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

* מציאת אהבת אמת – בטופס ApplicationForm ישנו כפתור שכתוב עליו Find Out.  
  בלחיצה על כפתור זה, המערכת בודקת איזה User עשה בסה"כ הכי הרבה לייקים על התמונות של המשתמש, ומציגה ב PictureBox הסמוך את תמונת הפרופיל של אותו User. בנוסף, ישנן שתי checkboxes המאפשרות למשתמש לסנן לפי מין את "אהבת האמת" שהוא מחפש.
* מציאת החבר הטוב ביותר – בטופס ApplicationForm ישנו כפתור נוסף עליו כתוב Show Me !. בעת לחיצה עליו, המערכת מחפשת איזה User מתוייג הכי הרבה בסה"כ בכל התמונות של המשתמש. המערכת מציגה תמונה של ה"חבר" הזה באמצעות פתיחת טופס חדש למשתמש ב UI.

### תבנית מס' 1 – Facade

* סיבת הבחירה / שימוש בתבנית:

במערכת שלנו יש ספרייה בשם FBAppCore שמטרתה לשמש מחלקה תשתיתית עבור Clients שונים. כדי לשרת מטרה זו, עלה לצורך לפשט את הממשק של הספרייה עם הלקוח, כך שבמקום שהלקוח יפנה למחלקות מרובות כדי לבצע פעולות לוגיות, יוכל לפנות לממשק יחיד וברור המסתיר ממנו את המימוש הפנימי.

לדוגמא, כדי למצוא "אהבת אמת" או "חבר טוב ביותר", השכבה הלוגית מממשת שתי מחלקות נפרדות אותן העדפנו שלא לחשוף ללקוח - BestFriendMatcher ו-TrueLoveMatcher . לכן, "הסתרנו" את המחלקות הללו מאחורי ה facade שנקרא ApplicationLogicHandler החושף בפני הלקוח את הפעולות שמחולקות אלה מציעות באופן נגיש יותר.

כמו כן, במידה ובעתיד הספרייה התשתיתית תרצה להרחיב את הפונקציונליות שלה נוכל להרחיב את המימוש הפנימי של ה facade כך שיכולות המערכת משתפרות ללא שינוי של הממשק עם הלקוח. לדוגמא, יהיה ניתן לאסוף נתוני שימוש של הלקוח בספריית התשתית, איסוף המידע הזה יוכל להתבצע בfacade עבור כל פעולה שלקוח מבצע (לדוגמא, לספור את מספר הפעמים שנקראה המתודה FindTrueLove כדי למצוא אהבה.)

* אופן המימוש:

מימשנו facade בשם ApplicationLogicHandler אשר מופיע בספריית FBAppCore. הfacade מכיל שדות פרטיים שהם מופעים של מחלקות לוגיות נוספות הכרחיות למערכת כגון TrueLoveMatcher, BestFriendMatcher, LikeHandler.(גם הן יושבות בספרייה זו).

את המחלקות הללו רצינו להסתיר מספריית הלקוח FBAppUI ומספריות לקוח אחרות ולכן הגדרנו אותן כinternal. הfacade שמימשנו הוא façade חצי שקוף היות ואמנם המחלקות הפנימיות שלו מוסתרות מן הלקוח, אבל ללקוח יש גישה לספרייה תשתיתית יותר בשם FacebookWrapper שממנה הלקוח יכול לשחזר למעשה את הפעולות אותן ה facade מבצע.

* המחלקות ותפקידיהן בPattern

ApplicationLogicHandler – בתפקיד Façade

ApplicationForm, ImageForm – בתפקיד Clients

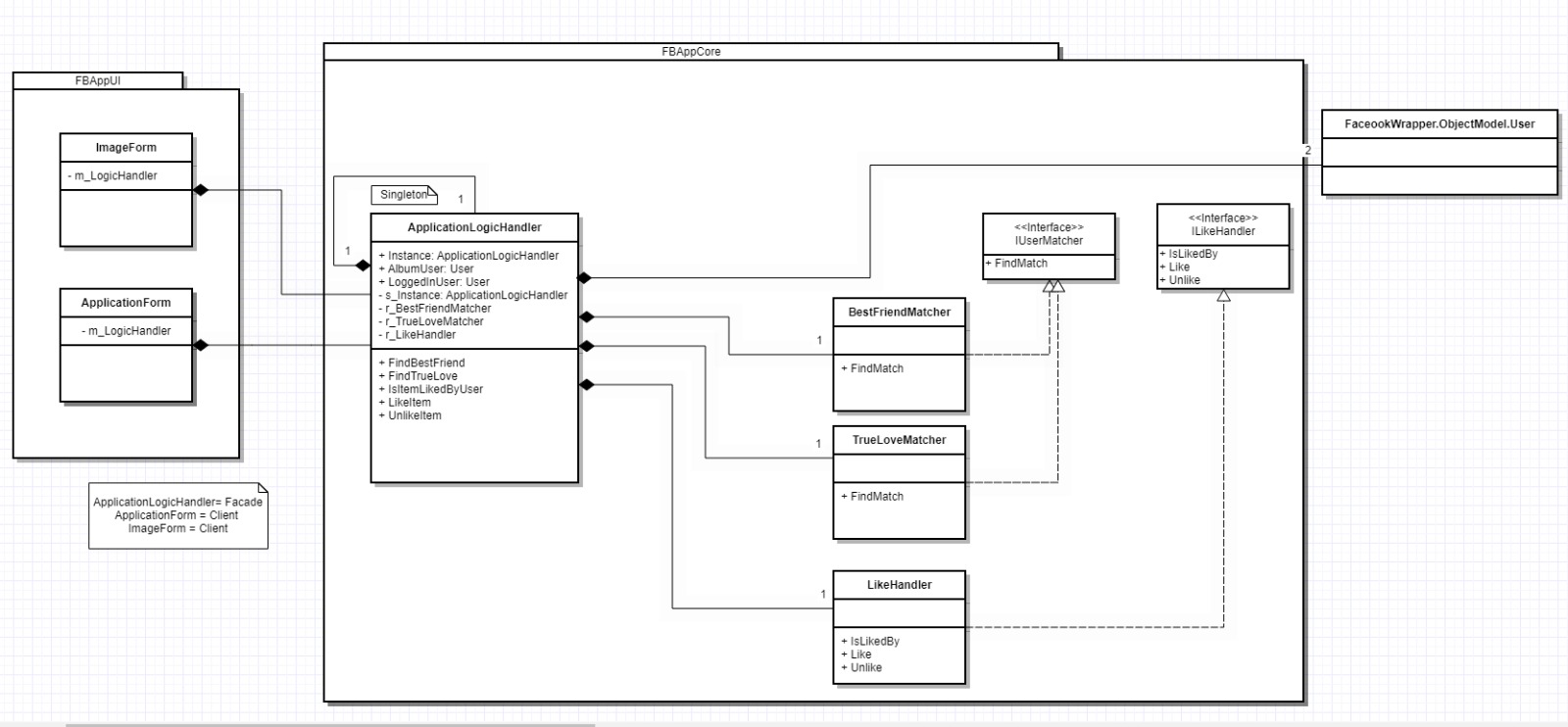
**דיאגרמות בעמ' הבא**

* Sequence Diagram

Graphical user interface, timeline

Description automatically generated

* Class Diagram



### תבנית מס' 2 – Proxy

* סיבת הבחירה / שימוש בתבנית:

במהלך הפרויקט, עלה הצורך לייצר כפתור משוכלל שמאפשר אכיפה של כמות קליקים על הכפתור בידי המשתמש. כלומר, בעת שהמשתמש לוחץ פעמים רבות מדי על הכפתור, הכפתור הופך לdisabled לזמן מוגדר, ומוקפץ למשתמש MessageBox שמציע לו לעבור לשלם על חבילת premium בכדי לשפר את חוויית השימוש.

רצינו שכפתור מיוחד זה יוכל להתווסף לטופס באותה קלות בה ניתן להוסיף כפתור רגיל, וזאת באמ' שימוש ב designer וללא תכנות נוסף. לכן, בחרנו להשתמש ב proxy pattern כדי לייצר button proxy באופן שבו הclient שהוא במקרה זה הטופס עצמו, יחשוב שהוא משתמש בכפתור רגיל כאשר למעשה מדובר בכפתור מיוחד.

* אופן המימוש:

את המימוש לproxy ביצענו בתיקיית controls שתחת ספריית הלקוח FBAppUI.

מימשנו מחלקה בשם ClickLimitButtonProxy שמהווה” “proxy by inheritance עבור המחלקה Button של Winforms (נאלצנו לעשות זאת היות ו-Button לא מממשת אף interface). כלומר, ClickLimitButtonProxy יורשת ממחלקת Button ומוסיפה לה פיצ'רים נוספים.

לצורך מימוש ההשהיה של הכפתור במצב disabled, השתמשנו במתודה הסטטית Delay של מחלקת System.Threading.Tasks.Task. כך שלאחר זמן מוגדר תופעל מתודה המחזירה את הכפתור למצב enabled. בנוסף, בסיטואציות בהן ה client "מודע" לשימוש במחלקה ClickLimitButtonProxy ולא מתייחס אליה כ-Button, הclient יכול להגדיר באופן ספציפי את התכונות הפומביות של האובייקט הכוללות ClickThreshold, TimeToEnabled, PromptMessage. במצבים אלו, מחלקה זו לא משמשת כ proxy מוחלט, אלא כפי שגיא כינה זאת בכיתה – "proxy עם קישוטים".

* המחלקות ותפקידיהן בPattern

ClickLimitButtonProxy – בתפקיד Proxy

Button – בתפקיד Subject

ApplicationForm – בתפקיד Client

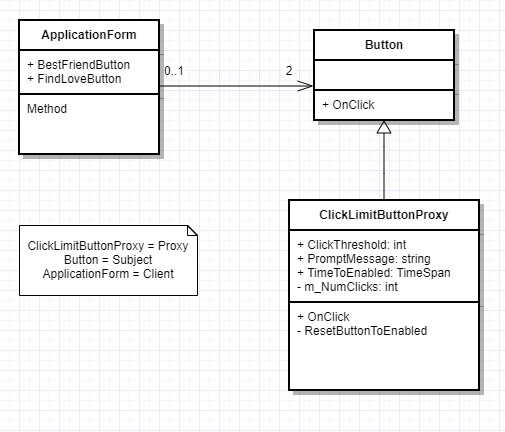
**דיאגרמות בעמ' הבא**

* Sequence Diagram

Diagram

Description automatically generated with medium confidence

* Class Diagram



### תבנית מס' 3 – Factory Method

* סיבת הבחירה / שימוש בתבנית:

בהקשר לפיצ'רים אותם מימשנו בתרגיל הקודם, שמנו לב שגם BestFriendMatcher וגם TrueLoveMatcher חולקים מאפיינים דומים כך שניתן להגדירם כמשפחה פולימורפית תחת הממשק IUserMatcher. כמו כן, בעת מימוש ה facade שלנו, שמנו לב שהוא מייצר מופעים של שתי המחלקות הללו בקונסטרקטור שלו. לכן, בכדי למנוע את יצירת המופעים ישירות בצד הלקוח (ה facade במקרה זה) החלטנו להגדיר מחלקה סטטית בעלת מתודה שתיצור מופעים ממשפחת IUserMatcher בהתאם לenum המגדיר באופן מפורש את המחלקות המממשות הנתמכות במערכת.

* אופן המימוש:

מימשנו את מחלקת UserMatcherFactory בתיקיית UserMatching בתוך הפרוייקט FBAppCore (בתוך תיקייה זו יושבות גם המחלקותTrueLoveMatcher, BestFriendMatcher, IUserMatcher, eUserMatcherType)

המחלקה UserMatcherFactory היא מחלקה סטטית שמממשת Factory Method בשם CreateMatcher המקבלת מופע של הenum eUserMatcherType ובהתאם לסוגו, משתמשת ב switch statement כדי ליצור מופע של מחלקה קונקרטית מתאימה – BestFriendMatcher או TrueLoveMatcher.

כמו כן, במידה והוכנס ערך שאינו נתמך במערכת, הfactory זורק NotSupportedException.

* המחלקות ותפקידיהן בPattern

UserMatcherFactory – בתפקיד Static Factory Class

ApplicationLogicHandler – בתפקיד Client

IUserMatcher – בתפקיד Abstract Product

BestFriendMatcher – בתפקיד Concrete Product A

TrueLoveMatcher – בתפקיד Concrete Product B

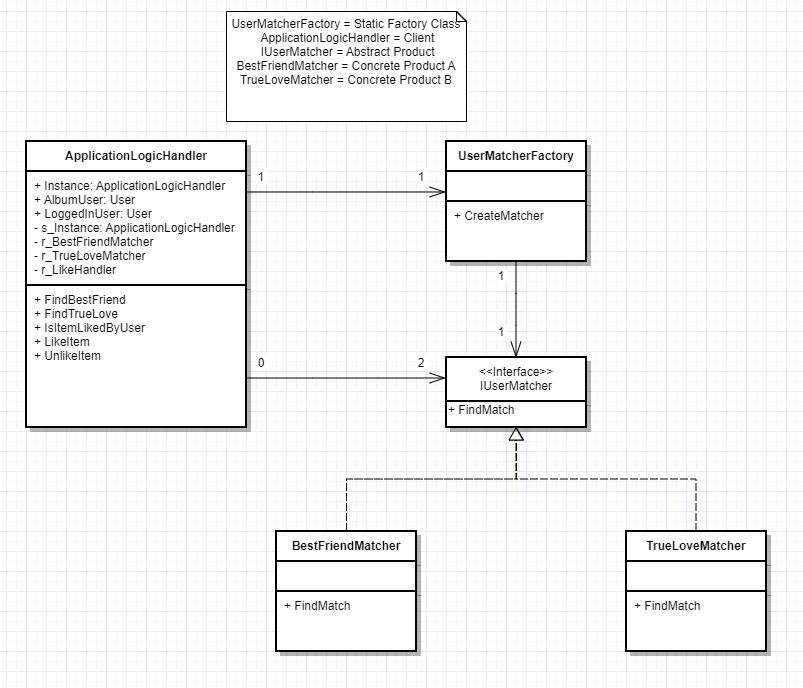
**דיאגרמות בעמ' הבא**

* Sequence Diagram

A picture containing timeline

Description automatically generated

* Class Diagram



### עבודה אסינכרונית

במסגרת התרגיל, בחרנו למממש עבודה אסינכרונית כדי לאפשר טעינה מהירה יותר של הטפסים, ושל המידע שמגיע משרתי Facebook.

לצורך כך, ראשית מימשנו רכיב לוגי בפרוייקט FBAppCore תחת תיקיית Multithreading – מחלקה בשם ThreadRunner.

מחלקה זו חושפת מתודה בשם RunMethodsAsSeperateThreads, המקבלת IEnumerable<Action> ועבור כל מתודה, מריץ אותה בThread נפרד (ישנה מתודה נוספת שהוא חושף שעושה אותו הדבר עבור Action בודדת לצורך קריאות בצד הלקוח).

בצד הלקוח, בטפסים ApplicationForm ו- AlbumPhotosForm מימשנו מתודות פרטיות שכל אחת פונה לשרתי פייסבוק בכדי להשיג מידע אחר, ואת שמות כל המתודות אנו מעבירים למתודה RunMethodsAsSeperateThreads במופע של המחלקה ThreadRunner כך שכל המתודות הללו מורצות באופן מקבילי.

את החלק של צד הלקוח ניתן למצוא בקוד במחלקות הטפסים שהוזכרו לעיל בתוך המתודה InjectData.

### Data Binding

במסגרת התרגיל, בחרנו למממש data binding בטופס ImageForm כדי להציג למשתמש את התגובות עבור תמונה מסויימת (המוצגת במסגרת ה ImageForm).

לשם כך, ייצרנו bindingSource חדש ששואב מידע מDataSource מסוג FacebookWrapper.ObjectModel.Photo. מ- DataSource זה יצאנו Control מסוג ListBox שיכיל את ה Comments עבור אותו ה Photo. כמו כן, הגדרנו את ה DisplayMember של הListBox להיות שדה ה Message של Comment בכדי שמה שיוצג למשתמש היא רשימת התגובות.

בעת טעינת הטופס ImageForm במתודה imageForm\_Load, אנו מריצים את המתודה bindPhotoToBindingSource בעזרת הThreadRunner שהוזכר לעיל, בכדי לקשר את הbindingSource לתמונה הספציפית ששייכת לטופס זה.