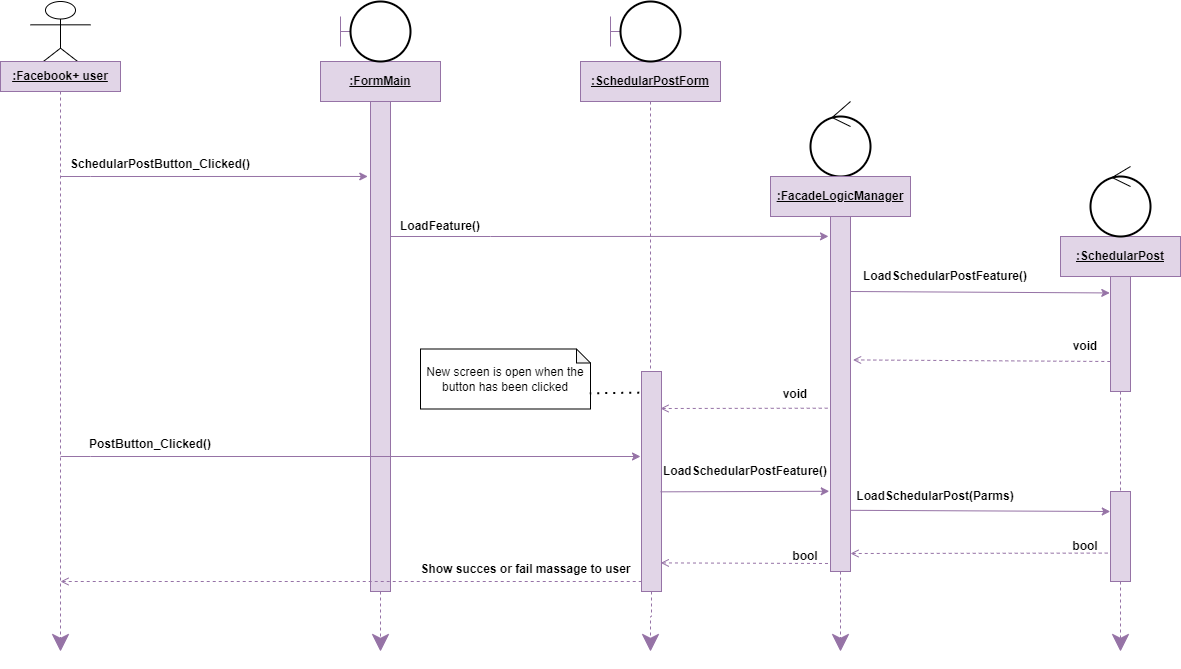
* **אופן המימוש:**

הפרדנו את הסולושיין לשני פרוייקטים - לוגי וממשק משתמש כאשר כל המחלקות בחלק הלוגי חלק מהמחלקות מוסתרות באמצעות internal בזמן שמחלקת ה-Facade מחזיקה את יתר המחלקות שנרצה לבצע בהם שימוש, מחלקת הFacade חושפת מטודות Public שמפשטות את ביצוע הפעולות המורכבות של החלק הלוגי במערכת .

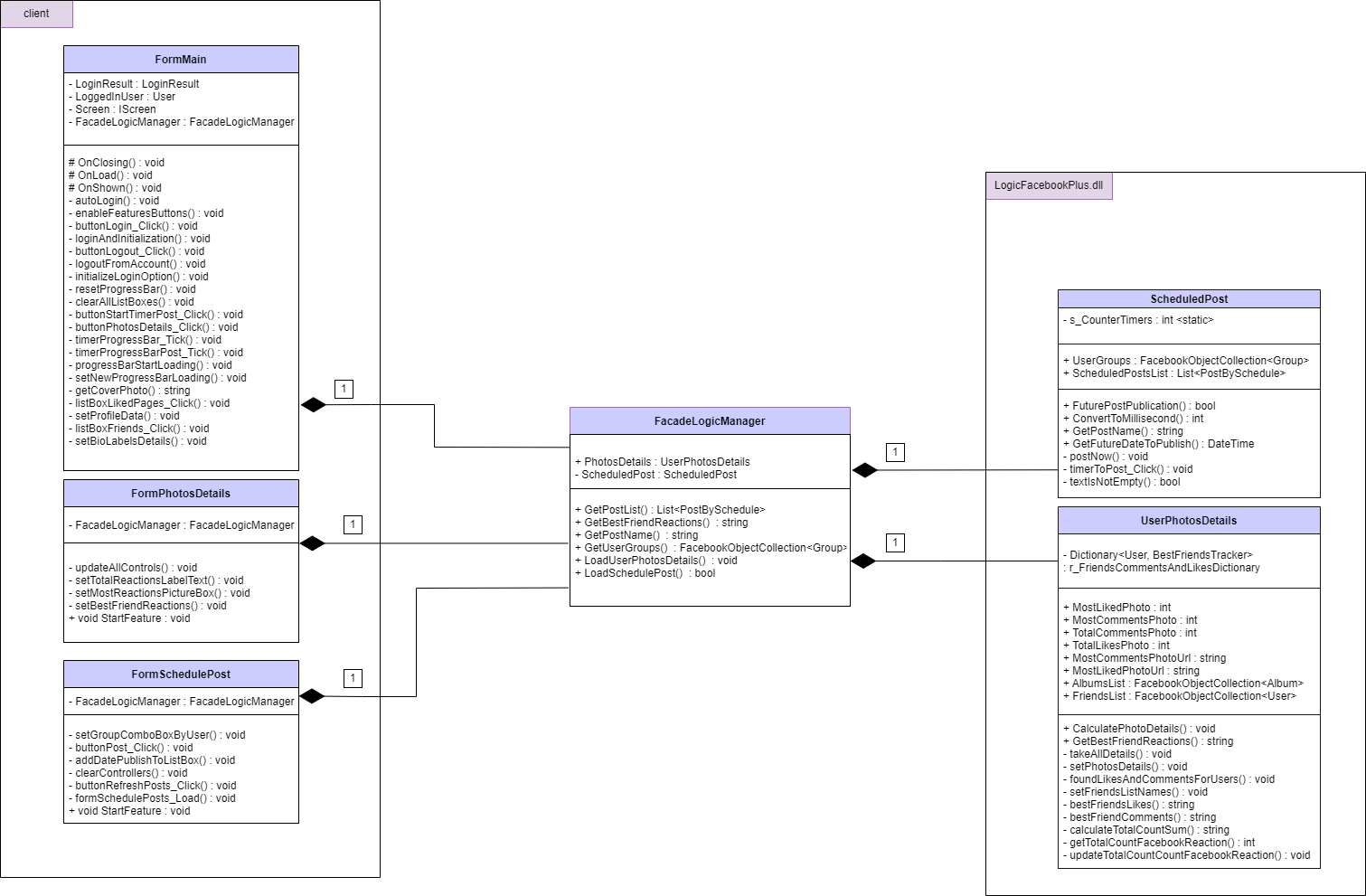
ניתן למצוא בקוד:

B22 Ex01 MatanBarEliyahu 318324258 DanielGelfand 209380773\LogicFacebookPlus\FacadeLogicManager.cs

* **: Sequence Diagram**



* **Class Diagram:**



### תבנית מס' 3 – Class Adapter

* **סיבת הבחירה / שימוש בתבנית:**

מטרתנו הייתה ליצור אחידות עבור פעולת ההתחברות של המשתמש ל – API . באמצעות השימוש בתבנית ה"מתאם" ניתן לבצע את פעולת ההתחברות בעזרת מתודת לוגין פשוטה שלא מקבלת פרמטרים.

כתוצא מפעולה זו במידה שהמערכת שפיתחנו תתמוך בAPI נוספים (כמו Instagram or Twiter ( נוכל לבצע זאת מבלי לשנות את צורת ההתחברות בקליינט לדוגמא על ידי הוספת InstagramAdapter.

* **"שחקנים" בתבנית:**

קליינט – FormMain.

ה"מתאם" שמממש את הממשק – FacebookAdapter

הממשק - ILoginable

ה-Adaptee - FacebookService

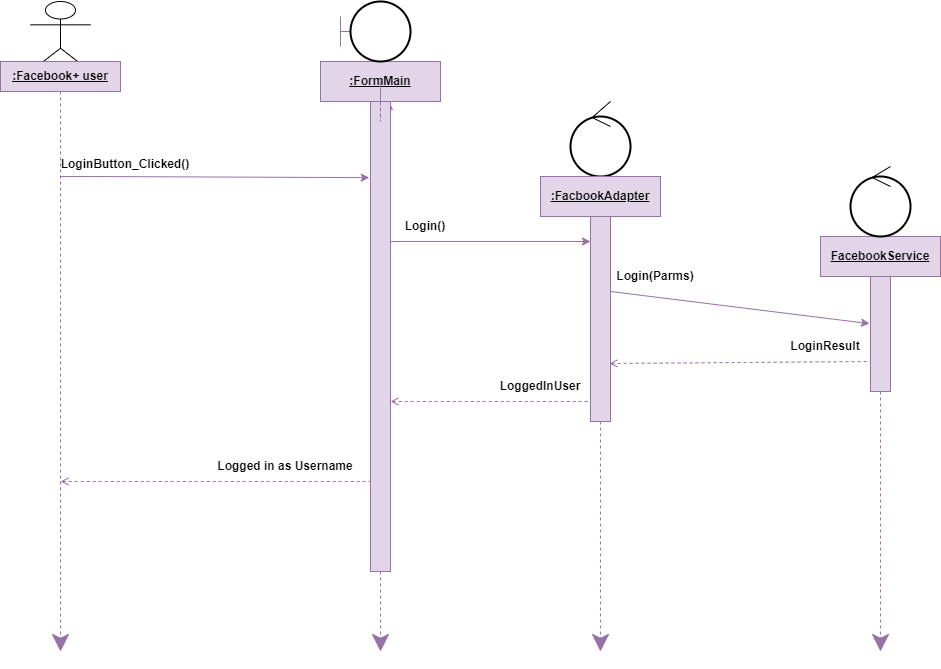
* **אופן המימוש:**

יצרנו אינטרפייס חדש שמי שמממש אותו צריך לממש מטודות התחברות ללא פרמטרים . בנוסף יצרנו קלאס שממש את האינטר פייס ויורש מ-פייסבוק סרביס שבעזרתו מבצעים את ההתחברות.

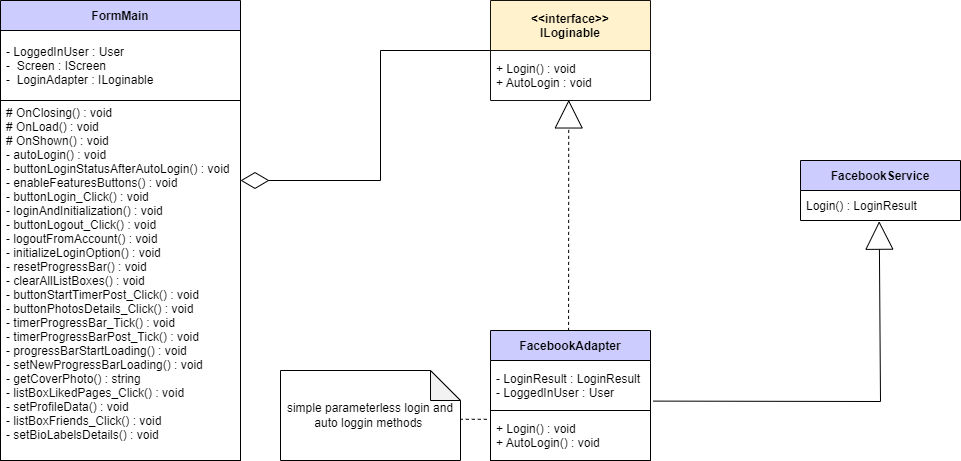
ניתן למצוא בקוד:

1. Ex01 MatanBarEliyahu 318324258 DanielGelfand 209380773\FacebookWinFormsApp\ILoginable.cs
2. B22 Ex01 MatanBarEliyahu 318324258 DanielGelfand 209380773\FacebookWinFormsApp\FacebookAdapter.cs

* **Sequence Diagram:**



* **Class Diagram**:



### שימוש ב-Threads & DataBinding:

**שימוש בThreads:**

ביצענו שימוש ב"תהלכים" במספר מקומות בקוד. השימוש בוצע בכדי לטעון מספר פיצ'רים המוצגים למשתמש במקביל (וזאת בכדי שהמשתמש יוכל להמשיך לעבוד עם המערכת ללא תקיעות).

התחברות למערכת באמצעותLogin או AutoLogin מתבצעת ע"י קריאה לThread נפרד. במהלך ההתחברות לאפליקציה מתבצעת טעינה של רשימת החברים, הדפים האהובים על המשתמש, לקיחת תמונת הרקע, לקיחת תמונת הנושא של המשתמש וטעינה של האוביקט FacadeLogicManager האחראי על התיווך בין ממשק המשתמש ללוגיקה.

ניתן למצוא את השימוש בThreads במחלקה FormMain בשורות 67 ו- 105.

**שימוש ב-DataBinding:**

ביצענו שימוש בפיצ'ר בשני מקומות

הראשון הוא בין ListBox לבין Page, בעזרת קישור המידע בינהם הצגנו למשתמש את שמות דפיו האוהבים, את תמונת הדף הנבחר ולצד התמונה קישור לדף באתר פייסבוק.

השני הוא בין ListBox לבין User, בעזרת קישור המידע בינהם הצגנו למשתמש את שמות חבריו בפייסבוק, את תמונת החבר לפי בחירת השתמש ולצד התמונה קישור לעמוד החבר באתר פייסבוק.