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

* MatchFinder – פיצ'ר זה מקנה למשתמש את האפשרות למצוא זיווג מתאים עפ"י קריטריונים של מין וטווח גילאים, מבין חבריו בפייסבוק.
* LikeSorter – פיצ'ר זה מקנה למשתמש את האפשרות למיין בסדר עולה את רשימת חבריו בפייסבוק לפי כמות הלייקים שניתנו מכל חבר עבור כלל הפוסטים של המשתמש.

### תבנית מס' 1 –Factory Method

* סיבת הבחירה / שימוש בתבנית:

בתרגיל זה הרחבנו את אפליקציית הפיייסבוק שלנו כך שהיא תומכת במספר רב יותר של פיצ'רים, אשר פולימורפיים אחד לשני ומממשים interface משותף.

בשל כך, על מנת להכמיס ולספק ממשק פשוט ליצירת אובייקט קונקרטי שלא ימומש בclient, ראינו לנכון להשתמש בתבנית FactoryMethod.

שכן, תבנית זו מקלה על תחזוקת המערכת בעתיד, ומכמיסה את לוגיקת יצירת האובייקט הקונקרטי כך שהclient משתמש בצורה זהה בכלל האובייקטים במשפחה הפולימורפית.

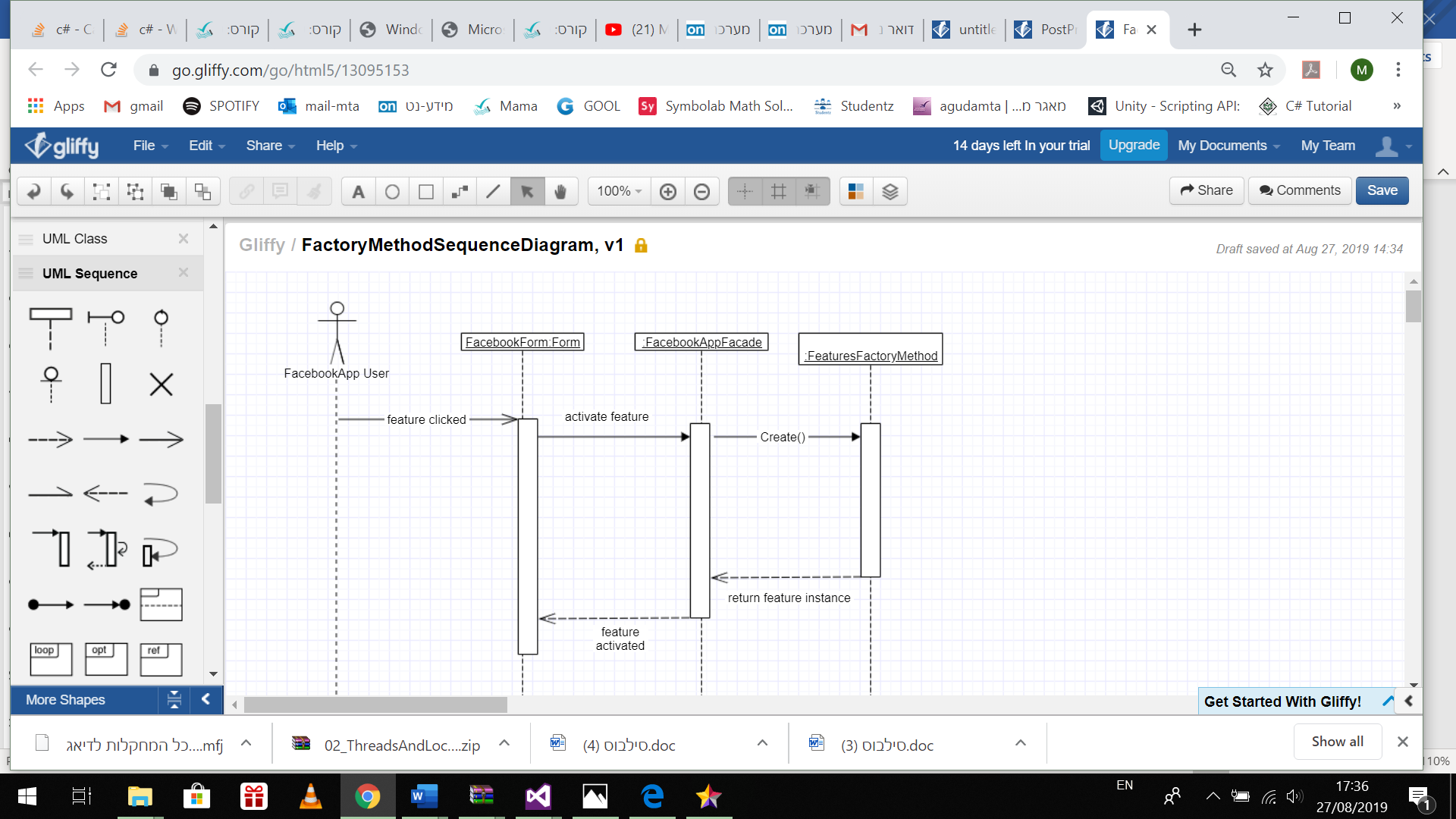
* אופן המימוש:

הגדרנו אב פולימורפי IFeature לפיצ'רים ,matchFinder,LikeSorter ,PostNewStatus שממומש כinterface המכיל מתודה FeatureCliked() אשר ממומשת בצורה שונה עבור כל פיצ'ר אך מופעלת בצורה זהה כאשר מתבצעת הפעלה של פיצ'ר מסויים מהנ"ל בUI.

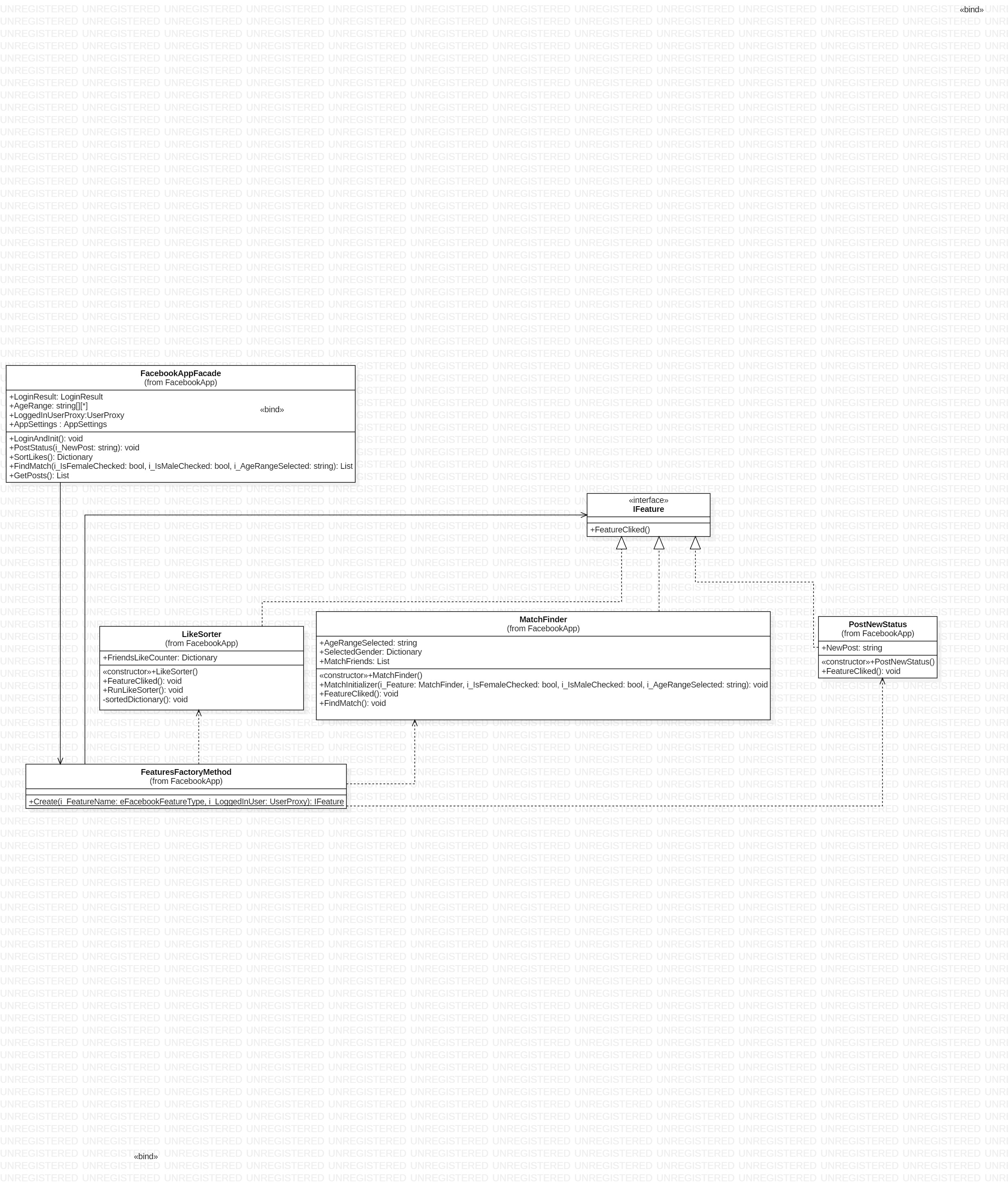
מכיוון שהפעלת הפיצ'רים הנ"ל מתבצעת ע"י לחיצת כפתור בUI, ראינו לנכון לממש אב פולימורפי מסוג זה.

בשל היווצרות של משפחה פולימורפית והרצון ל- reuse ואינקפסולציה, הפרדנו את מלאכת היצירה מהclient (FacebookAppForm) אל מחלקת FeaturesFactoryMethod שבה מומשה מתודת יצירה סטטית ()Create אשר מקבלת את סוג האובייקט הקונקרנטי שנרצה לייצר (eFacebookFeatureType), ותפקידה לייצר מופע שלו ולהחזיר אותו כטיפוס מסוג אובייקט האב(IFeature).

* Sequence Diagram



* Class Diagram



**Factory Method participants (**FeaturesFactoryMethod**):**

**Product:**

IFeature

**Concrete Product:**

PostNewStatus

MatchFinder

LikeSorter

**Creator:**

**ConcreteCreator:**

FeaturesFactoryMethod

### תבנית מס' 2 –Façade

### סיבת הבחירה / שימוש בתבנית:

בתרגיל הקודם מירב מימוש הלוגיקה בוצעה בתוך הclient (facebookAppForm) לכן, רצינו להפריד בין ממשק הUI והלוגיקה, ובנוסף לספק שירות קל ופשוט שיהווה מתווך בין השניים.

לשם פישוט המערכת השתמשנו בתבנית זו בצורה "שקופה" אשר עוזרת בתחזוקה עתידית של המערכת (ע"י עדכון של הfacade במידת הצורך מבלי לעדכן את המשתמשים בו) וכן בטיחות שימוש בclient, אך מאפשרת גם שימוש עצמאי של הלוגיקה המופעלת מאחוריו.

* אופן המימוש:

הגדרנו מחלקת FacebookAppFacade שמקבילה לשכבת הview model אשר מהווה הפרדה בין שכבת הview (UI) לבין שכבת הmodel (app logic) .

הclient מייצר מופע של מחלקה זו, ובאמצעותה חשוף ל "מתודות בסיסיות".

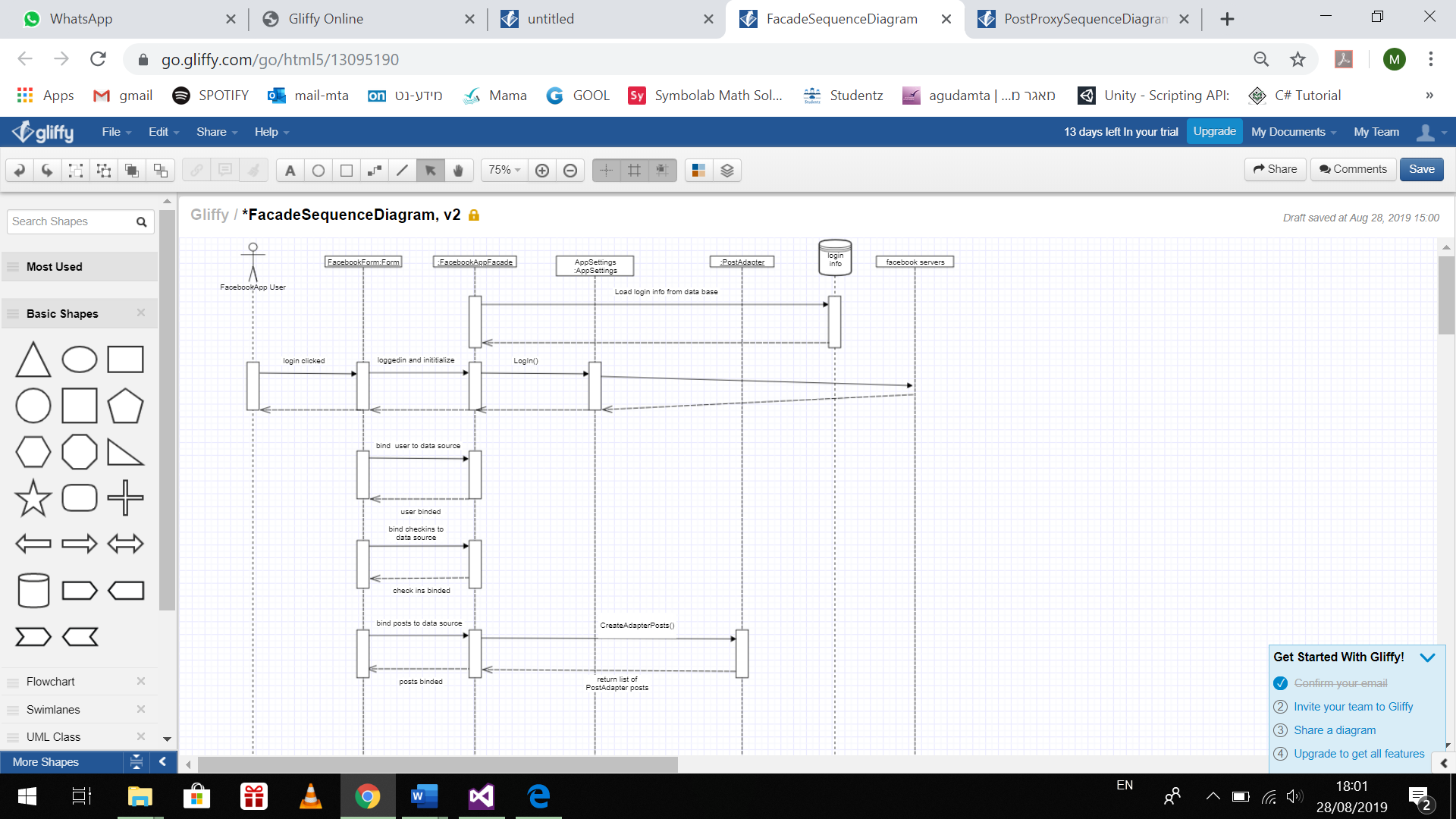
בתוך מתודות אלו בfacade יתבצע קישור וקריאה ללוגיקה הממומשת בשכבת הmodel.

מחלקת הfacade מחזיקה מופעים של userProxy, loginResult וappSettings על מנת לספק נוחות ובטיחות שימוש לclient כך שאחריות ניהול אובייקטים אלה יתבצע באופן מוסתר ומרוכז בשכבת הview Model.

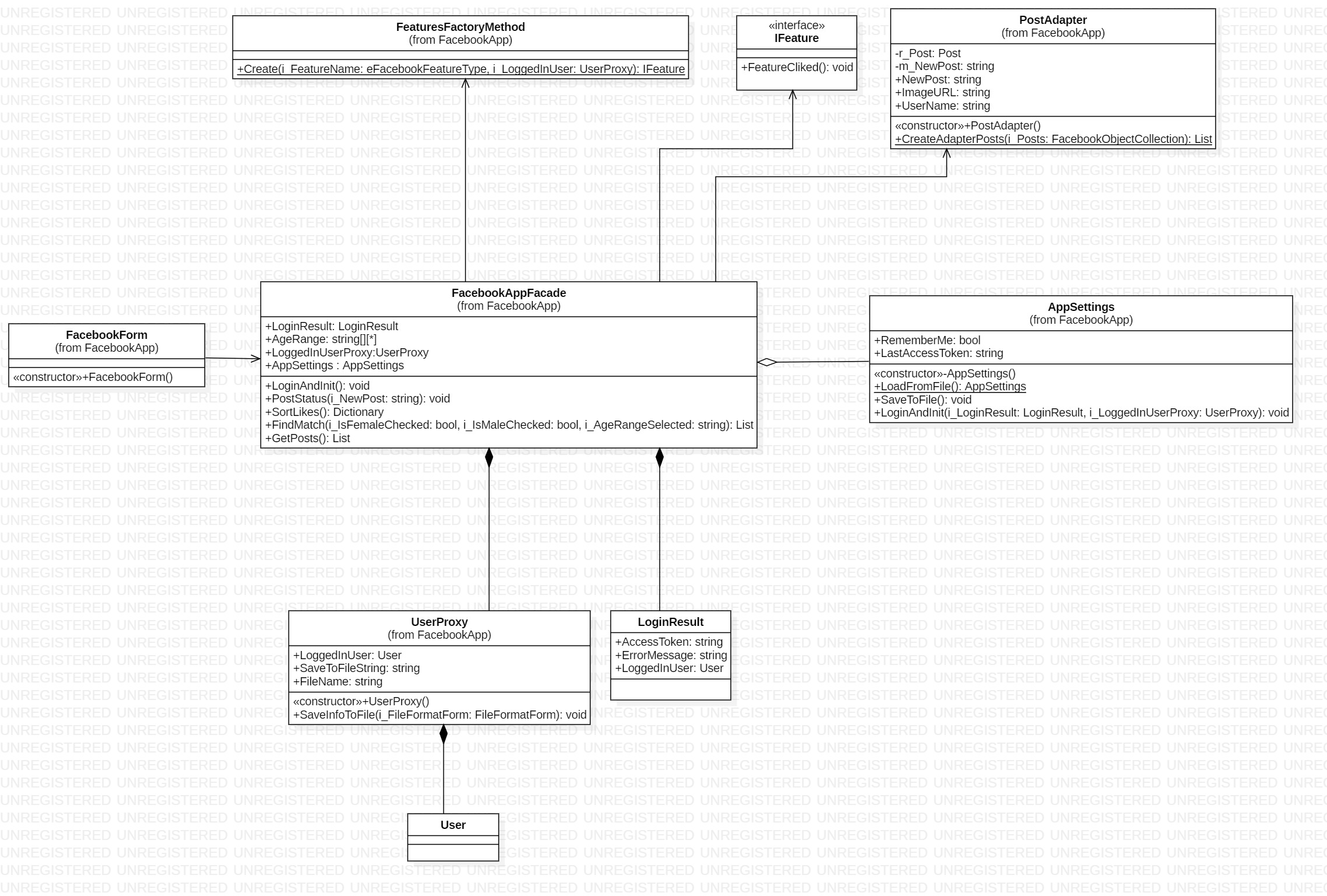
בנוסף, היא משתמשת בשירותיה של מחלקת FeaturesFactoryMethod כך שכל יצירה של אובייקט מסוג IFeature יתבע באופן lazy ,on first demand.

כלל הקריאות המפעילות את הפיצ'רים באפליקצייה עוברות דרך הfacade ומוחזרות דרכו אל הclient.

* Sequence Diagram



* Class Diagram



**Sub-system Classes**:

AppSettings

LoginResult

UserProxy

FeaturesFactoryMethod

PostNewStatus

MatchFinder

LikeSorter

**Façade:**

FacebookAppFacade

**Client:**

facebookAppForm

### תבנית מס' 3 –Adapter

* סיבת הבחירה / שימוש בתבנית:

בתרגיל זה רצינו להוסיף פיצ'ר לאפליקצייה אשר יאפשר עריכה של פוסטים קיימים.

היות והרכיב הקיים Post אשר ממומש בתוך הDllים שקיבלנו, לא מאפשר עריכה מסוג זה וכן הוא מהווה רכיב Legacy שאין לשנותו, ראינו לנכון למממש רכיב adapter אשר יתחזה לרכיב הlegacy המקורי וידאג לממש את לוגיקת הפיצ'ר באופן שבו הclient לא יבחין בשינוי.

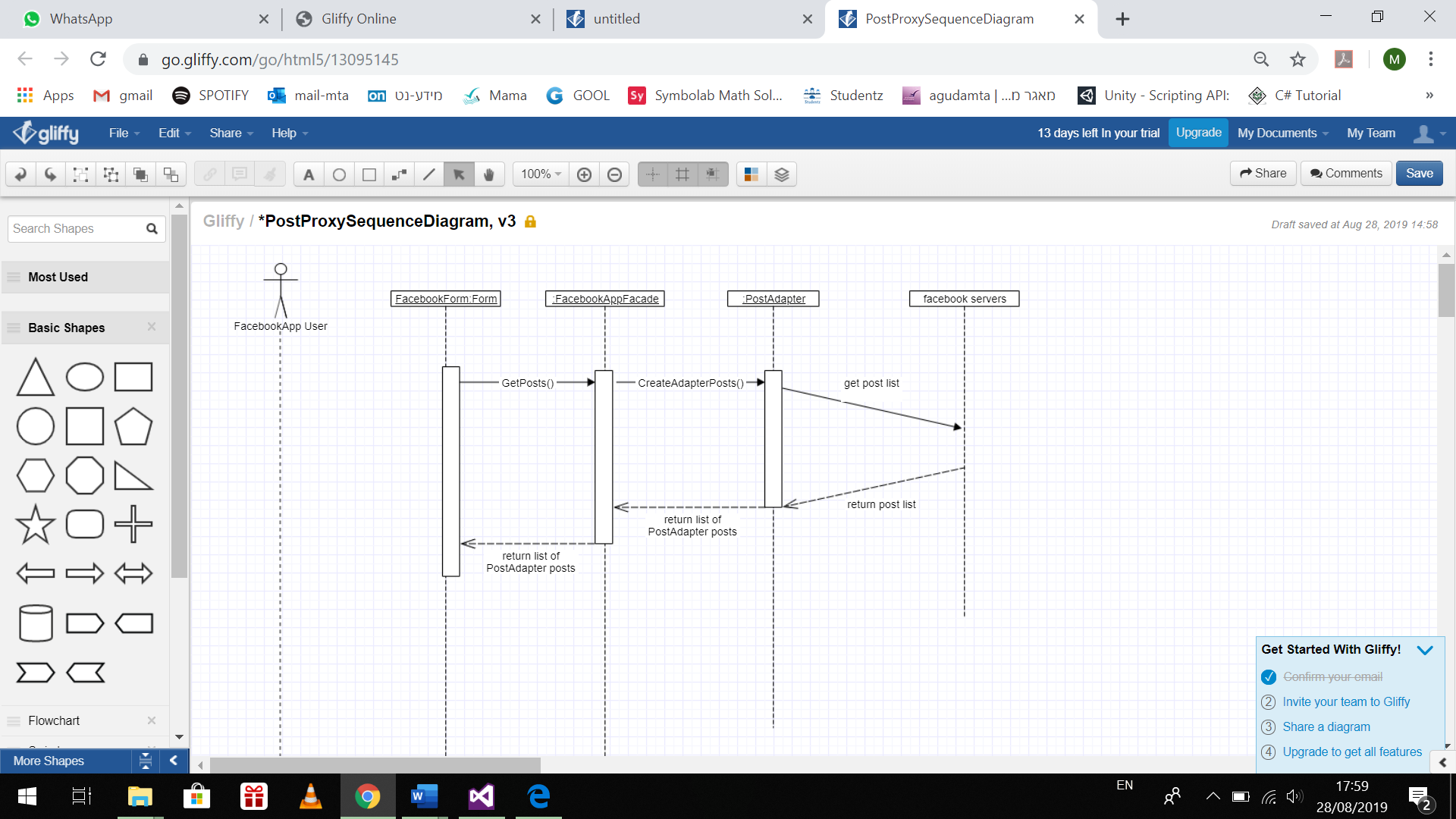
* אופן המימוש:

כפי שציינו בסיבת הבחירה בתבנית, הרכיב post לא מספר מתודת set עבור תוכן הפוסט ולכן הגדרנו מחלקה PostAdapter שתפקידה להתחזות לרכיב המקורי, היא מחזיקה post בקומפוזיציה ובנוסף מספקת שירות של set במידה והמשתמש מחליט לערוך פוסט קיים.

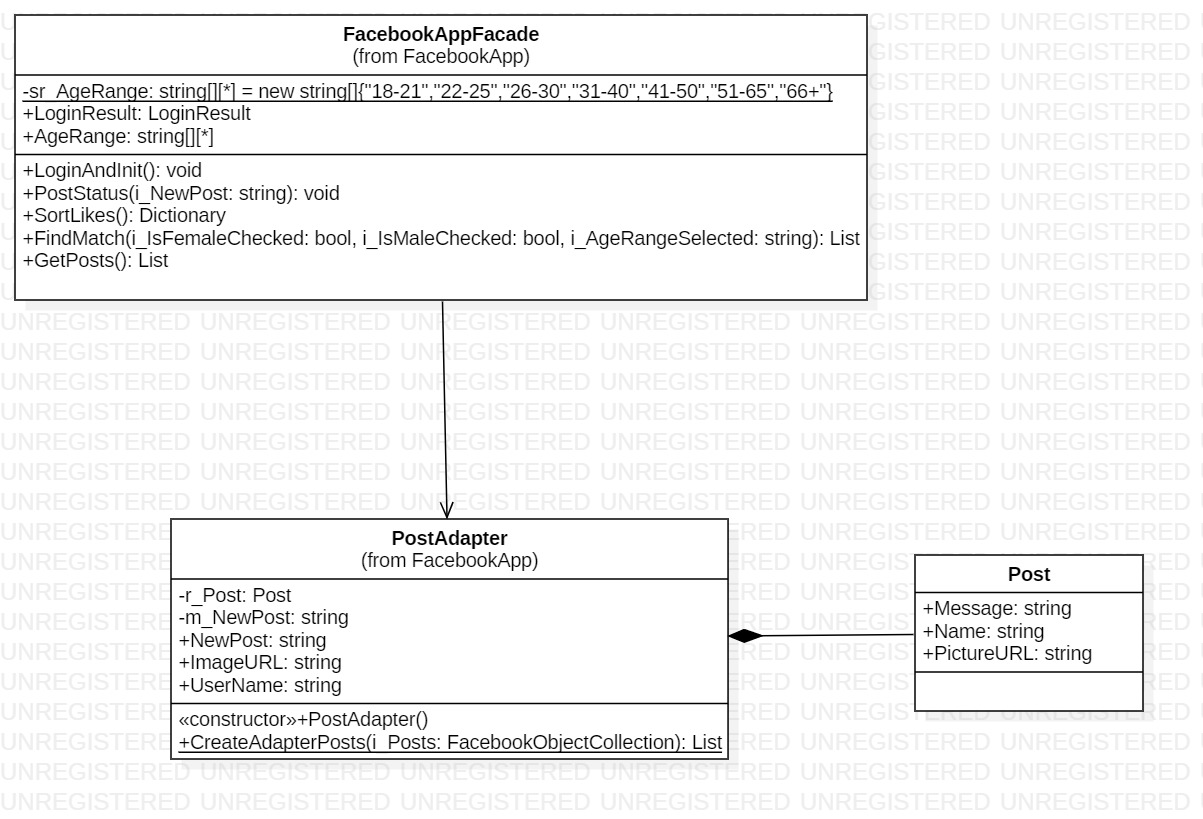
המחלקה PostAdapter מממשת מתודתCreatePostsAdapter() המקבלת רשימה של posts ודואגת להחזיק את הרשימה כרשימה של PostAdapter.

את מתודה זו מפעיל הclient FacebookAppFacade)) כאשר הוא מעוניין לגשת לרשימת הפוסטים של המשתמש.

* Sequence Diagram



* Class Diagram



**Target:**

**Adoptee:**

FacebookWrapper.ObjectModel.Post

**Adopter:**

PostAdapter

**Client:**

FacebookAppFacade

### שימוש בData Binding

* השתמשנו ב2 Way data binding בפיצ'ר ערכית הPost.

מכיוון שרכיב הPost המקורי שנמצא בdll שקיבלנו אינו מאפשר עריכה של פוסטים קיימים, ראינו לנכון לייצר רכיב אחר אשר יעטוף אותו ויספק את שירות העריכה. רכיב זה מחזיק בתוכו אובייקט מסוג Post לצד מתודת set המאפשרת את שירות העריכה בנוסף לשירותיו של רכיב Post.

לאחר מכן, יצרנו DataSource מהרכיב החדש – postAdaperBindingSource אשר מקושר ל listBoxPosts ובנוסף מציג את שם מפרסם הפוסט שאותו המשתמש בחר,תמונת הפרופיל,ואפשרות עריכה.

ברגע שהמתמש עורך פוסט, מתבצע השינוי בעת המעבר לרכיב אחר בטופס.

* בנוסף, השתמשנו ב1 way data binding עבור משיכת פרטי המשתמש.

לצורך כך, יצרנו DataSource בשם userBindingSource אשר מקושר לLoggedInUser.

### עבודה אסינכרונית

* חילקנו את טעינת הטופס ואת טעינת הפוסטים משרת הפייסבוק ל2 threads כאשר טעינת הטופס הראשי ונתוניו מהווים את הmain thread.