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

* מיון הדפים שהמשתמש עשה להם לייק לפי הפופולריות שלהם ומשחק ניחושים על מס' הלייקים של כל דף.
* בדיקת מזג האוויר במיקומי אירועים שהמשתמש עתיד ללכת אליהם בטווח של חמישה ימים.

### תבנית מס' 1 – Proxy

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית Proxy מכיוון שרצינו לאפשר עריכת נתוני פוסטים בצורה מותאמת בממשק המשתמש, תוך שמירה על הנתונים המקוריים של הפוסט (Post) כפי שסופקו על ידי הFacebookAPIWrapper.

הבעיה העיקרית הייתה ש- Post הוא רכיב Legacy שאינו מאפשר עריכה ישירה, ולכן נדרשה עטיפה שתתמוך בתכונות כמו Data Binding ותאפשר עדכון שינויים ל-Description באופן דינמי.

* אופן המימוש:

יצרנו מחלקה בשם PostProxy, שמבצעת קומפוזיציה עם מחלקת Post המקורית ומוסיפה תמיכה ב- INotifyPropertyChanged לצורך Data Binding.

המחלקה מאפשרת לממשק המשתמש לערוך את התיאור (Description) של הפוסט מבלי לשנות ישירות את האובייקט המקורי. מימוש המחלקה PostProxy נמצא תחת הפרויקט BasicFacebookFeatures, והשימוש בה מתבצע ברכיבי FeedItemControl לצורך תצוגה ועריכה של פוסטים ב-News Feed.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 2 – Singleton

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית Singleton כדי להבטיח שלמערכת יהיה מופע יחיד בלבד של המשתמש המחובר (LoggedInUser).

הדרישה נבעה מכך שרק משתמש אחד יכול להיות מחובר בזמן נתון, ושהמידע על המשתמש יהיה זמין באופן גלובלי למערכת.

* אופן המימוש:

יצרנו מחלקה בשם Singleton, שמממשת מנגנון Singleton גנרי כפי שנלמד בהרצאה.

יצרנו מחלקה נוספת בשם LoggedInUserSingleton עם:

* Ctor פרטי כדי למנוע יצירת מופעים מחוץ למחלקה.
* Property סטטי בשם Instance המחזיר את המופע היחיד.
* Property של LoginResult לאחסון נתוני המשתמש.

המחלקה ממומשת בפרויקט FacebookWinformsApp.

במחלקה FormMain, הLogin נשמר באמצעות הSingleton, ומשתמשים במידע של המשתמש המחובר במתודות נוספות.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### תבנית מס' 3 – Builder

* סיבת הבחירה / שימוש בתבנית:

בחרנו בתבנית Builder מכיוון שבניית קובץ ה-HTML להצגת נתוני מזג האוויר כוללת מספר שלבים מורכבים, כמו עיבוד נתונים שונים (תאריך, עיר, טמפרטורה, לחות וכדומה) והמרתם למבנה HTML אחיד. תבנית Builder מאפשרת לנו להפריד בין תהליך הבנייה לבין המבנה הסופי, לשפר את הקריאות והגמישות של הקוד.

* אופן המימוש:

יצרנו מחלקה בשם WeatherReportBuilder תחת פרויקט FacebookAppLogic, האחראית על בניית דוח מזג האוויר בשלבים.

כל שלב בנייה מתבצע באמצעות מתודות Set שמחזירות את ה- Builder עצמו, כדי לאפשר שרשור קריאות.

לבסוף, מתודת Build מחזירה את מחרוזת ה- HTML המלאה.

יצרנו מחלקה בשם WeatherReportComposer שתפקידה לנהל את תהליך הבנייה:

* היא משתמשת ב- WeatherReportBuilder כדי לקבל את המידע הנדרש.
* היא אחראית על הפעלת ה- Builder לפי סדר השלבים הנדרש.

השימוש בבנאי נמצא במחלקה WeatherCheckWindow בפרויקט FacebookWinformsApp במתודה displayWeatherForEvent שמציגה את הדוח בדפדפן.

* Sequence Diagram

[sequence diagram שמציגה את התהליך שקשור לתבנית והאינטראקציה בין המחלקות]

* Class Diagram

[class diagram שמתארת את המחלקות שלכם שמעורבות בתבנית (תיאור מלא שכולל Properties ו- Methods) והיחסים ביניהם (תאור מלא עבור כל יחס כפי שלמדנו בכיתה). עבור כל מחלקה שלכם, כיתבו מי המקבילה שלה (אם יש כזו) בתבנית (Pattern) שבחרתם

### Two Way Data Binding:

מימשנו את האפשרות לערוך Descripion של פוסט בפיד הראשי ע"י לחיצה כפולה עליו.

לאחר לחיצה כפולה, נפתחת תיבת טקסט במקום הLabel, ובה ניתן לערוך את התוכן של הפוסט וכאשר לוחצים Enter הפוסט מתעדכן.

המימוש נמצא במחלקת FeedItemControl.

הוספנו גם מחלקת PostProxy על מנת לערוך את התיאור שכן Description הוא ReadOnly במחלקת Post והיא Legacy Component.

### עבודה א-סינכרונית עם ממשק משתמש:

בקוד השתמשנו בתכנות א-סינכרוני במקומות הבאים:

1. **התחברות המשתמש** - **(LoginAsync)** מבצע את תהליך ההתחברות ברקע כדי למנוע חסימה של ה UIלאחר ההתחברות ומעדכן את הקומפוננטות הרלוונטיות באמצעות Invoke.
2. **טעינת פיד הפוסטים (loadNewsFeedAsync)** - טוען את הפוסטים ברקע ומוסיף אותם ל-flowLayoutPanelFeed באמצעות Invoke.
3. **טעינת רשימת החברים (loadFriendsAsync)** - טוען את החברים ברקע ומעדכן את listBoxFriends עם השמות והתמונות שלהם.
4. **טעינת אלבומים (loadAlbumsAsync)** - טוען את אלבומי המשתמש ברקע ומוסיף אותם ל-listBoxAlbums.
5. **טעינת דפים שאהבתי (loadPagesAsync)** - טוען את הדפים ברקע ומעדכן את listBoxLikedPages.

בכל המקרים, הפעולות מתבצעות ב-Threads נפרדים כדי להבטיח שהממשק המשתמש יישאר מגיב במהלך פעולות כבדות.