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

* Automate Facebook Tasks – למשתמש ישנה האפשרות לתזמן פעולות פייסבוקיות (העלאת תמונה, סטטוס או לינק) והפעולות יתבצעו במועד שנקבע על ידי המשתמש מראש.
* Shick O Shook – הגרלת חבר מרשימת החברים והצגת תמונה שלו ופרטים אודותיו. ניתן להחליף בין תמונותיו בלחיצה על התמונה, לבסוף ניתן לדרג "שיק" או "שוק" על החבר. לחיצה על אחד מהכפתורים הנ"ל תעלה סטטוס ובו רשומה הודעה בנוגע לבחירת היוזר.
* Calculate Statistics About Friends – המשתמש מקבל נתונים סטטיסטיים אודות רשימת החברים שלו (החבר הכי פעיל, יחס גברים\נשים, גילאים וכו').

### תבנית מס' 1 – Factory Method

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר Automate Facebook Tasks , המשתמש יכול לבחור מבין 3 אובייקטי פעולה שיתבצעו (העלאת פוסט, לינק או תמונה) בעלי בסיס פולימורפי משותף. יצירת אובייקט הפעולה הרלוונטי מתבצעת ע"פ בחירת המשתמש, ולכן סוג האובייקט המיוצר נקבע בזמן ריצה.   
הבעייה הכללית היא למעשה היכולת לייצר אוביקטים בזמן ריצה מבלי לציין את סוג אובייקט הפעולה הסציפי הנדרש ואת בעייה זו פתרנו על ידי השימוש ב Factory Method Pattern. בנוסף, השימוש בתבנית נותן לנו את היכולת להוסיף אובייקטי פעולה נוספים למערכת מבלי לשנות את הקוד.

* אופן המימוש:

את המימוש ניתן למצוא ב: Ex02.FacebookAppLogic.FBActionFactory.

מימוש התבנית מתבצע ע"י מתודת יצירה סטטית (כללית) אשר מקבלת את בחירת המשתמש ואחראית להפנות למתודת יצירה קונקרטית עבור אובייקט הפעולה הנדרש.

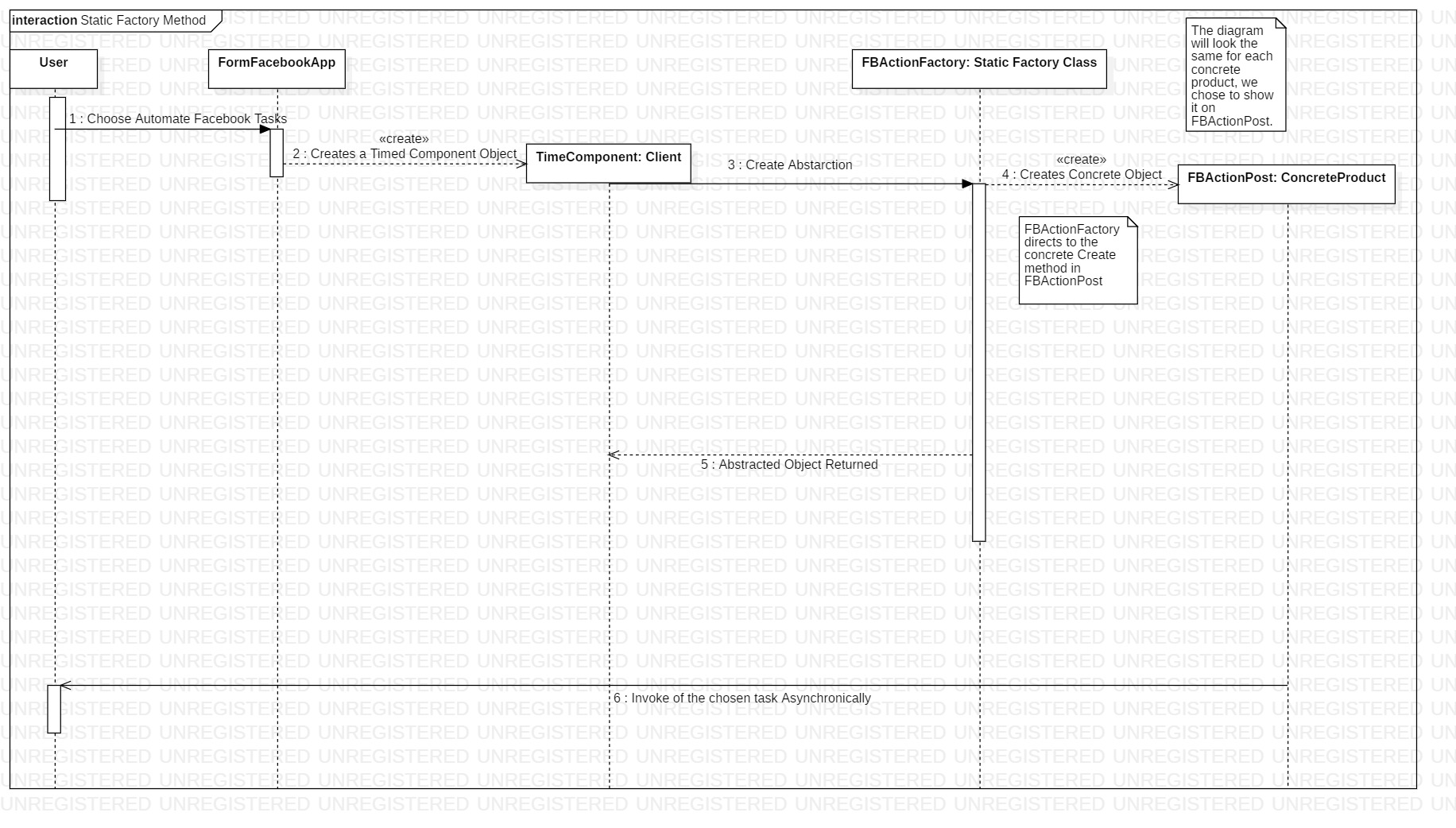
המחלקות המשתתפות בתבנית:

FBAction 🡪 Abstract Product – המחלקה הזו מגדירה את הבסיס הפולימורפי עבור האובייקטים הקונקרטיים אותם הFactory ייצר.

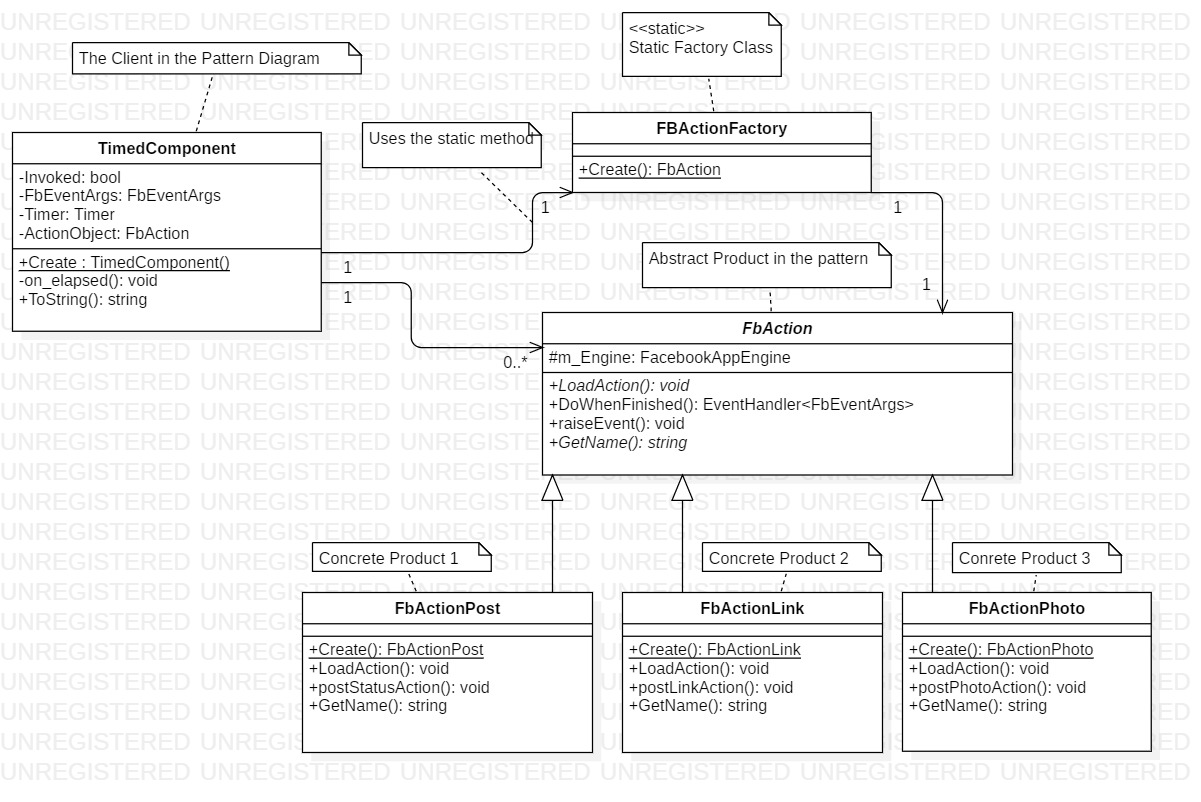
FBActionFactory 🡪 Static Factory Class (Creator) – במחלקה זו נמצאת מתודת הFactory אשר מפנה למתודת היצירה הקונקרטית עבור כל אובייקט פעולה.

FBActionPost, FBActionLink, FBActionPhoto 🡪 Concrete Product – אובייקטי פעולה מסוג פוסט, לינק ותמונה, אשר ממשים את הממשק המוגדר ב- FBAction.

* Sequence Diagram



* Class Diagram



### תבנית מס' 2 – Virtual Object Proxy

* סיבת הבחירה / שימוש בתבנית:

בפיצ'ר Automate Facebook Tasks המשתמש יכול לבחור מבין 3 אובייקטי פעולה (מפורטים מעלה). עבור כל אחד מאובייקטים אלו, קיים טופס (ה-Legacy Component במקרה שלנו), אשר בו המשתמש ממלא את הנתונים הנדרשים ולאחר סגירת הטופס, הנתונים נאספים ע"מ לייצר את אובייקט הפעולה המבוקש.  
יתכן והמשתמש כלל לא יבחר להשתמש בפיצ'ר ולכן אין טעם לייצר את הטפסים הנ"ל מראש. בחרנו להשתמש ב-Pattern Proxy בכדי לקבל את היכולת לייצר את הטפסים באופן Lazy , On First Demand.

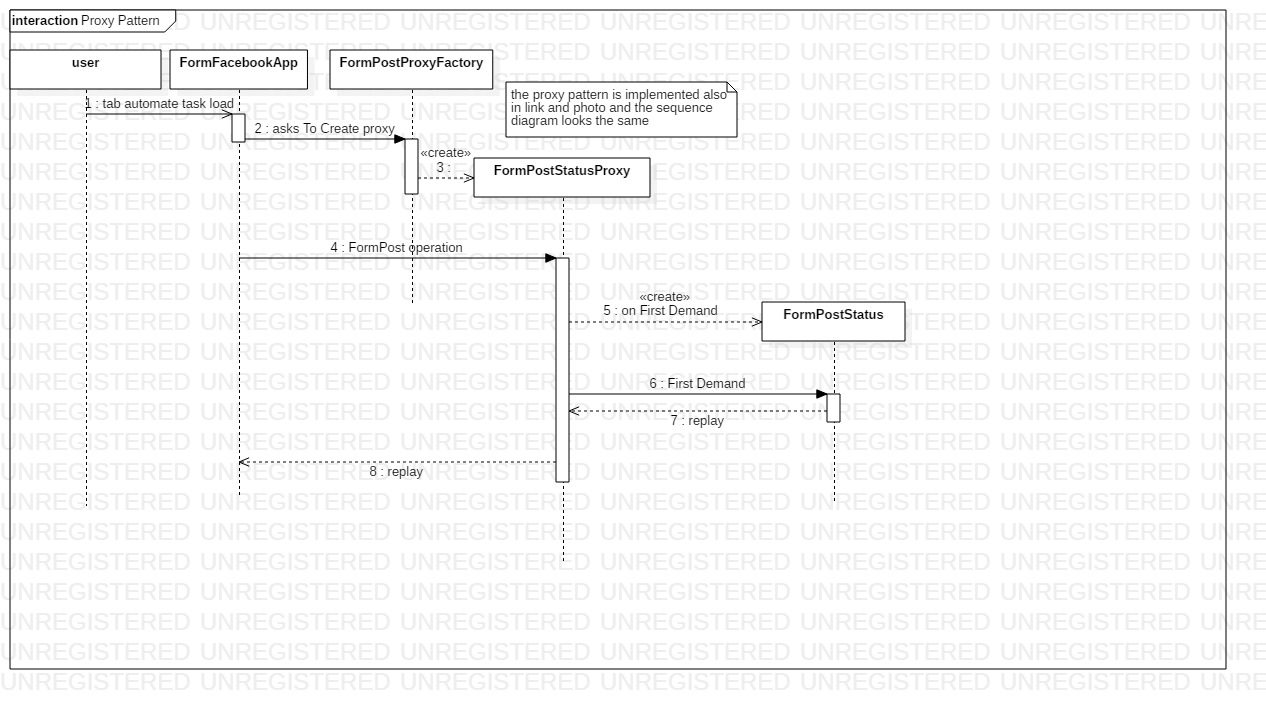
* אופן המימוש:

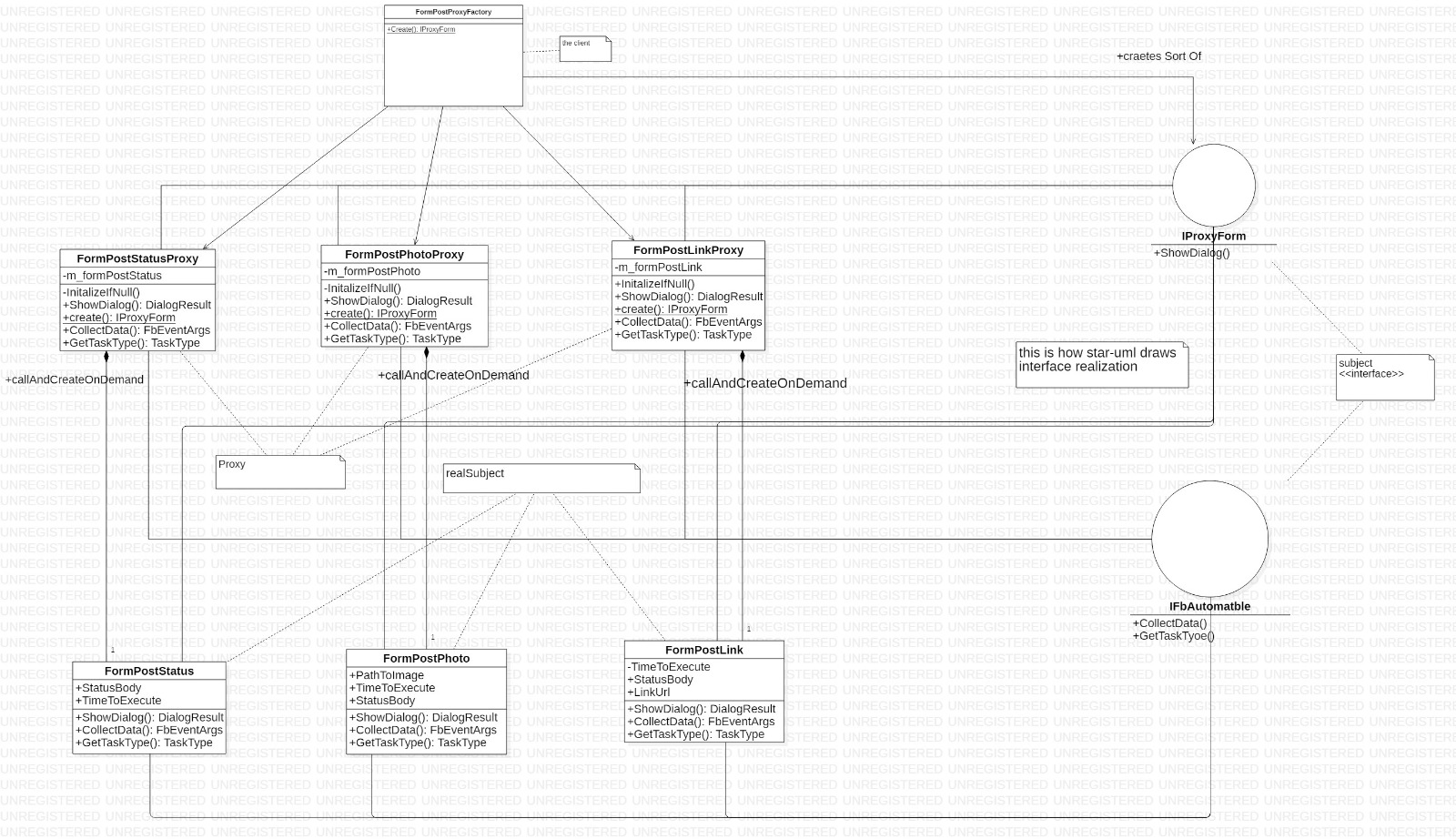
אנו מימשנו את הפטרן בכך ש-"הריל סאבג'קט" וה-"פרוקסי" ממשים את אותו ממשק( שני ממשקים : IProxy ,IFbAutomatable ובכך האובייקט פרוקסי יכול להתחזות ל- "ריל סאבג'קט". בנוסף האובייקט פרוקסי מכיל חבר מחלקה פרטי אשר הינו מסוג "ריל סאבג'קט" אשר בעת "פירסט דימנד" יצור אותו ויפעיל את הפעולה הרצויה.

ניתן למצוא את המימוש בקוד במחלקות: FormPostProxyFactory,FormPostStatusProxy, ,FormPostPhotoProxy, ,FormPostLinkProxy

ובממשקים:

IProxyForm,IFbAutomatable .

* Sequence Diagram
* Class Diagram



### תבנית מס' 3 – Façade

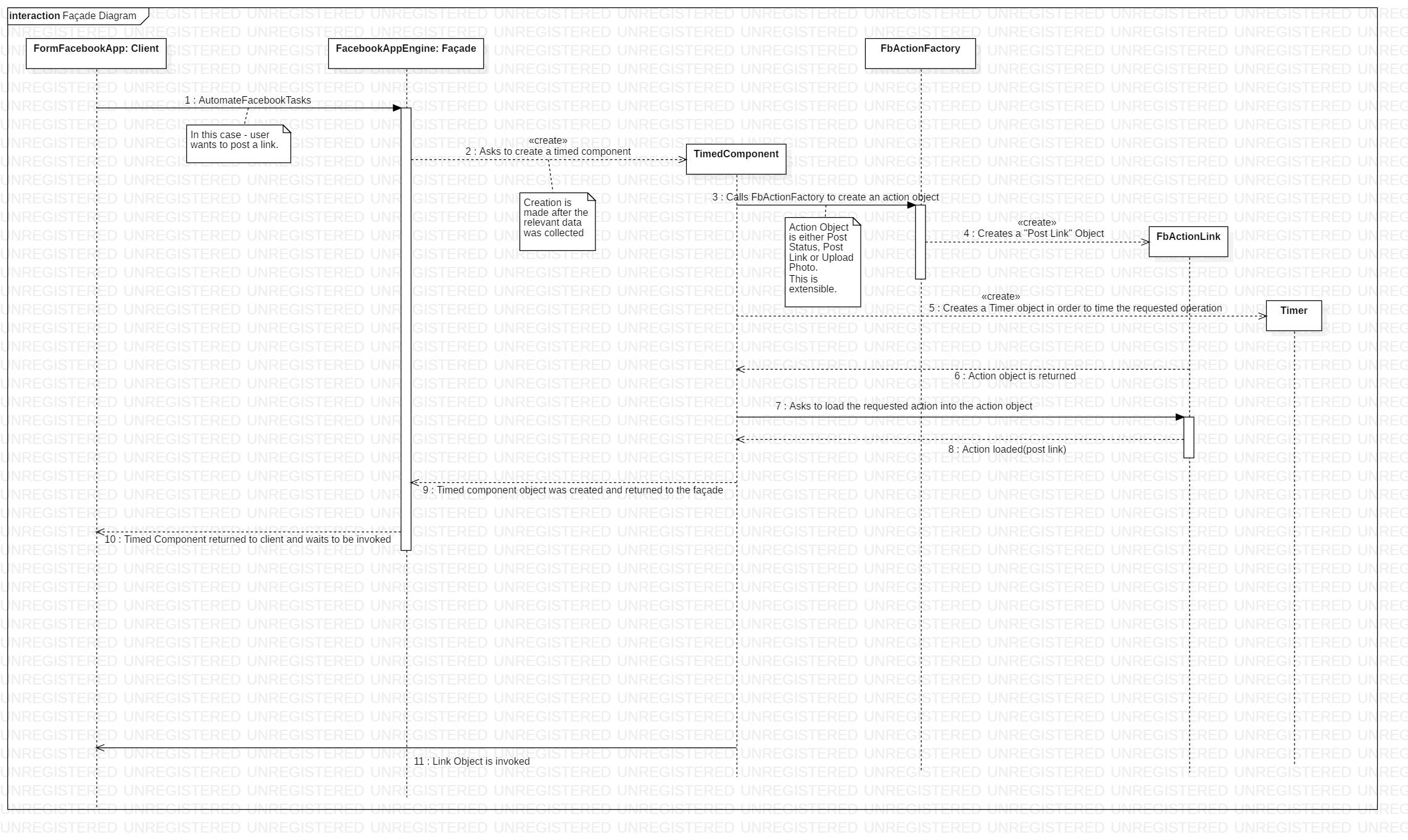
* סיבת הבחירה / שימוש בתבנית:

במערכת כולה, אנו משתמשים בתתי מערכות וברכיבים שונים, כדוגמת שירותי המחלקה FacebookService ואף הפיצ'רים ShickOShook ו- Statistics. בחרנו לספק ממשק שטוח לשירותים הנ"ל, באמצעות המחלקה FacebookAppEngine, אשר משמשת כ-Façade שקוף, ודרכה ה-Client יכול לגשת לכל השירותים באופן פשוט ומבלי לגשת לתתי הרכיבים באופן ישיר ומסובך.

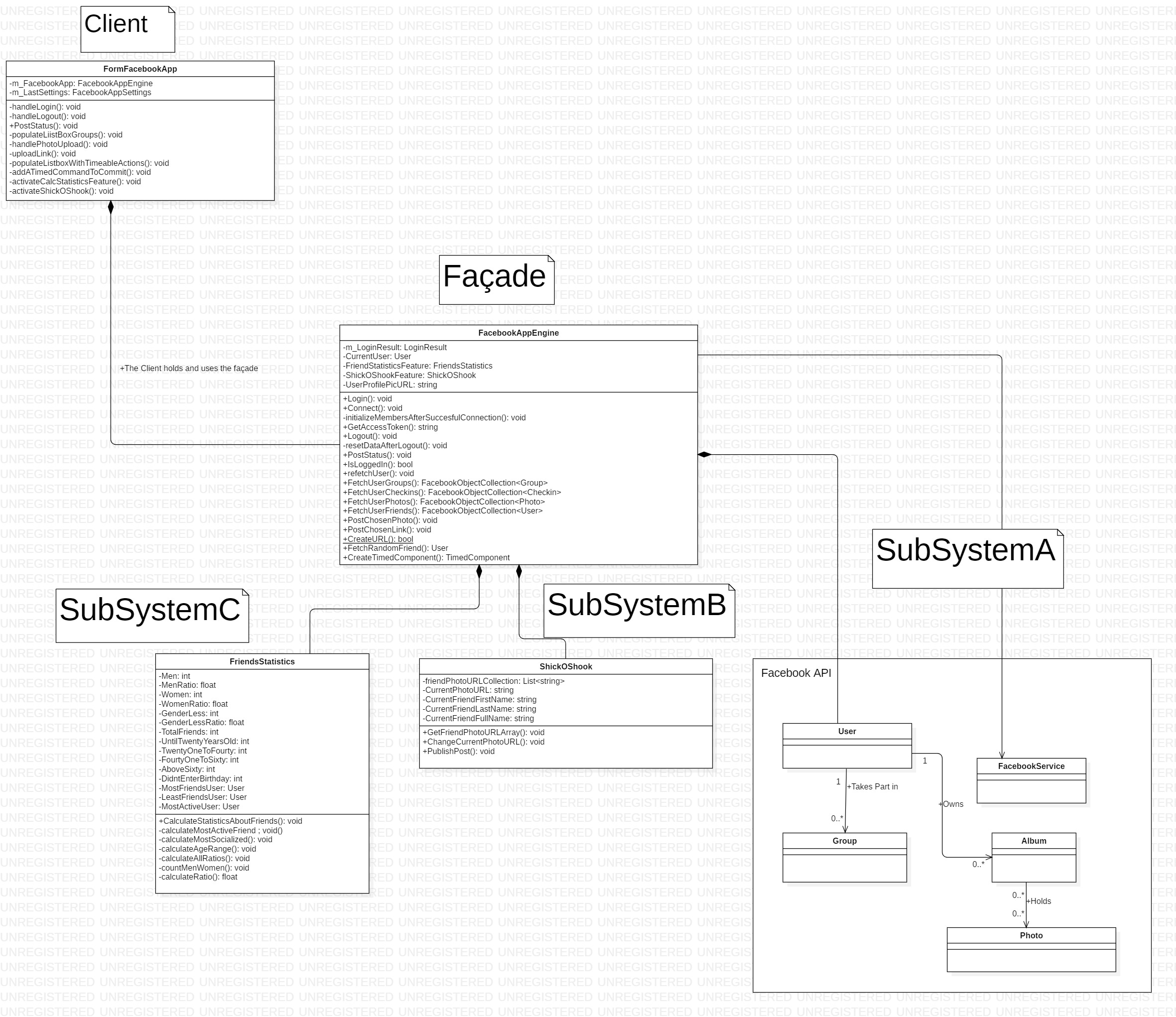
* אופן המימוש:

מימשנו את הPattern באופן הבא: המחלקה FacebookAppEngine מחזיקה בתוכה References לאובייקטיי הפיצ'רים (תתי-המערכות) הרלוונטיים, ובנוסף מחזיקה רפרנס ל-User. המחלקה חושפת ל-Client את השירותים הרלוונטיים למערכת בלבד, באופן פשוט ונוח ע"י מתודות אשר מבצעות את התהליכים עצמם אל מול הרכיבים הלוגיים, ולמעשה מחברת בין ממשק המשתמש(במקרה שלנו) לבין השכבה הלוגית.  
את המימוש ניתן למצוא במחלקה FacebookAppEngine.

* Sequence Diagram



* Class Diagram



### Two-Way Data Binding:

* ממומש בתרגיל ב2 מקומות בקוד:
  + FormFaceBookApp.buttonFetchGroups\_Click – כאשר מבקשים לקבל מידע אודות הקבוצות אליהן משתייך המשתמש.
  + FormFacebookApp.activateShickOShook – כאשר מביאים חבר רנדומלי, מעבר להבאת תמונותיו (מומש בתרגיל קודם), גם משתמשים ב- Data Binding ע"מ להביא נתונים עליו.

### עבודה אסינכרונית:

* ממומשת במקומות הבאים:
  + FormFacebookApp.addTimedCommandToCommit – שימוש ב-Lambda Expression על מנת שבמידה והפעולה המתבקשת לא מצליחה, שכבת המידע תדע להפעיל מתודה רלוונטית ע"פ ה-Lambda Expression שממשק המשתמש מספק. בנוסף, נעשה שימוש בפרמטר הנמצא ב-Scope העוטף את ה-Lambda Expression.
  + FormFacebookApp.Timer\_Elapsed – הפעלת מתודה אשר מרעננת את הטקסט המוצג בListBox שמחזיק את הפעולות שממתינות לביצוע. המתודה תופעל ב-Thread שבו נמצא ה-ListBox הנ"ל.
  + FbAction, FbActionPost, FbActionLink, FbActionPhoto, TimedComponent – למשתמש ניתנת האפשרות לתזמן ביצוע פעולות פייסבוקיות כמפורט מעלה. האובייקטים FbActionPost, FbActionPhoto, FbActionLink מממשים את המתודה LoadAction() שבה הם טוענים את המתודה אשר אמורה להתבצע בסוף הזמן המוגדר בעת זמן הריצה (נוספת אל ה-EventHandler). המתודה אשר נבחרה ע"י המשתמש מתבצעת באופן א-סינכרוני כאשר מגיע הזמן המבוקש לביצוע – החיבור הנ"ל (בין הזמן המבוקש לביצוע לבין המתודה המבוקשת) מתבצע במחלקה TimedComponent.