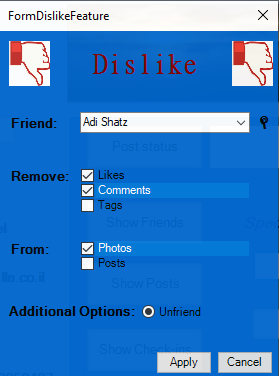
### תיאור קצר של הפיצ'רים שבחרנו לממש בתרגיל הקודם:

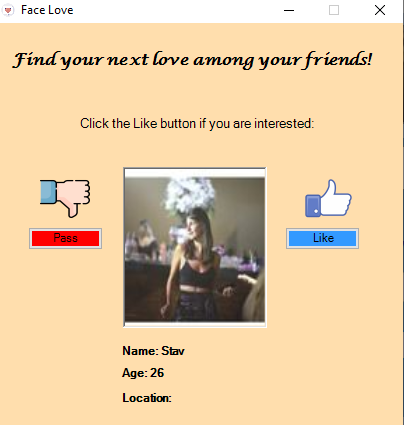
**feature FaceDislike**   
פיצ׳ר שמאפשר למשתמש לאתר משתמש מרשימת החברים שלו ולבצע כל אחת מן הפעולות הבאות:

* ביטול לייקים על תמונות\פוסטים של החבר.
* מחיקת תגובות על תמונות\פוסטים של החבר.
* הסרת תיוגים מתמונות\פוסטים משותפים עם החבר.
* אופציה לביטול החברות עם אותו המשתמש

**FaceLove feature**

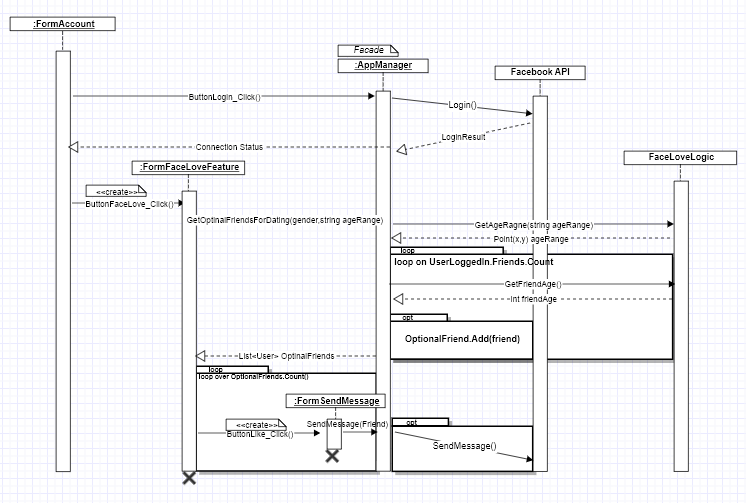
פי'צר שנותן למשתמש פלטפורמת הכרויות עם חבריו לפסייבוק.

המשתמש בוחר את המין בו הוא מעוניין ואת טווח הגילאים, הפיצ'ר מציג לו אחד אחרי השני את רשימת החברים העונים על הקריטריונים שבחר, והמשתמש יכול ללחוץ "like" או “pass” כאשר לחיצה על “like” נותנת למשתמש אפשרות לשלוח הודעה אישית לחבר שאהב.

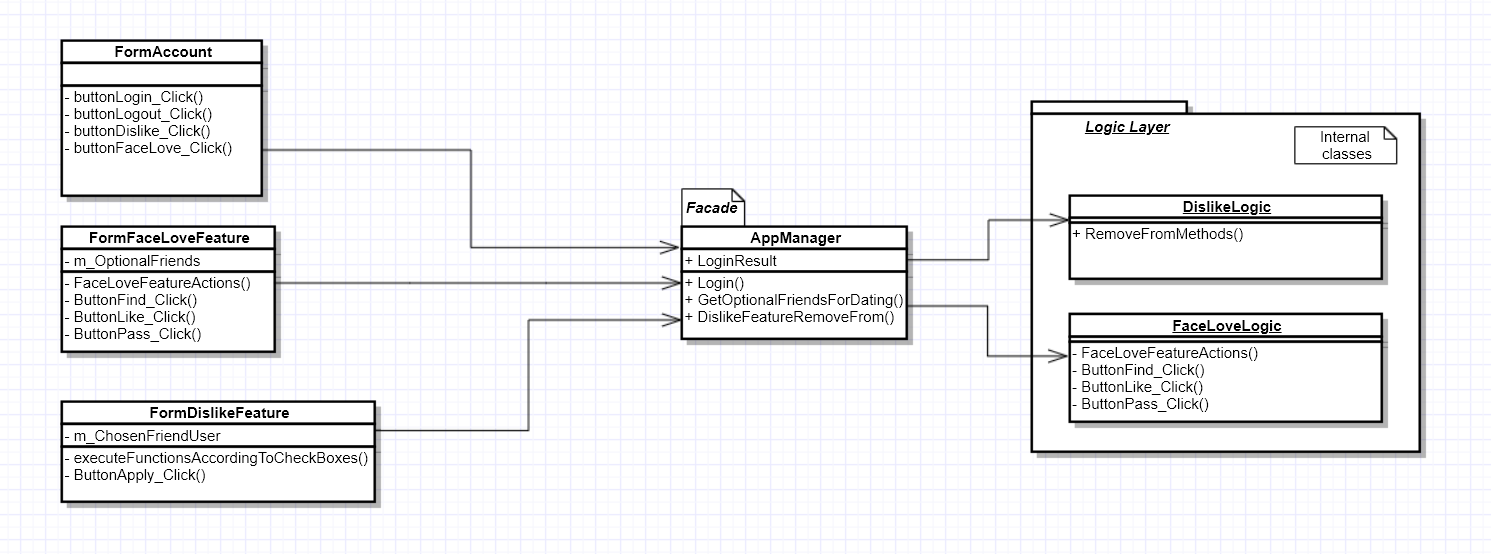


### תבנית מס' 1 – Facade

* **סיבת הבחירה / שימוש בתבנית:**
* הפרדה בין ממשק המשתמש לעולם הלוגי, כאשר העולם הלוגי מרובה תהליכים לדוגמא עבודה מול שרת Facebook ועיבוד המידע עבור הפיצ׳רים השונים באפליקציה.
* בחרנו לממש Facade שתפקידו להנגיש לפשט ולהשטיח את המידע על מנת להסתיר את המורכבות של אופן הבאת המידע במחלקות הלוגיות ובכך להקל על מי שמשתמש בהן.
* הפרדה זו מאפשרת בעתיד את החלפת ממשק המשתמש עבור הפיצ׳רים.
* **אופן המימוש:**
* בחרנו לממש את ה-**AppManager** כ-Facade סינגלטוני כיוון שאנו סבורים כי הוא צריך להיות המנהל היחיד והבלעדי של האפליקצייה שלנו, ולא ראינו לנכון לממש מספר facadים מכיוון שמדובר באפליקצייה שמימדייה תואמים מימוש של חוצץ יחיד בין הלוגיקה לממשק המשתמש.
* בחרנו לממש את ה-Façade כאטום מכיוון שמימוש בצורה זו תורמת לנוחות תחזוקה בעתיד ובטיחות המשתמש בפונקצונליאות האפליקצייה. כלומר, ממשק המשתמש לא מכיר את כל תתי המערכות והתהליכים בלוגיקה ולכן בעתיד אם נרצה לשנות משהו בלוגיקה לא נצטרך לעדכן את ממשק המשתמש.
* Sequence Diagram



* Class Diagram



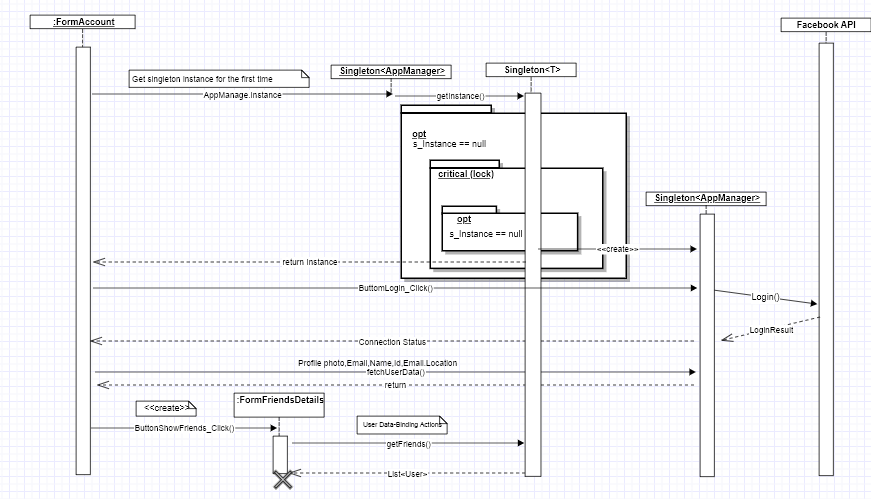
### תבנית מס' 2 – Singleton

* סיבת הבחירה / שימוש בתבנית:
* בחרנו לממש את ה-**AppManager** כסינגלטון כיוון שאנו סבורים כי הוא צריך להיות המנהל היחידי והבלעדי של האפליקצייה שלנו מהסיבות הבאות:
* מחלקת **AppManager** מחזיקה כmember- את LoginResult שהתקבל מפעולת ה-Login ב-Facebook API. באפליקציה ישנו משתמש אחד ויחיד מחובר בלבד ולא יכול להיות מצב שבו יהיה משתמש מחובר נוסף.
* בנוסף, בחרנו ב-Pattren Singleton מכיוון שאנו רוצים למנוע מצב שבו ישנם מספר מופעים של AppManager המחזיקים בתוכם State שונה (במקרה שלנו, user מחובר שונה) דבר שיגרום למצב לא תקין של האפליקציה.
* כמו כן, ברוב הטפסים המוצגים בזמן השימוש באפליקציה ניגשים למידע של המשתמש המחובר בהתאם לצרכיהם, כלומר יש צורך מובהק בגישה למחלקה **AppManager** (הממומשת כסינגלטון) ממחלקות לוגיות ו-UI שונות בקוד. כלומר להעביר את אופן הגישה לאחריות המחלקות – הן אלו שייזמו את הגישה אל המידע של המשתמש המחובר במקום שכל מחלקה בהכרח תחזיק רפרנס אל היוזר המחובר.
* בעתיד, אם נרצה להרחיב את יכולות האפליקציה ולממש פיצ׳רים נוספים המצריכים מידע על המשתמש נוכל לעשות זאת בקלות על ידי פעולה יזומה של המחלקות החדשות לקבלת Instance מהסיגנלטון **AppManager**.
* **אופן המימוש:**
* שימוש במחלקה סטטית גנרית Singelton T
* בעת הכניסה למערכת, מופעלת פונקציה Instance מהמחלקה Singelton T שמייצרת בפעם הראשונה והיחידה את FacebookManager, באופן

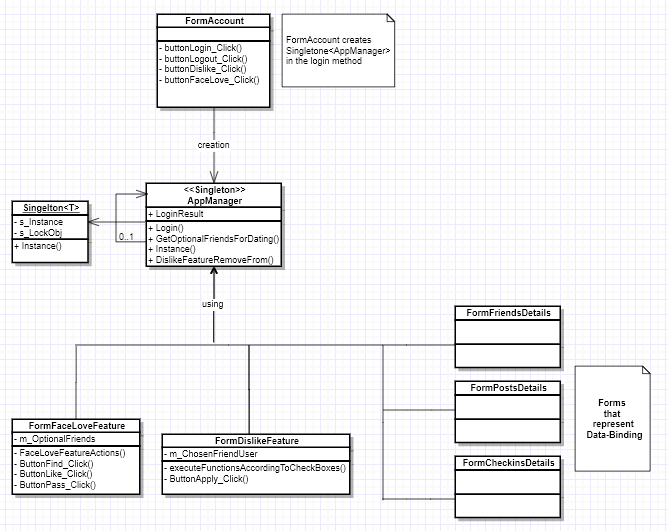
fully lazy, on demand and threat safe.

הטפסים השונים משתמשים כעת במידע הרלוונטי בו הם מעוניינים ע״י פניה למופע הסינגלטון (במקום לקבל את המשתמש המחובר כרפרנס כפי שמימשנו בתרגיל 1).

* Sequence Diagram

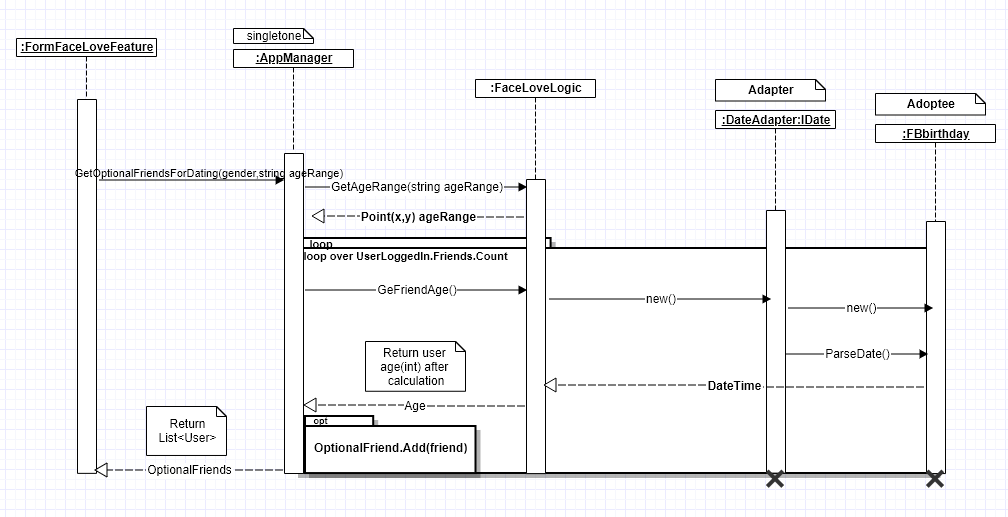


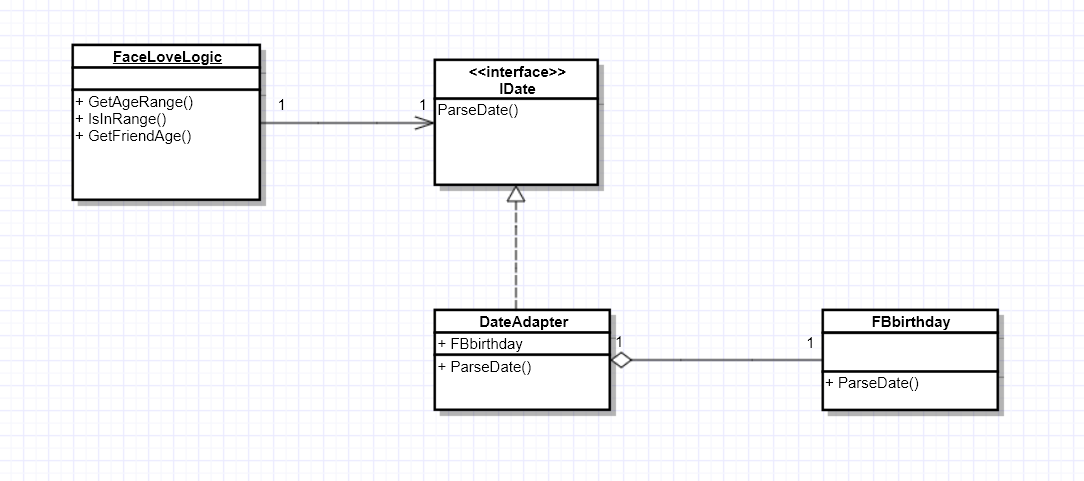
* Class Diagram



### תבנית מס' 3 – Adapter

* **סיבת הבחירה / שימוש בתבנית:**
* הפיצ׳ר FaceLoveשבחרנו לממש עושה שימוש בנתון ״תאריך לידה״ של משתמשי הפייסבוק. תאריך הלידה הוא propery מסוג string ביוזר של פייסבוק, האפליקצייה שלנו מעוניינת בתאריך זה בייצוג של DateTime על מנת לבצע חישובים מדוייקים (למשל חישוב הגיל של המשתמש על מנת לדעת האם החבר הנ''ל הוא מתאים) בחרנו לממש את Pattern Adapter על מנת להתגבר על חוסר ההתאמה בממשק בין הפיצ'ר לבין תאריך לידתו של המשתמש.
* אופן המימוש:
* מימוש IDate Interface , מחלקה DateAdapter המממשת את IDate כ-**Adapter** ומחלקה FBbirthdate כ-**Adoptee**.
* ההתאמה מ-string ל-DateTime מתבצעת ע״י המתודה ParseDate של המחלקה FBbirthdate.
* Sequence Diagram



* Class Diagram 

### עבודה אסינכרונית

* ניתן למצוא במחלקה FormAccount כאשר לוחצים על כפתור ה-Login ולשם הבאת המידע של היוזר המחובר אל הטופס הראשי (המידע כולל: תמונה ראשית, תמונת רקע, פרטים אישיים..)
* החלטנו להשתמש בעבודה אסינכרונית בחלק זה באפליקצייה, מכיוון שאנו מעוניינים שכאשר משתמש מתחבר למערכת, הטופס יהיה אינטראקטיבי תוך כדי שהוא טוען את הנתונים ולא יגרום למשתמש לחשוב שהתוכנית נתקעה ולגרום לבלבול/חוסר הבנה של המתמש לגבי תקינותה של האפליקציה.

### עבודה עם Data Binding

* ניתן למצוא בטופס הראשי לאחר לחיצה על הכפתורים: Show Friends, Show Posts, Show Checkins. לחיצה על כל אחד מהכפתורים תגרום לפתיחת טופס חדש שבו יוצגו ליסט-בוקס (**שהData Source שלו מקושר ל-Data Binding שיצרנו**) ופאנל (**שמציג את הפרטים שיצרנו בעזרת Data Binding Generator**).
* החלטנו להשתמש בData Binding מכיוון שהוא חוסך בשורות קוד (data binding ידני לוקח מספר שורות קוד ואילו בשיטה הנ''ל נדרשנו לשורת קודת בלבד עבור כל data binding שחברנו לממש) , ומקל על העבודה מול הנתונים שאנו מחזיקים בלוגיקה.