

Chat Stream

## **עבודת גמר בתכנון ותכנות מערכות במסלול פיתוח שרותי אינטרנט, תכנות אסינכרוניות ומסדי נתונים**

מוגש על ידי דניאל צימקובסקי  
תעודת זהות 326288016  
העבודה בוצעה בהנחיית יאנה זמוסטיאנו

תשפ"ג 2023

## ספר פרוייקט - דניאל צימקובסקי

פרויקט Chat Stream

דניאל צימקובסקי 326288016

שם הפרויקט: Chat Stream

מבוא:

### תיק פרויקט -מבוא:

- **רקע על הפרויקט:**
- **תיאור קצר:** הפרויקט הינו "רשת חברתית" שבה ניתן להתכתב עם אנשים בקבוצות או בזוגות, וניתן לפרסם פוסטים (מודעות) בפרופיל של המשתמש. משתמשים אחרים יכולים לתת לייק לפוסט ולהגיב.

**פוסטים** - עבור פוסט ניתן לקבוע אם הוא פרטי (רק חברים של אותו משתמש/ יכולים לראות) או ציבורי (כל משתמש/ יכול/ה לראות את הפוסט).

**חברויות** - כל שני אנשים יכולים לתת אחד לשני סטטוס של "חברים" בעזרת בקשת חברות. חברים מאפשרים אחד לשני לראות את הפוסטים הפרטיים.

**צ'אטים** (התכתבויות) - צ'אטים, הם התכתבות online הנשמר בזמן אמת כדי לאפשר למשתמשים לחזור לצ'אט ולראות את ההתכתבות. צ'אטים ניתן לחלק לשני סוגים: התכתבות ישירה בין זוג אנשים, וקבוצת התכתבות עם כמות מרובה של משתמשים. צ'אטים קבוצתיים יכולים להיות ציבוריים (כל אחד ואחת יכול/ה להצטרף והמנהלים יכולים גם לצרף), וצ'אטים פרטיים (רק המנהלים יכולים לצרף אנשים).

**קבוצת התכתבות** (צ'אט קבוצתי) - בכל קבוצה יש מנהלים שיכולים להסיר משתמשים, לחסום משתמשים (מניעת כניסה מחדש לקבוצה) ולהוסיף משתמשים. המנהל יכול לשנות את פרטי הקבוצה כמו שם הקבוצה, תיאור, והאם היא פרטית. רק מנהלים יכולים להוסיף או לאשר כניסה של משתמשים לקבוצה. מנהלים יכולים להוסיף את חבריהם מבלי לשלוח הזמנה. כשמנהל רוצה לצרף משתמש/ת שאינו חברו נשלחת הזמנה לאותו משתמש.

**חסימה** - כל אדם יכול גם לחסום אנשים מסויימים (משמעות החסימה היא לא לאפשר לנחסם לתקשר עם החוסם) או לדווח על משתמשים שאינם מתנהגים באופן הולם ומכבד כלפי כלל המשתמשים.

- **קהל יעד:** אנשים מעל גיל 13, שהינו הגיל המינימלי לשימוש באפליקציות מסוג זה, שמעוניינים לתקשר עם אנשים קרובים בעזרת צ'אט או לדבר עם אנשים שונים בצ'אטים ציבוריים. מיועד גם למי שרוצה לשתף מחשבות בעזרת פוסט.

- בחרתי בנושא זה כי אני מעוניין לממש רשת תקשורת בין אנשים. התחום של רשתות חברתיות מתפתח בתאוצה גדולה בימינו. אני רוצה להתנסות בבניית רשת חברתית קטנה ופשוטה, כדי להיכנס קצת לעולם הזה, ולצבור ניסיון בתחום.

### מטרות המערכת:

מטרות על:

- ליצור תקשורת בין משתמשים, בין אם זה בצ'אטים פרטיים או קבוצתיים.
- ליצור פלטפורמה שבה ניתן לשתף מידע, מחשבות ואירועים משמעותיים בצורת פוסטים.

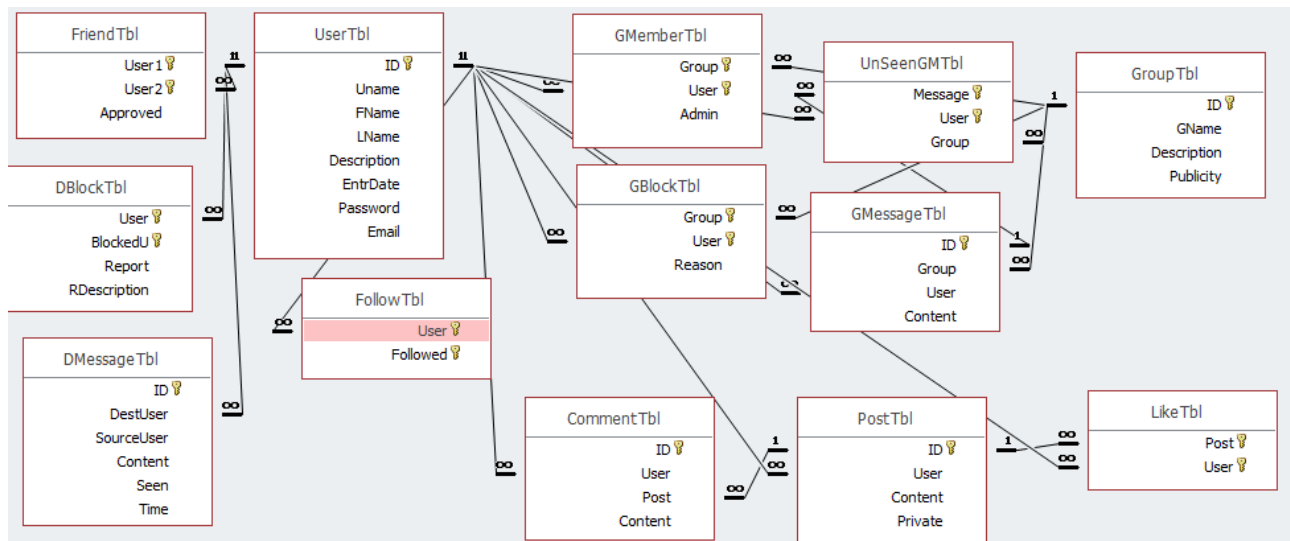
מטרות נלוות:

- לכל קבוצה של התכתבות יהיו מנהלים. המנהל יכול להוסיף ולהסיר משתמשים, הוא קובע מי נמצא ומי לא נמצא בקבוצה. הוא זה שיכול לשנות מידע על הקבוצה כמו התיאור שלה, והשם שלה.
- הפרדה בין פוסטים פרטיים שרק אנשים שהוגדרו כחברים יכולים לראות, ופוסטים של המשתמש שכולם יכולים לראות.
- אפשרות לעשות לייקים לפוסטים במידה והמשתמש אהב את הפוסט, ותגובות על מנת שיוכל המשתמש להגיב.
- אפשרות בחירה אם קבוצה היא ציבורית או לא, על מנת שהמנהלים יוכלו לקבוע אם כל אחד יכול לראות את הקבוצה ולבקש והצטרף או שאף אחד לא יכול לראות אותה, ורק מי שנמצא בקבוצה יכול לראות אותה.
- חסימת משתמש: לא לאפשר למשתמש לתקשר עם המשתמש שחסם אותו.
- חסימת משתמש מקבוצה: להוציא את המשתמש לאלתר- לא לאפשר לא להיכנס או לשלוח בקשת הצטרפות. מנהלים יכולים לקבוע אם המשתמש מורחק, ויכולו להחזיר משתמש מורחק רק אם יבטלו קודם את החסימה.
- קליטת קלט בעייתי: קלט שיכול להשפיע על הקוד sql ולגרום לפקודה לא רצויה. לאפשר לקלוט גרשיים- איפה שצריך לאפשר.
- התחברות והרשמה של המשתמש - כל השדות צריכים להיות מלאים

תיאור המערכת:

המערכת הינה רשת חברתית שבה משתמשים יכולים להתכתב בקבוצות או אחד לאחד, להעלות פוסטים, להגיב עליהם ועוד.  
כל משתמש תחילה מתחבר או נרשם למערכת ומשם נפתחות לכל המשתמש את האופציות להתכתב, ולראות פוסטים של משתמשים אחרים.  
המשתמש

## תיאור מערכת מסדי הנתונים: הרחבת DATABASE



טבלאות ראשיות:

UserTbl:

UserTbl
ID
Uname
FName
LName
Description
EntrDate
Password
Email

טבלת המידע על המשתמש - מכילה את המידע על המשתמש.

שדות המידע על המשתמש הם:

ID מספר מזהה - מסמל את המשתמש

Uname שם משתמש - מזוהה אם המשתמש

FName שם פרטי, LName שם משפחה

Description תיאור - ביוגרפיה קצרה של המשתמש

EntrDate תאריך כניסה - תאריך הצטרפותו של המשתמש לאפליקציה

Password סיסמה - סיסמת כניסה

Email אימייל - כתובת דוא"ל לתקשורת.

\*\*כל אדם שנכנס לאפליקציה הינו משתמש רגיל

קבוצת התכתבות

ID - מספר מזהה שמסמל את הקבוצה.

GName - שם הקבוצה

Description - תיאור הקבוצה - מידע שמנהלי הקבוצה יכולים להוסיף פירוט ומידע נוסף על הקבוצה.

Publicity - האם כל אחד יכול למצוא ולהצטרף לקבוצה או שהקבוצה פרטית, ורק המנהלים יכולים לצרף אנשים.

טבלת פוסטים:

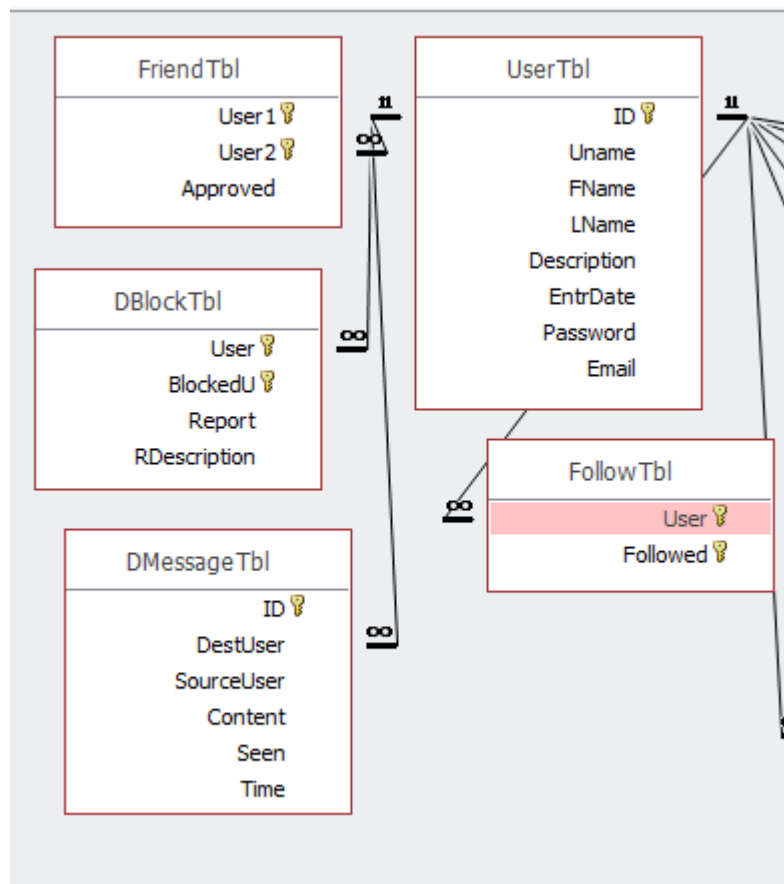
ID - מספר מזהה פוסט (מבדיל בין פוסטים)

GroupTbl
ID
GName
Description
Publicity

PostTbl
ID
User
Content
Private

User - מספר שמזהה את המשתמש היוצר  
Content - תוכן הפוסט, התוכן שהמשתמש מעוניין להציג  
Private - האם רק מי שמחזיק בסטטוס חבר יכול לראות? אחרת כל המשתמשים יכולים לראות

טבלאות משניות:



DBlockTbl : טבלת חסימות ישירות- מניעת שליחת הודעות ותקשורת

User- משתמש חוסם

BlockedU - המשתמש החוסם

Report - האם גם לדווח על המשתמש על שימוש לרעה

RDescription - תיאור סיבת הדיווח

FollowTbl : טבלת עוקבים בין אנשים

User: המשתמש העוקב

Followed: המשתמש שעוקבים אחריו

FriendTbl : טבלת החברויות בין אנשים

User1, User2 - שני האנשים אשר נותנים זה לזה סטטוס חבר

User 1 - שלח את הבקשה

User 2 - קיבל את הבקשה

### DMessage - הודעות פרטיות בין שני משתמשים

ID - מספר מזהה של ההודעה

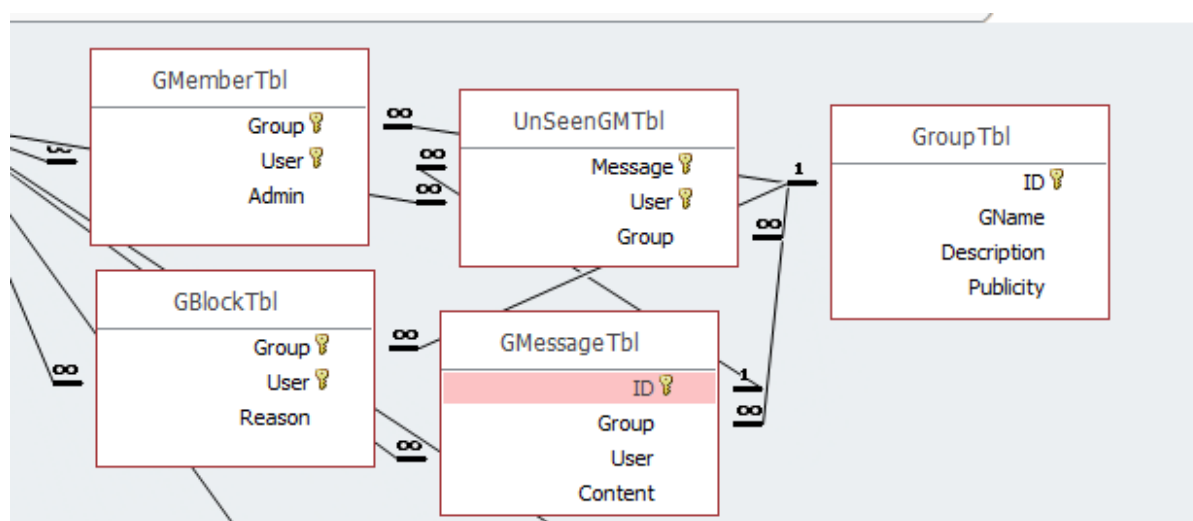
DestUser - המספר המזהה של משתמש/ת יעד

SourseUser - המספר המזהה של המשתמש/ת ששלח/ה את ההודעה

Content - תוכן ההודעה

Seen - האם המשתמש/ת יעד ראה וקיבל את ההודעה

Time - הזמן שההודעה נשלחה



### GMemberTbl: טבלת החברים בקבוצת התכתבות

Group - המספר המזהה של קבוצת ההתכתבות

User - מספר המזהה של המשתמש/ת שנמצא בקבוצה

Admin - האם באותה קבוצה יש למשתמש/ת מעמד של מנהל

### GBlockTbl: רשימת המורחקים מקבוצה

Group - הקבוצה ממנה הורחק המשתמש

User - מספר מזהה של המשתמש/ת המורחק/ת

Reason - סיבת ההרחקה

### GMessageTbl: טבלת ההודעות בתוך הקבוצות

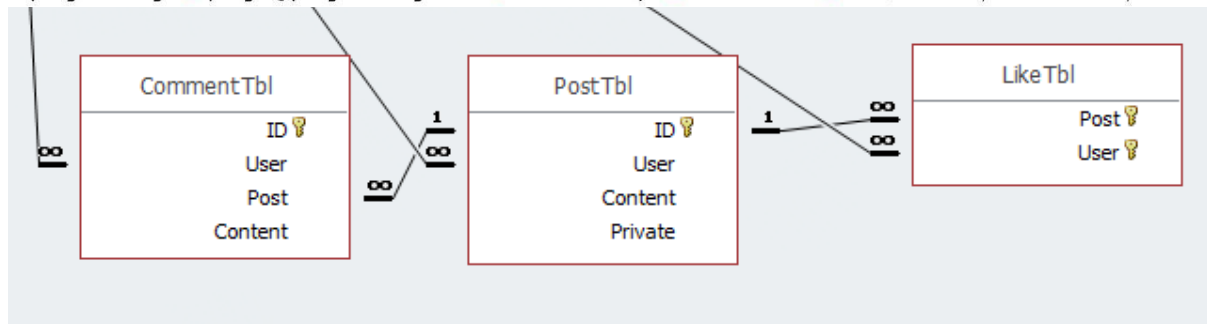
ID - מספר מזהה של ההודעה

Group - המספר המזהה של הקבוצה בה נשלחה ההודעה

User - המספר המזהה של המשתמש/ת שכתב/ה את ההודעה

Content - תוכן ההודעה

Time - זמן השליחה



### CommentTbl: טבלת התגובות לפוסטים

- ID - מספר מזהה של התגובה
- User - המספר המזהה של כותב/ת התגובה
- Post - המספר המזהה של הפוסט שעליו המשתמש מגיב/ה
- Content - תוכן התגובה

### LikeTbl: הוספת סימון "אהבתי" לפוסט

- Post - המספר המזהה של הפוסט שסומן בלייק
- User - המספר המזהה של המשתמש שאהב את הפוסט

ModelView - מקבץ כל המחלקות שניגשות למסד נתונים ומוציאות מידע מתוך המסד נתונים

### קשרי גומלים:

קשרי גומלין בין טבלאות database מקשרים שדות משותפים שמקשרים שורות בין שני הטבלאות המקושרות. הקשרים האלה נועדו למנוע מצב שבו טבלה אחת מופנית לרשומה בטבלה אחרת שלא קיימת, בכך שיש אכיפה בין קשרי הגומלין.

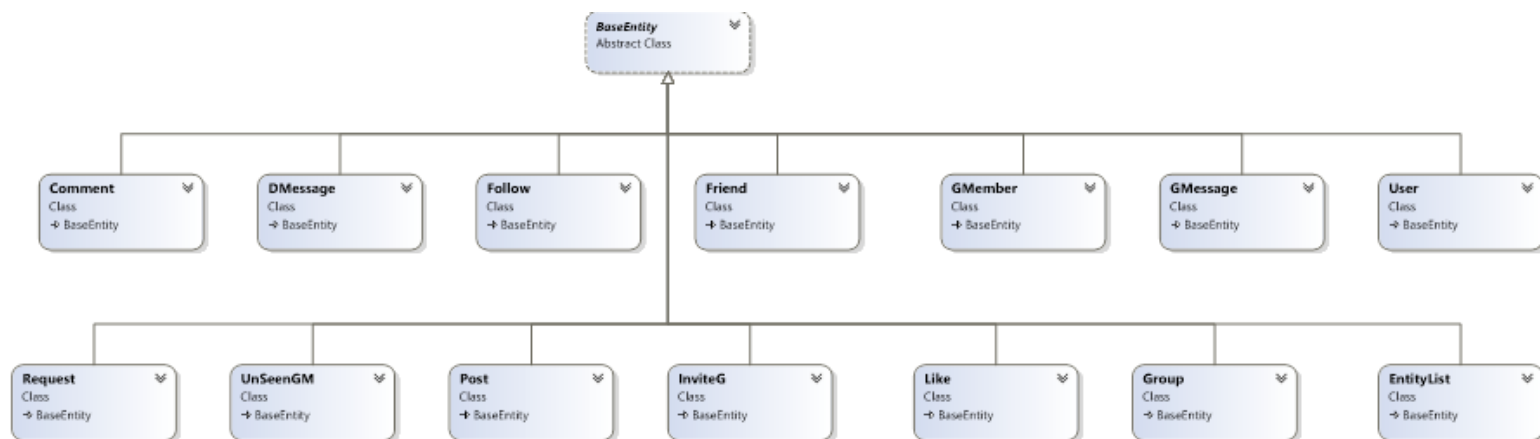
### יחיד לרבים:

עבור כל מופע בטבלה 1 יכולים להיות מספר רשומות מותאמות בטבלה 2. שדה אחד בטבלה 1 מותאם לשדה אחד יחיד בטבלה 2, אך יכולות להיות חזרות. לדוגמא LikeTbl ו UserTbl יש קשר של יחיד לרבים כי משתמש אחד יכול לעשות לייק לכמה פוסטים, ככה שיכול להיות חזרה של לייקים שנעשו על ידי המשתמש. שדה ID ב- UserTbl מקושר בקשר זה לשדה User ב LikeTbl.

### רבים לרבים:

שדה אחד בטבלה 1 מתאים למספר שדות בטבלה 2, ויכולות להיות חזרות על השדות. לדוגמא בטבלאות FollowTbl ו UserTbl. לשדה ID ב UserTbl, מותאמים שני שדות ב FollowTbl שהם User, Followed, מאחר ואנו רוצים לקשר שני משתמשים מ UserTbl בתוך הטבלה FollowTbl.

### :Model



99+ references  
 public abstract class BaseEntity  
 {  
 }

BaseEntity היא מחלקה אבסטרקטית אשר לא ממושת. בעזרת הירושה של כל המחלקות מן המחלקה הזו, ניתן להשתמש בפולימורפיזם מה שמאפשר להשתמש בתבניות כלליות כדי למנוע חזרה של קוד כאשר אנו רוצים להכניס או להוציא מידע מהdatabase.

מחלקות נבחרות:

```

public class User : BaseEntity
{
    //properties
    private int id; //מספר מזהה של המשתמש
    private string email; //כתובת דוא"ל
    private string uname; //שם משתמש
    private string fname; //שם פרטי
    private string lname; //שם משפחה
    private string description; //ביוגרפיה או תיאור של המשתמש
    private DateTime entrDate; //תאריך יצירת משתמש
    private string password; //סיסמת המשתמש
}
    
```



```
5 references
public string Description { get => description; set => description = value; }
2 references
public DateTime EntrDate { get => entrDate; set => entrDate = value; }
8 references
public string Password { get => password; set => password = value; }
76 references
public int ID { get => iD; set => iD = value; }
9 references
public string Uname { get => uname; set => uname = value; }
3 references
public string Fname { get => fname; set => fname = value; }
3 references
public string Lname { get => lname; set => lname = value; }
5 references
public string Email { get => email; set => email = value; }
```

פעולות קובעות ומאחזרות נוצרו בעזרת צורת כתיבה lambda, בעזרתה ניתן להגדיר פעולות מבלי להכריז עליהן מראש.

```
66 references
public class Group : BaseEntity
{
    //properties
    private int iD; //מספר מזהה של הקבוצה
    private string gNname; //שם הקבוצה
    private string description; //תיאור הקבוצה
    private bool publicity; //האם הקבוצה ציבורית או פרטית
    //getters and setters
    23 references
    public int ID { get => iD; set => iD = value; }
    4 references
    public string GNname { get => gNname; set => gNname = value; }
    3 references
    public string Description { get => description; set => description = value; }
    3 references
    public bool Publicity { get => publicity; set => publicity = value; }
}
```

```
8 references
public class GroupList : List<Group>
{
    0 references
    public GroupList() { }
    0 references
    public GroupList(IEnumerable<Group> list) : base(list) { }
    2 references
    public GroupList(IEnumerable<BaseEntity> list) : base(list.Cast<Group>().ToList()) { }
}
```

רשימות של עצמים כמו GroupList הם מחלקות שהוגדרו לכל מחלקה שיורשת מ BaseEntity שמטרתן להגדיר list של אותם עצמים כאשר מה שהמחלקה מקבלת הינו IEnumerable של אותם עצמים.

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
IEnumerable הוא סוג האובייקט שמוחזר מה-database, הבנאי ממיר את הרשימה המתקבלת ל-list של אובייקטים מאותו הסוג של הlist הרצוי. במקרה זה המרנו את התוצאות ל-GroupList היורש את תכונותיו מ <List<Group

## Friend

```
39 references
public class Friend : BaseEntity
{
    //properties
    private User user1; //first user who initiated friend request
    private User user2; //second user
    private bool approved; //is the friendship is approved by the second user.

    //constructors
    0 references
    public Friend(User user1, User user2, bool approved)
    {
        this.user1 = user1;
        this.user2 = user2;
        this.approved = approved;
    }
    1 reference
    public Friend() { this.approved = false; }
    0 references
    public Friend(User user1, User user2)
    {
        this.user1 = user1;
        this.user2 = user2;
        this.approved = false;
    }

    //getters and setters
    6 references
    public User User1 { get => user1; set => user1 = value; }
    6 references
    public User User2 { get => user2; set => user2 = value; }
    3 references
    public bool Approved { get => approved; set => approved = value; }
}
```

מחלקה בעלת שני משתמשים כתכונות שלה. User1 הוא המתמשש שיזם מלכתחילה את בקשת החברות וUser2 הוא המתמשש שקיבל את הקשה. השדה Approved מסמל האם User2 אישר את בקשת החברות.

DMessage:

33 references

```
public class DMessage : BaseEntity
```

```
{
```

```
    //properties
```

```
    private int id; //מספר מזהה של ההודעה
```

```
    private User destUser; //משתמש יעד של ההודעה
```

```
    private User sourceUser; //משתמש שכתב את ההודעה
```

```
    private string content; //תוכן ההודעה
```

```
    private bool seen; //האם ההודעה נשלחה ונראתה על ידי משתמש היעד
```

```
    private DateTime time; //זמן שליחת ההודעה
```

```
//constructors
```

1 reference

```
public DMessage()
```

```
{ }
```

0 references

```
public DMessage(User destUser, User sourceUser, string content)
```

```
{
```

```
    this.DestUser = destUser;
```

```
    this.SourceUser = sourceUser;
```

```
    this.content = content;
```

```
    this.seen = false;
```

```
    this.id = 0;
```

```
}
```

```
//getters and setters
```

3 references

```
public int ID { get => id; set => id = value; }
```

3 references

```
public string Content { get => content; set => content = value; }
```

4 references

```
public bool Seen { get => seen; set => seen = value; }
```

1 reference

```
public DateTime Time { get => time; set => time = value; }
```

9 references

```
public User DestUser { get => destUser; set => destUser = value; }
```

8 references

```
public User SourceUser { get => sourceUser; set => sourceUser = value; }
```

המחלקה DMessage היא מחלקה שנועדה ליצור אובייקטים שמחזיקים מידע על הודעה שנשלחה ממשתמש אחד למשתמש אחר.

SourceUser- הוא המשתמש ששלח את ההודעה אל DestUser

Time מכיל את הזמן שבו ההודעה נשלחה

Conetnt מכיל את תוכן ההודעה

Seen מסמל האם ההודעה נשלחה אל המשתמש UserDest.

הסימון הזה נועד כדי שהשרת ידע אילו הודעות לשלוח למשתמש יעד בזמן אמת, כדי לשלוח

רק את ההודעות שלא נראו בזמן אמת, במקום לטעון את כל ההתכתבות כל כמה שניות.

6 references

```
public class EntityList : BaseEntity
```

```
{
```

```
    private List<BaseEntity> entities;
```

1 reference

```
    public EntityList()
```

```
{
```

```
        this.Entities = new List<BaseEntity>();
```

```
}
```

0 references

```
    public EntityList(List<BaseEntity> entities)
```

```
{
```

```
        this.entities = entities;
```

```
}
```

5 references

```
    public List<BaseEntity> Entities { get => entities; set => entities = value; }
```

```
}
```

EntityList - מחלקה שירשת מ BaseEntity שנועדה ליצור רשימה של BaseEntity, על מנת שיהיה אפשר לשלוח כמה אובייקטים של BaseEntity לפעולה שיוצרת שאילתת SQL, מאחר והיא מוגבלת לקבל רק BaseEntity.

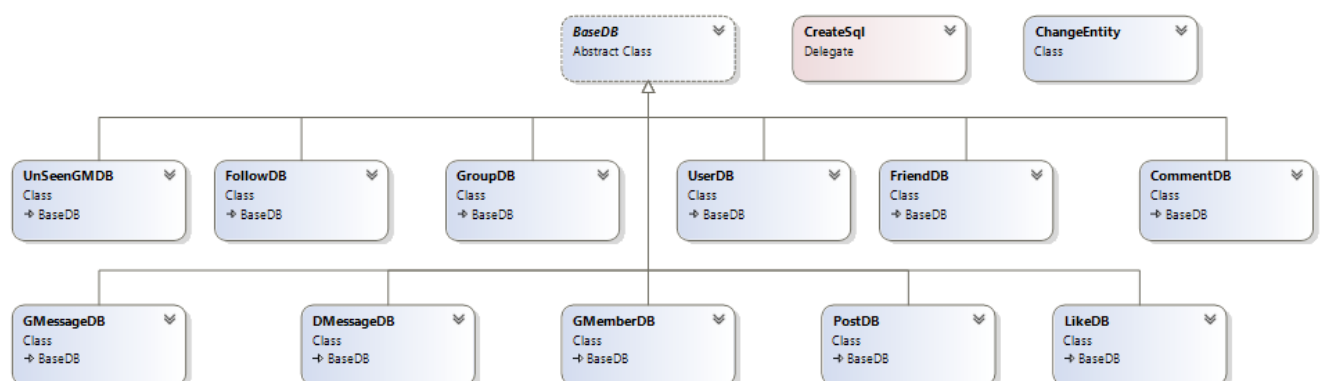
```

23 references
public class UnSeenGM : BaseEntity
{
    private GMessage message; //group message
    private int user; // user ID to reduce memory usage
    private int group; // group ID to reduce memory usage
    //constructorss
    1 reference
    public UnSeenGM() { }
    4 references
    public UnSeenGM(GMessage message, int user, int group)
    {
        this.message = message;
        this.user = user;
        this.group = group;
    }
    //getters and setters
    5 references
    public GMessage Message { get => message; set => message = value; }
    3 references
    public int User { get => user; set => user = value; }
    4 references
    public int Group { get => group; set => group = value; }
}
    
```

UnSeenGM- מחלקה שנועדה ליצור אובייקטים המסמלים האם ההודעה בקבוצה נראתה על ידי משתמש מסוים. נועד כדי לדעת האם לשלוח הודעה חדשה למשתמש שמבקש לקבל את ההודעות שלא ראה.  
חלק מהתכונות אינן עצם אלא מספר מזהה של אותם עצמים על מנת לחסוך מקום אחסון מיותר והעתקת מידע שאינו לצורך.

### :View Model

ה-ViewModel מכיל את כל המחלקות שאחראיות על הכנסת והוצאת מידע מתוך מסדי הנתונים. כל מחלקה בתוך ה-ViewModel אחראית להכניס ולהוציא מידע מתוך טבלה מסוימת במסד הנתונים.



```
public delegate string CreateSql(BaseEntity entity);
41 references
public class ChangeEntity
{
    private BaseEntity entity;
    private CreateSql createSql;

    34 references
    public ChangeEntity(CreateSql createSql, BaseEntity entity)
    {
        this.entity = entity;
        this.createSql = createSql;
    }

    5 references
    public BaseEntity Entity { get => entity; set => entity = value; }
    3 references
    public CreateSql CreateSql { get => createSql; set => createSql = value; }
}
```

ChangeEntity - מחלקה שנועדה כדי ליצור מופעים של שאילות שמשנות את מסד הנתונים כגון הוספה, עדכון ומחיקה. המחלקה בעלת שתי תכונות: entity - מופע של אובייקט של השאילתה createSql - הוא delegate של פעולות שיוצרות בקשת sql שמשנות את תוכן מסד הנתונים.

לשתי התכונות הללו יש פעולות קובעות ומאחזרות משלהן. delegate - סוג משתנה המייצג הפניה לפונקציה עם חתימה מסוימת. יצירת delegate נועדה כדי לאפשר לשמור הפניות לפונקציה בעלות חתימה מסוימת. ניתן להשתמש ב-delegate, על מנת להעביר מצביע למימוש של פעולה שיוצרת שאילתת SQL מאובייקט מסוג BaseEntity מסויים, ולאחר מכן, ניתן להשתמש באותה פעולה כאשר ה-delegate מחזיק מצביע לפעולה.

### מחלקת BaseDB:

מחלקת BaseDB היא מחלקת הבסיס שבעזרתה המשתמש ניגש אל מסד הנתונים, המחלקות שיוורשות ממחלקה זו ממלאות פעולות הכנסה והוצאת מידע המותאמות לסוג המידע של המחלקה, ולפעולות שניתן לעשות לכל סוג מידע. המחלקה הינה תבנית שחוזרת בין כל המחלקות היוורשות.



```
//Properties of BaseDB;
protected string connectionString;
protected OleDbConnection connection;
protected OleDbCommand command;
protected OleDbDataReader reader;
protected List<ChangeEntity> inserted = new List<ChangeEntity>();
protected List<ChangeEntity> deleted = new List<ChangeEntity>();
protected List<ChangeEntity> updated = new List<ChangeEntity>();
```

connectionString - כתובת המסד, שבעזרתו ניתן לגשת למסדי הנתונים  
connection - החיבור למסד נתונים  
command - שאילתא שאנו שולחים למסד נתונים  
reader - קורא את התוצאות של כל שאילתא

שלוש רשימות מסוג ChangeEntity כאשר ChangeEntity בעל שני תכונות, תכונה שמחזיקה אובייקט מסוג BaseEntity, אשר מכיל מידע שצריך להוסיף/לעדכן/לשנות, ותכונה מסוג delegate של CreateSQL עבור הפעולה שיוצרת שאילתת sql לאובייקט.  
inserted - כל האובייקטים מסוג ChangeEntity שנועדו להוסיף מידע חדש.  
deleted - כל האובייקטים מסוג ChangeEntity שנועדו למחוק מידע.  
updated - כל האובייקטים מסוג ChangeEntity שנועדו לעדכן מידע.

```
protected abstract BaseEntity NewEntity();
12 references
protected abstract BaseEntity CreateModel(BaseEntity entity);
22 references
public abstract void Insert(BaseEntity entity);
18 references
public abstract void Update(BaseEntity entity);
22 references
public abstract void Delete(BaseEntity entity);

23 references
protected abstract string CreateInsertSQL(BaseEntity entity);
22 references
protected abstract string CreateUpdateSQL(BaseEntity entity);
21 references
protected abstract string CreateDeleteSQL(BaseEntity entity);
```

פעולות שמחלקות יורשות צריכות לממש:

```
//BaseDB Class abstract Methods:
13 references
protected abstract BaseEntity NewEntity();
12 references
protected abstract BaseEntity CreateModel(BaseEntity entity);
22 references
public abstract void Insert(BaseEntity entity);
18 references
public abstract void Update(BaseEntity entity);
22 references
public abstract void Delete(BaseEntity entity);

23 references
protected abstract string CreateInsertSQL(BaseEntity entity);
22 references
protected abstract string CreateUpdateSQL(BaseEntity entity);
21 references
protected abstract string CreateDeleteSQL(BaseEntity entity);
```

New Entity - פעולה אשר יוצאת מופע ריק של האובייקט של סוג המידע שאנו רוצים להוציא מן המסד, לדוגמא מחלקת UserDB היורשת ממחלקת הבסיס תוציא תחזיר אובייקט של User.

CreateModel - הפעולה מחזירה את המופע של אובייקט אותה היא קיבלה כקלט, ומחזירה אותו כאשר השדות בו מלאים במידע שנקרא ממסד הנתונים. הפעולה קוראת ממסד הנתונים מידע וממלאת את המופע של האובייקט במידע.

Insert - הפעולה מכניסה ChangeEntity לתוך הרשימה inserted ואחראית על הוספת מידע.

Update - הפעולה מכניסה ChangeEntity לתוך הרשימה updated ואחראית על עדכון מידע.

Delete - הפעולה מכניסה ChangeEntity לתוך הרשימה deleted ואחראית על מחיקת מידע.

CreateInsertSQL - הפעולה מקבלת את מופע האובייקט שאותו רוצים להוסיף ויוצרת שאילתת הוספת מידע על העצם שקיבלה.

CreateUpdateSQL - הפעולה מקבלת מופע של אובייקט אותו רוצים לעדכן, ויוצרת שאילתת עדכון עבור אותו עצם.

CreateDeleteSQL - הפעולה מקבלת מופע של אובייקט אותו אנו רוצים למחוק מן המסד, ויוצרת שאילתת מחיקה עבור אותו עצם.

בנאי המחלקה:



```
public BaseDB()
{
    connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\
    connection = new OleDbConnection(connectionString);
    command = new OleDbCommand();
}
```

```
public int SaveChanges()
{
    int records = 0;
    try
    {
        command.Connection = this.connection; //add connection reference to query
        connection.Open(); //open connection
        foreach (var item in updated)
        {
            command.CommandText = item.CreateSql(item.Entity); //receive string of the query
            records = command.ExecuteNonQuery(); // execute update query
        }
    }
```

```
        foreach (var item in inserted)
        {
            command.CommandText = item.CreateSql(item.Entity); //receive string of the query
            records = command.ExecuteNonQuery(); // execute update query
            if (item.Entity is User)
            {
                User user = (User)item.Entity;
                command.CommandText = "Select @@Identity"; //if the entity is user, get it's identity property
                user.ID = (int)command.ExecuteScalar(); // receive and add property
            }
        }
        foreach (var item in deleted)
        {
            command.CommandText = item.CreateSql(item.Entity); //receive string of the query
            records = command.ExecuteNonQuery(); // execute update query
        }
    }
```

```
catch (Exception e)
{
    System.Diagnostics.Debug.WriteLine(e.Message + "\nSQL:" + command.CommandText); //show error if occurred
}
finally
{
    //clear queries lists:
    inserted.Clear();
    updated.Clear();
    deleted.Clear();
    //close connection:
    if (connection.State == System.Data.ConnectionState.Open)
        connection.Close();
}
return records;
```

בתוך המחלקה BaseDB ישנה פעולה בשם SaveChanges ששומרת את המידע החדש שהוזן, מבצעת את כל השאילתות של השינויים במסדי הנתונים. הפעולה מחזירה את הID של עצמים חדשים מסוימים שנכנסים למסד נתונים.

```
protected List<BaseEntity> Select()
{
    List<BaseEntity> list = new List<BaseEntity>(); // create list to store the results of the query
    try
    {
        command.Connection = connection; //adds the database connection to the command
        //command text
        connection.Open(); //opens the connection
        reader = command.ExecuteReader(); //executes the query
        while (reader.Read()) //for each result:
        {
            BaseEntity entity = NewEntity(); //create the entity to store the information.
            list.Add(CreateModel(entity)); // store the result in the entity created.
        }
    }
    catch (Exception e)
    {
        //show the error in case of error
        System.Diagnostics.Debug.WriteLine(e.Message + "\nSQL: " + command.CommandText);
    }
    finally
    {
        //close the reader and the connection in the end.
        if (reader != null)
            reader.Close();
        if (connection.State == System.Data.ConnectionState.Open)
            connection.Close();
    }
    return list;
}
```

Select הינה פעולה בbaseDB שמבצעת שאילתה לקבלת מידע מן מסד הנתונים, ומחזירה את התוצאות של השאילתה. הפעולה שולחת את השאילתה למסד הנתונים, קוראת את המידע המתקבל ממסד הנתונים, ומחזירה את המידע למשתמש.

פעולות נוספות:

:RemoveNonAlphanumeric

```
/// <summary>
/// Removes all non-alphanumeric characters from the input string.
/// </summary>
/// <param name="input">The string to remove non-alphanumeric characters from.</param>
/// <returns>A new string that contains only alphanumeric characters.</returns>
0 references
public static string RemoveNonAlphanumeric(string input)
{
    // Define a regular expression that matches all non-alphanumeric characters
    Regex regex = new Regex("[^a-zA-Z0-9]");

    // Use the regular expression to remove all non-alphanumeric characters from the input string
    string result = regex.Replace(input, "");

    return result;
}
```

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
RemoveNonAlphanumeric פעולה שמורידה כל תו שאינו מספר או אות במחרוזת על מנת להבטיח כי אינם יופיעו בשדות שלא אמורים להיות תווים אחרים כמו שם משתמש, שם פרטי, ושם משפחה.

:ToSqlStr

```
/// <summary>
/// Sanitizes the input string for safe use in database queries to prevent SQL injection attacks.
/// </summary>
/// <param name="original">The string to sanitize.</param>
/// <returns>A sanitized string that can be safely used in database queries.</returns>

17 references
protected string ToSqlStr(string original)
{
    StringBuilder builder = new StringBuilder(original);
    builder.Replace("'", "''");
    builder.Replace("\\", "\\");

    return builder.ToString();
}
```

הפעולה ToSqlStr הופכת מחרוזת שיש מרכאות ' או צירוף של "\" ומכפילה אותם, במידה ויש, על מנת למנוע פריצה על ידי החדרת קוד sql לשדה טקסט. פריצה בעזרת הזנת מידע המכיל מרחאות ומשבש את השאילתא נקראת Sql Injection.

מימוש MessageDB:

```

public class DMessageDB : BaseDB
{
    22 references
    public override void Delete(BaseEntity entity)
    {
        DMessage message = entity as DMessage;
        if (message != null)
        {
            deleted.Add(new ChangeEntity(this.CreateDeletesSQL, entity));
        }
    }

    23 references
    public override void Insert(BaseEntity entity)
    {
        DMessage message = entity as DMessage;
        message.Seen = false;
        if (message != null)
        {
            inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
        }
    }

    18 references
    public override void Update(BaseEntity entity)
    {
        DMessage message = entity as DMessage;
        if (message != null)
        {
            updated.Add(new ChangeEntity(this.CreateUpdatesSQL, entity));
        }
    }
}

```

הוספה, מחיקה, ועדכון מידע. הפעולות מכניסות את ChangeEntity שמורכב מהאובייקט שאותו אנו רוצים להוסיף/למחוק/לעדכן ביחד עם delegate של הפעולה שיוצרת את שאילתת ה-sql עבור אותו אובייקט.

21 references

protected override string CreateDeleteSQL(BaseEntity entity)

```
{
    DMessage message = entity as DMessage;
    StringBuilder sql_builder = new StringBuilder();
    sql_builder.AppendFormat("DELETE FROM DMessageTbl WHERE ID={0}", message.ID);
    return sql_builder.ToString();
}
```

23 references

protected override string CreateInsertSQL(BaseEntity entity)

```
{
    DMessage message = entity as DMessage;
    return string.Format("INSERT INTO DMessageTbl ([SourceUser], [DestUser], " +
        "[Content], [Time], [Seen]) VALUES( {0}, {1}, '{2}', '{3}', {4})",
        message.SourceUser.ID, message.DestUser.ID, ToSqlStr(message.Content),
        DateTime.Now.ToString(), false.ToString());
}
```

22 references

protected override string CreateUpdateSQL(BaseEntity entity)

```
{
    DMessage message = entity as DMessage;
    return $"UPDATE DMessageTbl SET [Content]='{message.Content}', [Seen]='{message.Seen}' " +
        $" WHERE ID={message.ID}";
}
```

פעולות שיוצרות שאילתות sql

פעולות אלו יוצרות שאילתות sql לפי התכונות של האובייקט.

מחיקה - לפי ID של האובייקט. מוחק את הרשומה בעלת ID זהה.

DELETE FROM [tablename] WHERE [תנאי עבור הרשומה]

לדוגמא DELETE FROM DMessageTbl WHERE ID=1 מוחקת מטבלה

DMessageTbl את הרשומה בעלת מספר מזהה 1.

הוספה - מוסיף את כל השדות חוץ מהID שערכו מספר שנקבע ע"י מספור אוטומטי.

INSERT INTO [tablename] (ערכים) VALUES(שדות)

לדוגמא

INSERT INTO DMessageTbl ([SourceUser], [DestUser], " +  
" [Content], [Time], [Seen]) VALUES(1, 2, 'hello', '01/01/2024  
00:00:00', False)

שולחת ממשתמש בעל ID=1 למשתמש בעל ID=2 הודעה בעלת התוכן 'hello' בתאריך

01/01/2024 כאשר ההודעה כמובן תסומן כ"לא נשלחה" כי המשתמש השני עדיין לא קיבל

אותה מאחר וההודעה רק התווספה למסד הנתונים.

עדכון- מעדכן שדות נבחרים של הרשומה על פי תנאי שבמקרה הזה ID = למספר המזהה

עם ההודעה.

UPDATE [tablename] SET feild = value, stringfeild = 'value'...

UPDATE DMessageTbl SET [Content]='what's up?', [Seen]=True" +  
\$" WHERE ID=4

משנה את התוכן של ההודעה ל"what's up" ומעדכנת את הסטטוס "ההודעה התקבלה" לאמת.

יצירת מופע של אובייקט:

```
12 references
protected override BaseEntity NewEntity()
{
    return new DMessage();
}
```

CreateModel קריאת רשומות שהתקבלו מהמסד נתונים.  
SelectByID - המרת המספר המייצג את הרשומה המקושרת לאובייקט של אותו אובייקט.

```
12 references
protected override BaseEntity CreateModel(BaseEntity entity)
{
    UserDB userDB = new UserDB();
    DMessage message = entity as DMessage;
    message.ID = (int)reader["ID"];
    message.DestUser = userDB.SelectByID((int)reader["DestUser"]);
    message.SourceUser = userDB.SelectByID((int)reader["SourceUser"]);
    message.Content = (string)reader["Content"];
    message.Seen = (bool)reader["Seen"];
    message.Time = (DateTime)reader["Time"];
    return message;
}
```

בחירת רשומה קיימת לפי מספר מזהה של אותו אובייקט:

```
0 references
public DMessage SelectMessage(int ID)
{
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE [ID] = {ID}";
    DMessageList list = new DMessageList(base.Select()); //recive results
    return list.Count > 0 ? list[0] : null; // return chosen value by condition if true return list[0] else return null
}
```

תנאים WHERE [שם הטבלה] FROM [שדות נבחרים, כוכבית=כל השדות] Select  
מבצע את הפעולה select במחלקת הבסיס BaseEntity שמריצה את השאילתה, עבור כל  
רשומה אותה פעולה מריצה את CreateModel של המחלקה הנוכחית, ומחזירה את  
התוצאות.

ניתן גם לראות שימוש בקיצור של תנאי if else

condition ? value\_if\_true : value\_if\_false



אם התנאי אמת, הערך המוחזר הוא value\_if\_true, אחרת מוחזר הערך value\_if\_false  
לדוגמא:

list.Count > 0 ? list[0] : null;

תחזיר list[0] אם התנאי list.Count > 0, אחרת, הפעולה תחזיר null.

פעולות יחודיות:

```
2 references
private void UpdateSeen(User main, User chattedU)
{
    EntityList pair = new EntityList();
    pair.Entities.Add(chattedU);
    pair.Entities.Add(main);
    updated.Add(new ChangeEntity(this.CreateUpdateSeenSQL, pair));
    this.SaveChanges();
}
```

הפעולה UpdateSeen מעדכנת את כל ההודעות שהמשתמש קיבל "נראו", על מנת לסמן שקיבל אותן.

הפעולה יוצרת EntityList בו היא מאחסנת את שני המשתמשים, אחד שהוא המשתמש הנוכחי והשני שהוא משתמש היעד, ומכניסה אותם ביחד עם הפעולה שיוצרת שאילתה עדכון נראה לאחרונה ChangeEntity לתוך updated.  
לאחר מכן מורצת הפעולה SaveChanges.  
שימוש בפעולה זו הוא בקבלת הודעות שלא נראו או קבלת כל ההודעות בהתכבות מסויימת.  
בשני המקרים על השרת לעדכן שהמשתמש השני קיבל את כל ההודעות.

הפעולה CreateUpdateSeenSql יוצרת שאילתת sql שמעדכנת את כל ההודעות

```
1 reference
protected string CreateUpdateSeenSQL(BaseEntity entity)
{
    EntityList entityList = entity as EntityList;
    User source = (User)entityList.Entities[0];
    User dest = (User)entityList.Entities[1];
    return $"UPDATE DMessageTbl SET [Seen]= True WHERE [DestUser]= {dest.ID} AND [SourceUser] = {source.ID}";
}
```

שנשלחות למשתמש עד כה "נראו". הפעולה מקבלת EntityList שמכיל את משתמש שכתב את ההודעות ומשתמש היעד.

```
1 reference
public DMessageList SelectUnSeen(User main, User chattedU)
{
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID} AND [Seen] = False) ";
    DMessageList dMessages = new DMessageList(base.Select());
    UpdateSeen(main, chattedU);
    foreach (DMessage message in dMessages)
    {
        message.Seen = true;
    }
    return dMessages;
}
```

הפעולה SelectUnSeen מחזירה את כל ההודעות שלא נראו על ידי משתמש היעד. הפעולה מסמנת את אותן הודעות כ"נראו" בעזרת הפעולה UpdateSeen לפני שהיא שולחת אותם אל משתמש היעד.

```
1 reference
public DMessageList SelectAllDMsChat(User main, User chattedU)
{
    UpdateSeen(main, chattedU);
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID}) " +
        $"OR ([DestUser] = {chattedU.ID} AND [SourceUser] = {main.ID})";
    return (new DMessageList(base.Select()));
}
```

הפעולה SelectAllDMsChat מחזירה את כל ההודעות בתוך הצ'אט ומסמנת את ההודעות של המשתמש השני כ"נראו". על מנת לקבל את כל ההודעות, השאילתה בוחרת בכל ההודעות בהן: משתמש היעד הוא main ומשתמש הכותב הוא chattedU או משתמש היעד הוא chattedU ומשתמש הכותב הוא main.

```
1 reference
public bool AreChatting(User main, User chattedU)
{
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID}) " +
        $"OR ([DestUser] = {chattedU.ID} AND [SourceUser] = {main.ID}) LIMIT 1";
    return (new DMessageList(base.Select())).Count > 0;
}
```

הפעולה AreChatting בודקת אם שני משתמשים התכתבו עד כה.



הפעולה מקבלת את המשתמש הנוכחי ומשתמש השני ומחזירה אמת אם המשתמשים התכתבו, אחרת שקר. היא קוראת ממסד הנתונים לכל היותר שורה אחת של הודעה אחת על מנת לבדוק אם יש לפחות הודעה אחת על מנת לקבוע אם המשתמשים התכתבו. 1 LIMIT מגבילה את התשובה של השאילתה לרשומה אחת על מנת לקצר את זמן הבדיקה. הפעולה מקבלת הודעה אחת לכל היותר מהתכתבות בין שני משתמשים, ואם התקבלה הודעה אחת, הפעולה תחזיר אמת מאחר ואחד המשתמשים כבר רשם למשתמש השני הודעה. אחרת שקר.

```

/// <summary>
/// the method gets the current user and returns all the chats
/// with that user.
/// </summary>
/// <param name="user"></param>
/// <returns>User's list</returns>
1 reference
public UserList GetAllDMChats(User user)
{
    DMessageList sourceUsers = null;
    DMessageList destUsers = null;
    UserList users = new UserList();
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE [DestUser] = {user.ID}"; //get all Messages that the main user recived sql
    sourceUsers = new DMessageList(base.Select());
    command.CommandText = "SELECT * FROM DMessageTbl" +
        $" WHERE [SourceUser] = {user.ID}"; //get all Messages that the main user sent sql
    destUsers = new DMessageList(base.Select());
    foreach (DMessage userS in sourceUsers) //for each user that the main user recived
        if (!(IsUserInUist(users, userS.SourceUser))) //if the second user is not in list
            users.Add(userS.SourceUser); //add the user
    foreach (DMessage userD in destUsers) //for each user that the main user recived
        if (!(IsUserInUist(users, userD.DestUser))) //if the second user is not in list
            users.Add(userD.DestUser); //add the user
    return users;
}

```

הפעולה GetAllDMChats מחזירה את כל ההתכתבויות של משתמש מסויים עם אנשים אחרים. על מנת למצוא את כל המשתמשים איתם המשתמש הרצוי התכתב צריך לקבל את כל ההודעות שהמשתמש הרצוי הוא הכותב, ולהכניס אותם לlist אם אותו משתמש שני, היעד, אינו כבר נמצא ברשימה. לאחר מכן הפעולה מקבלת את כל ההודעות בהן משתמש היעד הוא המשתמש הרצוי, ומוסיפה את משתמש המקור במידה והוא כבר לא נמצא ברשימה של האנשים איתם המשתמש מתכתב. לאחר מכן מוחזרת הרשימה של כל המשתמשים איתם המשתמש הרצוי user מתכתב.

```
2 references
private bool IsUserInUist(UserList list, User userToAdd)
{ //the method gets UserList and user wanted to add and returns true if the user is already in the list
  // else false
  foreach (User userDM in list)
  {
    if (userToAdd.ID == userDM.ID )
      return true;
  }
  return false;
}
```

הפעולה בודקת אם userToAdd נמצא בlist. אם אכן נמצא, הפעולה תחזיר אמת, אחרת שקר.

```
//returns a value that symbolizes the state of all chats
//with that user the purpose is to check this update state value
//to know if the the chats list should be refreshed
1 reference
public int GetChatsDMUpdateState(User user)
{
  UserList users = GetAllDMChats(user);
  int checksum = 0, count = 0;
  foreach (User userTemp in users)
  {
    checksum += (userTemp.ID * (int)userTemp.Uname[0]);
    count++;
  }
  checksum *= count;
  return checksum;
}
```

פעולה זו מחזירה מספר המסמל את מצב הצ'אטים של המשתמש. המספר הזה נועד כדי לבדוק אם רשימת הצ'אטים השתנתה (המספר השתנה) ואם כן, הצד לקוח ירענן את רשימת הצ'אטים. אנו לא רוצים לשלוח למשתמש כל פעם מחדש את כל הצ'אטים כדי לעדכן את רשימת המשתמשים על מנת לחסוך זמן ריצה בצד האפליקציה של צד הלקוח.

UserDB:

```
18 references
public class UserDB : BaseEntity
{
  12 references
  protected override BaseEntity NewEntity()
  {
    return new User() as BaseEntity;
  }
}
```

הפעולה מחזירה מופע ריק של User כ-BaseEntity.

12 references

```
protected override BaseEntity CreateModel(BaseEntity entity)
{
    User user = entity as User;
    user.ID = (int)reader["ID"];
    user.Fname = (string)reader["Fname"];
    user.Lname = (string)reader["Lname"];
    user.Uname = (string)reader["Uname"];
    user.Description = (string)reader["Description"];
    if (user.Description == null)//if user doesn't have description
        user.Description = "";
    user.EntrDate = (DateTime)reader["EntrDate"];
    user.Password = (string)reader["Password"];
    user.Email = (string)reader["Email"];
    return user;
}
```

הפעולה קוראת מרשומה המתקבלת בתשובה ממסד הנתונים. הפעולה ממלאת מופע ריק של העצם, ומחזירה אותו בחזרה עם כל המידע שהתקבל ממסד הנתונים. המידה ומסד הנתונים החזיר ערך null במקום string ריק, הפעולה תכניס מחרוזת ריקה.

23 references

```
public override void Insert(BaseEntity entity)
{
    User people = entity as User;
    if (people != null)
    {
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
    }
}
```

18 references

```
public override void Update(BaseEntity entity)
{
    User people = entity as User;
    if (people != null)
    {
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
    }
}
```

22 references

```
public override void Delete(BaseEntity entity)
{
    User people = entity as User;
    if (people != null)
    {
        deleted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
    }
}
```

```

23 references
protected override string CreateInsertSQL(BaseEntity entity)
{
    User user = entity as User;
    return string.Format("INSERT INTO UserTbl (Uname, FName, LName, EntrDate, [Password], Description, Email)" +
        " VALUES('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}')" +
        RemoveNonAlphanumeric(user.Uname), RemoveNonAlphanumeric(user.FName), RemoveNonAlphanumeric(user.LName),
        user.EntrDate.ToString(), ToSqlStr(user.Password), ToSqlStr(user.Description), ToSqlStr(user.Email));
}

21 references
protected override string CreateDeleteSQL(BaseEntity entity)
{
    User user = entity as User;
    StringBuilder sql_builder = new StringBuilder();
    sql_builder.AppendFormat("DELETE FROM UserTbl WHERE [Uname]={0} AND [Password] = {1}", ToSqlStr(user.Uname), ToSqlStr(user.Password));
    return sql_builder.ToString();
}

22 references
protected override string CreateUpdateSQL(BaseEntity entity)
{
    User user = entity as User;
    return $"UPDATE UserTbl SET Uname='{RemoveNonAlphanumeric(user.Uname)}', FName='{RemoveNonAlphanumeric(user.FName)}', " +
        $" LName='{RemoveNonAlphanumeric(user.LName)}', " +
        $" Description='{ToSqlStr(user.Description)}', [Password]='{ToSqlStr(user.Password)}' WHERE ID={user.ID}";
}

0 references
public UserList SelectAll()

```

עבור כל מחרוזת שבעזרתה ניתן להחדיר קוד SQL לבקשה, שיניתי באותה מחרוזת את כל  
התווים הבעייתיים בעזרת הפעולות RemoveNonAlphanumeric, ToSqlStr

```

1 reference
public UserList SelectUserData(string Uname, string Password)
{
    command.CommandText = string.Format("SELECT * FROM UserTbl WHERE Uname = '{0}' AND [Password] = '{1}' ", ToSqlStr(Uname), ToSqlStr(Password));
    return new UserList(base.Select());
}

```

הפעולה selectUserData הינה ההתחברות עצמה למשתמש, הפעולה הזו מחזירה את  
המשתמש כולל הסיסמא בניגוד לבקשות SQL אחרות של UserDB

```
1 reference
public UserList SelectByUName(string Uname) //select all usernames starts with the Uname
{
    command.CommandText = string.Format("SELECT * FROM UserTbl WHERE [Uname] LIKE '{0}%' ", ToSqlStr(Uname));
    UserList userL = new UserList(base.Select());
    foreach (User user in userL)//for each user remove sensitive information.
    {
        user.Password = "";
        user.Email = "";
    }
    return userL;
}

2 references
public UserList SelectUNameExist(string Uname) //select the User with the exact UserName
{
    command.CommandText = string.Format("SELECT * FROM UserTbl WHERE [Uname] = '{0}' ", ToSqlStr(Uname));
    UserList userL = new UserList(base.Select());
    foreach (User user in userL)//for each user remove sensitive information.
    {
        user.Password = "";
        user.Email = "";
    }
    return userL;
}
```

SelectByUName - חיפוש משתמשים על פי תווים ראשונים של השם משתמש.  
SelectUNameExist - מחפש משתמש על פי שם המשתמש.  
שני הפעולות מוחקות תוצאות שהתקבלו מהשדות Password ו-Email מאחר וזה מידע שאנו לא רוצים שיעבור לצד לקוח, מאחר ואנחנו לא רוצים שיהיה לו מידע אישי על משתמש אחר מסויים.

בחירת משתמש על פי מספר מזהה:

```
12 references
public User SelectByID(int ID) //select user by id
{
    command.CommandText = string.Format("SELECT * FROM UserTbl WHERE ID = {0} ", ID);
    UserList userL = new UserList(base.Select());
    foreach (User user in userL)//for each user remove sensitive information.
    {
        user.Password = "";
        user.Email = "";
    }
    return userL.Count > 0 ? userL[0] : null;
}
```

הפעולה מחזירה User רק אם התקבל User שתואם את הID. במידה ובמסד נתונים אין משתמש כזה, הפעולה תחזיר null.

UnSeenGMDB: מסד נתונים שנועד לסמן אנשים שלא קיבלו עדיין הודעות שנשלחו בקבוצה, במטרה לדעת אילו הודעות יש לשלוח לאותם משתמשים בזמן אמת, כאשר ההתכתבות פתוחה. שליחת כל ההודעות כל הזמן לשם עדכון יכולה להעמיס על הצ'אט.

```
23 references
public override void Insert(BaseEntity entity)
{
    //DataBases needed for the insert
    GMemberDB gMemberDB = new GMemberDB();
    UnSeenGM unSeen = entity as UnSeenGM;
    GMessageDB gMessageDB = new GMessageDB();
    if (unSeen != null)
    {
        //get the group in order to make sure it exists and correct.
        Group group = gMessageDB.SelectGMessageByID(unSeen.Message.ID).Group;
        unSeen.Group = group.ID; //add the group id to the unSeen object
        List<User> users = gMemberDB.GetUserMembers(group); //get all the members if the group
        foreach (User user in users)
        {
            //if the user isn't the writer of the message, add
            //an UnSeen for this user in order to make sure what messages the user
            //hasn't received yet.
            if (user.ID != unSeen.User)
            {
                UnSeenGM temp = new UnSeenGM(unSeen.Message, user.ID, group.ID);
                inserted.Add(new ChangeEntity(this.CreateInsertSQL, temp));
            }
        }
    }
}
```

הפעולה מכניסה UnSeen להודעה מסויימת. הפעולה מוסיפה למסד נתונים UnSeen עבור כל משתמש שאינו המחבר של ההודעה.

```
/// <summary>
/// the method deletes an UnSeen entity from the UnSeenGMTbl Database
/// in order to mark that the user saw the message
/// </summary>
/// <param name="entity"></param>
22 references
public override void Delete(BaseEntity entity)
{
    UnSeenGM unSeen = entity as UnSeenGM;
    if (unSeen != null)
    {
        deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
    }
}
```

הפעולה מוחקת רשומה של UnSeen על מנת לסמן כי הודעה נשלחה למשתמש מסויים, לכן ניתן למחוק את הרשומה.



```

/// <summary>
/// the method gets all the messages that the user in the group hasn't seen yet
/// </summary>
/// <param name="user"> the user that want to receive unseen messages</param>
/// <param name="group">the group that the user in</param>
/// <returns>list of GMessages that the user hasn't received yet</returns>
1 reference
public List<GMessage> GetChatsUnSeenMessages(User user, Group group)
{
    command.CommandText = "SELECT * FROM UnSeenGMTbl" +
        $" WHERE [Group] = {group.ID} AND [USER] = {user.ID}" ;
    List<GMessage> messages =(new UnSeenGMList(base.Select())).Select(unseen => unseen.Message).ToList();
    Delete(new UnSeenGM(null, user.ID, group.ID)); //delete all messages that the user receives from unseen database
    SaveChanges();//save the changes in the group unseen messages database
    return messages;
}

```

הפעולה מחזירה את כל ההודעות שמשתמש מסויים בקבוצה מסוימת לא קיבל. הפעולה לוקחת ממסד הנתונים את כל הרשומות שבהן סומן שהמשתמש מסויים לא ראה הודעה בקבוצה, ממירה את כל אלו להודעות על פי השדה Message בעזרת הפעולה Select שמחזירה מערך על פי פעולה המחזירה עצם/ערך כלשהו שהיא מקבלת. היא יוצרת ערכים על פי הפעולה שהיא מקבלת. יש שימוש lambda כדי להגדיר פעולה המחזירה את התכונה Message. לאחר הרשימה של כל ההודעות ש-Select מחזירה ממירים ל-List.

```

// <summary>
// makes sql string that inserts a message to an unseen group messages database
// in order to mark a message as unseen by user
// </summary>
// <param name="entity"></param>
// <returns></returns>
3 references
protected override string CreateInsertSQL(BaseEntity entity)

    UnSeenGM unSeen = entity as UnSeenGM;
    return string.Format("INSERT INTO UnSeenGMTbl ([User], [Message], [Group]) VALUES( {0}, {1}, {2} )",
        unSeen.User, unSeen.Message.ID, unSeen.Group);

```

הפעולה יוצרת שאלתת הוספה לUnSeen

```
/// <summary>
/// receives data from database about unseen message by user
/// </summary>
/// <param name="entity">An empty UnSeenGM object pointer that the method fills </param>
/// <returns>filled recieved entity entity</returns>
12 references
protected override BaseEntity CreateModel(BaseEntity entity)
{
    UnSeenGM unSeen = entity as UnSeenGM;
    GMessageDB gMessageDB = new GMessageDB();
    unSeen.Message = gMessageDB.SelectGMessageByID((int)reader["Message"]);
    unSeen.User = (int)reader["User"];
    unSeen.Group = (int)reader["Group"];
    return unSeen;
}
```

הפעולה ממלאת את העצם הריק בתוצאות המתקבלות ממסד הנתונים.

namespace ModelView

```
{
    public class GMessageDB : BaseDB
    {
        /// <summary>
        /// deletes a group message
        /// </summary>
        /// <param name="entity">group message to delete</param>
        public override void Delete(BaseEntity entity)
        {
            GMessage message = entity as GMessage;
            if (message != null)
            {
                deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
            }
        }

        /// <summary>
        /// adds a group message, also marks the message as unseen for each member except
        /// the author
        /// </summary>
        /// <param name="entity">the GMessage to add to DataBase</param>
        public override void Insert(BaseEntity entity)
        {
            GMessage message = entity as GMessage;
            UnSeenGMDB unSeenGMDB = new UnSeenGMDB();
            if (message != null)
            {
                inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
                SaveChanges();
                unSeenGMDB.Insert(new UnSeenGM(message, 0, message.Group.ID));
                unSeenGMDB.SaveChanges();
            }
        }
    }
}
```



• הסבר על הפעולת הוספה:

הפעולה מקבלת הודעה בקבוצה שרוצים להוסיף למסד הנתונים, היא מכניסה את ההודעה ומקבלת בחזרה את הID של ההודעה שנכנסה. לאחר מכן מוסיפים את ההודעה לUnSeenDB על מנת לסמן כי אך אחד לא קיבל את ההודעה (חוץ מכותב/ת ההודעה כמובן).

```
/// <summary>
/// updates(changes) a group message
/// </summary>
/// <param name="entity">the updated group message</param>
public override void Update(BaseEntity entity)
{
    GMessage message = entity as GMessage;
    if (message != null)
    {
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
    }
}

/// <summary>
/// makes sql to delete a group message
/// </summary>
/// <param name="entity">the GMessage to delete</param>
/// <returns>sql that deletes the message</returns>
protected override string CreateDeleteSQL(BaseEntity entity)
{
    GMessage message = entity as GMessage;
    StringBuilder sql_builder = new StringBuilder();
    sql_builder.AppendFormat("DELETE FROM GMessageTbl WHERE ID= {0}",
message.ID);
    return sql_builder.ToString();
}
```

• הפעולה יוצרת שאילתת SQL שמוחקת הודעה בקבוצה על פי המספר המזהה שלה

```
/// <summary>
/// makes sql to inserts a group message
/// </summary>
/// <param name="entity">the GMessage to insert</param>
/// <returns>sql that inserts the message</returns>
protected override string CreateInsertSQL(BaseEntity entity)
{
    GMessage message = entity as GMessage;
    return string.Format("INSERT INTO GMessageTbl ([User], [Group], [Content])
VALUES( {0}, {1}, '{2}' )",
message.User.ID, message.Group.ID, ToSqlStr(message.Content));
}
```

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il

- הפעולה יוצרת שאילתת sql שמוסיפה את ההודעה לקבוצה על פי שם משתמש, שם הקבוצה ותוכן ההודעה.

```
/// <summary>
/// fills the feilds in the group message with the received information
/// from the database
/// </summary>
/// <param name="entity"></param>
/// <returns>filled entity</returns>
protected override BaseEntity CreateModel(BaseEntity entity)
{
    UserDB userDB = new UserDB();
    GroupDB groupDB = new GroupDB();
    GMessage message = entity as GMessage;
    message.ID = (int)reader["ID"];
    message.User = userDB.SelectByID((int)reader["User"]);
    message.Group = groupDB.SelectGroup((int)reader["Group"]);
    message.Content = (string)reader["Content"];
    return message;
}
```

- הפעולה ממלאת מופע של הודעה בעצם חדש. על פי המספר המזהה המתקבל של User ו-Group ממסד הנתונים ניתן למצוא את Usern וה-Group.

```
/// <summary>
/// makes sql to updates a group message
/// </summary>
/// <param name="entity">the GMessage to update</param>
/// <returns>sql that updates the message</returns>
protected override string CreateUpdateSQL(BaseEntity entity)
{
    GMessage message = entity as GMessage;
    return $"UPDATE GMessageTbl SET [Content]='{ToSqlStr(message.Content)}'
WHERE [ID]={message.ID}";
}
```

- פעולה שיוצרת שאילתת SQL של עדכון הודעה. ניתן לעדכן רק את תוכן ההודעה.

```
//creates an empty GMessage object
protected override BaseEntity NewEntity()
{
    return new GMessage() as BaseEntity;
}
```

```
/// <summary>
/// returns the Gmessage with the ID given
/// </summary>
/// <param name="ID">id of wanted message</param>
/// <returns>the group message</returns>
```

```
public GMessage SelectGMessageByID(int ID)
```

```
{
    command.CommandText = "SELECT * FROM GMessageTbl" +
        $" WHERE [ID] = {ID}";
    GMessageList list = new GMessageList(base.Select());
    return list.Count > 0 ? list[0] : null;
}
```

- הפעולה מחזירה הודעה אם התקבלה רשומה ממסד הנתונים בעלת ID רצוי, אחרת null.

```
/// <summary>
```

```
/// receives all the messages in a group. also marks the messages as seen
```

```
/// </summary>
```

```
/// <param name="user">the user in the group</param>
```

```
/// <param name="group">the group</param>
```

```
/// <returns>GMessageList of the messages</returns>
```

```
public GMessageList GetMessages(User user, Group group)
```

```
{
    UnSeenGMDB unSeenDB = new UnSeenGMDB();
    unSeenDB.Delete(new UnSeenGM(null, user.ID, group.ID));
    unSeenDB.SaveChanges();
    command.CommandText = "SELECT * FROM GMessageTbl" +
        $" WHERE [Group] = {group.ID}";
    return new GMessageList(base.Select());
}
```

- הפעולה שולחת את כל ההודעות בהתכתבות קבוצתית. הפעולה תסמן את כל ההודעות שלא נשלחו עד כה למשתמש כ"סומנו" בעזרת UnSeenGMDB.

```
}
}
```

## WCF שירותים שהשרת מציע

WCF היא התשתית לבניית שכבת תקשורת בין מערכות ואפליקציות. ובפרויקט זה, היא תשתית תקשורת של הלקוח, משתמש עם השרת. הטכנולוגיה המאפשרת להגדיר את השירות עצמו, מסייעת בבניית המארח של השירותים (מארח ובבניית צרכים של השירותים (לקוח). השתמשתי ב-WCF מכיוון שתהליך התקשורת בין הלקוח לשרת פשוט יותר ודומה לתהליך התכנות של המשתמש במתודות, טיפוסים וממשקים של השרת. על מנת לשלוח מידע ב-WCF צריך להגדיר מספר דברים: היכן ניתן למצוא את השירותים? איך מתחברים לשירות? מה השירות מספק? כל שלושת הדברים האלו מגדירים EndPoint. השאלות האלו נקראות לפעמים גם כ-ABC של EndPoint: Address - כתובת ה-URL של השירות Binding - התקשורת המתאימה לשירות Contract - מה השירות מציע ומה הפעולות האפשריות בו.

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
SOA היא ארכיטקטורה שמגדירה את העקרונות התכנוניים לפיתוח מערכות תוכנה מבזרות(מפוצלת בין מחשבים). היא מאפשרת למערכות שונות לתקשר באופן אחיד בתשתית משותפת, על פי הצרכים של המשתמשים.

בשירות WCF בתוכנה, יש שילוב בין הטכנולוגיה לארכיטקטורת SOA מאפשר למפתחי התוכנה לבנות מערכות מבזרות המאפשרות לשרתים וללקוחות לתקשר עם זה בצורה יעילה ומותאמת. בארכיטקטורה זו כדי להפעיל אפליקציה מבוססת שירותים.

בפרוייקט זה WCF היא התשתית התקשורת של הלקוח, המשתמש עם השרת. הגדרתי שירותים של השרת שאני רוצה לתת ללקוח - שהינו האפליקציה של המשתמש. הטכנולוגיה מאפשרת להגדיר את השירות עצמו, מסייעת בבניית המארח של השירות (Host) ובבניית הלקוח הצורך את השירותים (Client). השתמשתי ב-WCF מכיוון שהליך התקשורת בין הלקוח לשרת פשוט יותר ודומה לתהליך תכנותי המשתמש במתודות, טיפוסים וממשקים של השרת.

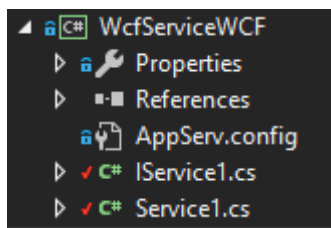
#### בניית שירות:

- ServiceContract מעטר את הממשק עצמו ומכריז שהממשק ישמש כחוזה בין השירות ללקוח.
- OperationContract הוא חלק בלתי נפרד מה- ServiceContract ומגדיר את המתודות שהשירות יחשוף ללקוח.
- כמובן שניתן להגדיר בממשק מתודות שלא יחשפו, פשוט לא נעטר אותם ב- OperationContract.

ServiceContract – מגדיר את הממשק כממשק המשמש כ-Contract

OperationContract – מגדיר את המתודות שיחשפו ללקוח.

פרוייקט WCF Service Library



הגדרת ServiceContract מגדירה את החוזה של כל הפעולות שהשרת מספק, כאשר OperationContract הוא פעולה המופיעה בחוזה.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using Model;
using ModelView;

namespace WcfServiceWCF
{
    [ServiceContract]
    1 reference
    public interface IService1
    {
        [OperationContract]
        1 reference
        string GetData(int value);

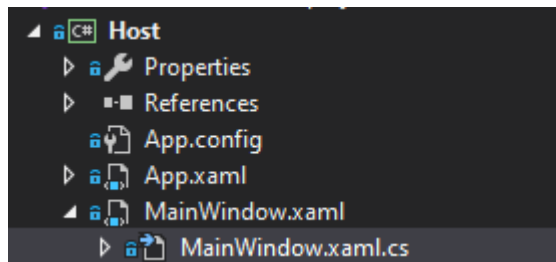
        [OperationContract]
        1 reference
        CompositeType GetDataUsingDataContract(CompositeType composite);

        [OperationContract]
        1 reference
        void DeleteComment(Comment entity);
    }
}
```

אחרי שהוגדר הממשק של כל השירותים של המשתמש, יש להגדיר את המימוש לכל הפעולות הללו

```
1 reference
public class Service1 : IService1
{
    //databases used:
    private CommentDB commentDB = new CommentDB();
    private UserDB userDB = new UserDB();
    private FollowDB followDB = new FollowDB();
    private FriendDB friendDB = new FriendDB();
    private LikeDB likeDB = new LikeDB();
    private PostDB postDB = new PostDB();
    private DMessageDB dMessageDB = new DMessageDB();
    private GroupDB groupDB = new GroupDB();
    private GMemberDB gMemberDB = new GMemberDB();
    private GMessageDB gMessageDB = new GMessageDB();
    private UnSeenGMDB unSeenGMDB = new UnSeenGMDB();
    //operations implementation
    1 reference
    public void DeleteComment(Comment entity)
    {
        commentDB.Delete(entity);
        commentDB.SaveChanges();
    }
    1 reference
    public void InsertComment(Comment entity)
    {
        commentDB.Insert(entity);
    }
}
```

פתיחת השירות בחלון חדש של המשתמש בפרוייקט HOST שהינו WPF.

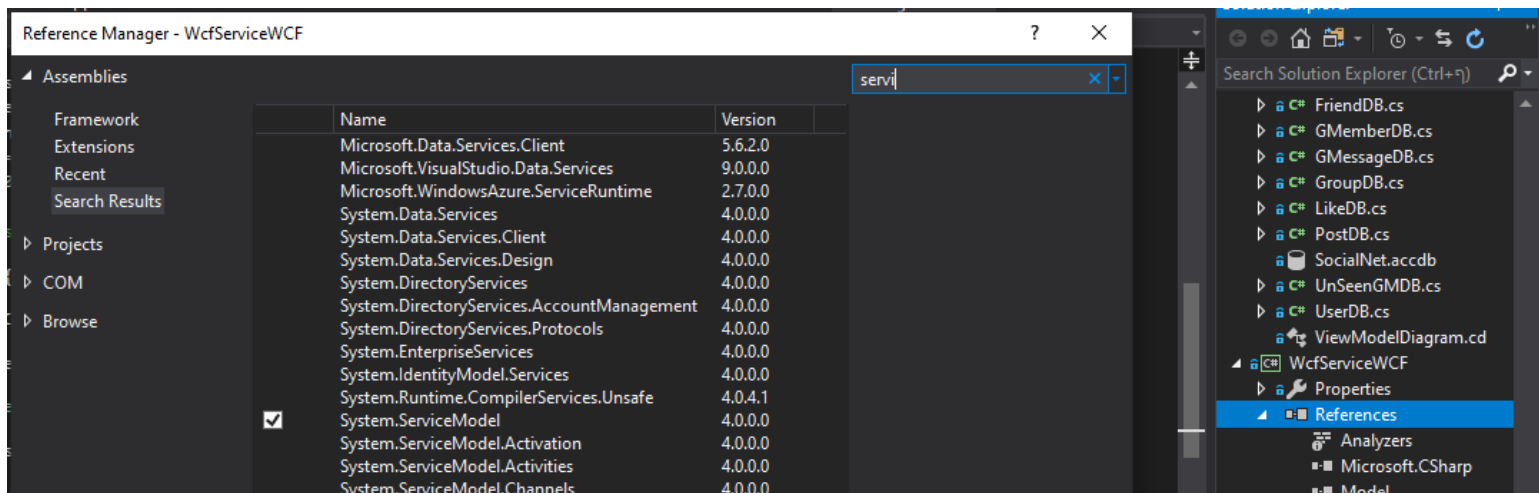


```
2 references
public partial class MainWindow : Window
{
    0 references
    public MainWindow()
    {
        InitializeComponent();
        ServiceHost srv = new ServiceHost(typeof(WcfServiceWCF.Service1));
        srv.Open();
    }
}
```

החלון עצמו ריק והפונקציות היחידה שלו היא להריץ את השרת.

```
<system.serviceModel>
  <services>
    <service name="WcfServiceWCF.Service1">
      <host>
        <baseAddresses>
          <add baseAddress = "http://localhost:8733/Design_Time_Addresses/WcfServiceWCF/Service1/" />
        </baseAddresses>
      </host>
      <!-- Service Endpoints -->
      <!-- Unless fully qualified, address is relative to base address supplied above -->
      <endpoint address="" binding="basicHttpBinding" contract="WcfServiceWCF.IService1">
        <!--
          Upon deployment, the following identity element should be removed or replaced to reflect the
          identity under which the deployed service runs. If removed, WCF will infer an appropriate identity
          automatically.
        -->
        <identity>
          <dns value="localhost"/>
        </identity>
      </endpoint>
      <!-- Metadata Endpoints -->
      <!-- The Metadata Exchange endpoint is used by the service to describe itself to clients. -->
      <!-- This endpoint does not use a secure binding and should be secured or removed before deployment -->
      <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange"/>
    </service>
  </services>
```

הוספת service model לreference של הפרויקט של wcf:





```
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />
  </startup>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IService1" />
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://localhost:8733/Design_Time_Addresses/WcfServiceWCF/Service1/"
        binding="basicHttpBinding" bindingConfiguration="BasicHttpBinding_IService1"
        contract="ServiceReference1.IService1" name="BasicHttpBinding_IService1" />
    </client>
  </system.serviceModel>
</configuration>
```

עבור כל חלון יש לכלול את הספרייה:

```
using ChatStream.ServiceReference1;
```

ועבור כל חלון יש להכיל תכונה של אובייקט שניגש לשירותי השרת.

```
Service1Client srv = new Service1Client();
```

השרת צריך לפעול ברקע כל הזמן על מנת לספר שירותים ללקוח.

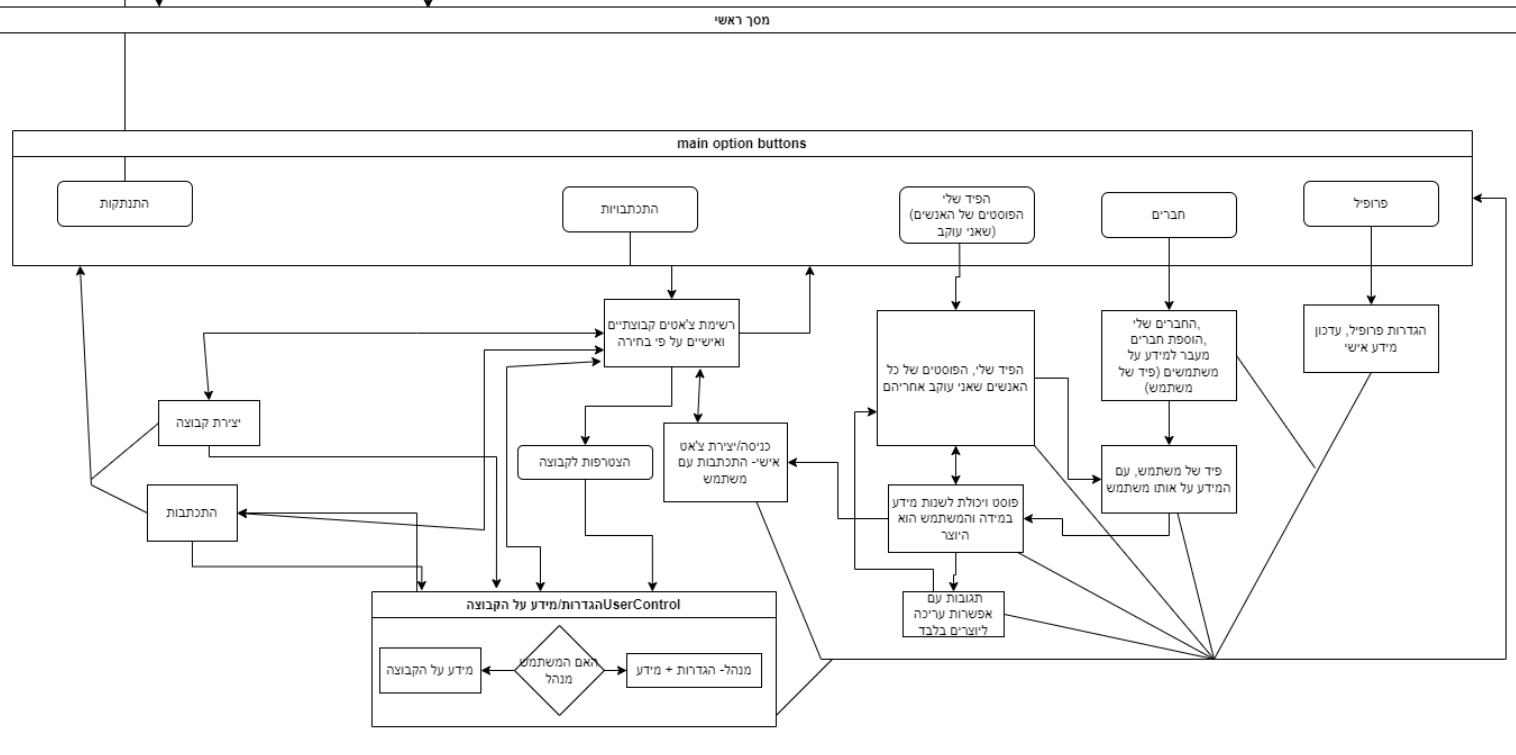
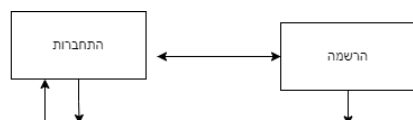


צד לקוח:

צד לקוח הינו התצוגה של כל מידע המתקבל, ביצוע ויזימת פעולות שמוסיפות או משנות מידע בצד שרת. לקוח הוא האפליקציה שהמשתמש רואה, ומבצע פעולות. הצד לקוח שלי מורכב משלושה חלונות, ומספר רב של UserControl. UserControl הוא אובייקט ב-WPF המכיל קוד Xaml וקוד C# ומאפשר להגדיר גרפיקה ותצורה מורכבת של מרכיבי UI (מרכיבי תצוגה). למרות ש-UserControl מכיל אובייקטים כמו חלון רגיל, ההבדל העיקרי הוא ש-UserControl ניתן לשימוש חוזר, בנוסף ניתן להשתמש בו כמרכיב בתוך חלון או בתוך מרכיבי UI אחרים.

Windows and UserControls:





## Windows:

- Login - חלון התחברות למשתמש
- SignUp - חלון הרשמה למשתמש חדש
- MainWindow - החלון הראשי בו מוצגת כל הפעילות של המשתמש

## UserControls:

- OpenMenu - תצוגת פתיחה של החלון הראשי
- Profile - מידע על הפרופיל של המשתמש ושינוי מידע
- FriendListShow - רשימת חברים של המשתמש, ובקשות חברות, עם חלונות של friendShow שבעזרתם ניתן לשנות סטטוס חבר.
- FriendShow - חלונות עם שם המשתמש, הפנייה לפיד ושינוי סטטוס חבר, אישור בקשת חברות עם הסטטוס של החברות הוא לחכות לאישור.
- Feed - כל הפוסטים שהמשתמש מעוניין לראות. הפוסטים של כל המשתמשים שהמשתמש עוקב אחריהם. ניתן לחפש משתמשים ולהיכנס ל-UserFeed של הפוסטים שלהם ובכך להחליט אם לעקוב אחריהם או להיות חברים שלהם.
- UserFeed - חלון כל הפוסטים והמידע על המשתמש. ישנה אופציה לראות את כל הפוסטים שמיועדים לכולם, ואם המשתמשים חברים לראות גם את הפוסטים הפרטיים של אותו משתמש שמיועדים רק לחברים.

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il

**CreatePost** - נועד כדי ליצור פוסטים. המשתמש ממלא תוכן, והפוסט מתפרס ב**UserFeed** שלו וב**Feed** של מי שעוקב אחריו. אם הפוסט פרטי רק חברים של אותו משתמש יכולים לגשת אליו.

**PostView** - תצוגת פוסט. ניתן לעשות לייק לפוסט, להגיב לפוסט וכמובן לקרוא אותו. אם המשתמש הנוכחי הוא היוצר שלו, יש לו את האפשרות גם לערוך אותו, ולקבוע אם לשנות אותו לפרטי (רק חברים) או לציבורי (כולם).

**CommentView** - תצוגה של תגובה לפוסט. מציג את התוכן של הפוסט, יש כפתור מעבר ל**UserFeed** של אותו משתמש יוצר התגובה. כותב/ת התגובה יכול/ה לערוך את התגובה.

**ChatsShow** - תצוגת כל הצ'אטים של המשתמש, בין שני משתמשים או קבוצות, עם אפשרות לחיפוש משתמשים או קבוצות.

ישנה אפשרות להצטרף לקבוצות ציבוריות.

**ChatWithUser** - ההתכתבות בין שני משתמשים, שליחה וקבלת הודעות בזמן אמת תוך שמירה של אותן הודעות

**ChatWithGroup** - התכתבות בתוך קבוצת התכתבות, שליחה וקבלת הודעות בזמן אמת תוך שמירה של אותן הודעות בקבוצה.

**GroupInfo** - חלון המידע על הקבוצה, ישנה אפשרות לערוך מידע של הקבוצה אם המשתמש מנהל/ת.

חלון התחברות: Login

הכנסת שם משתמש חוקי (אותיות אנגליות עם ספרות) וסיסמא על מנת להתחבר.

```

/// <summary>
/// the button click collects the user's password and username, and logs in if the info
/// is valid
/// </summary>
/// <param name="sender">button</param>
/// <param name="e"></param>
1 reference
private void EnterB(object sender, RoutedEventArgs e)

```

```
private void EnterB(object sender, RoutedEventArgs e)
{
    User user = null;
    if(UName.Text.Length >= 3 && UPassword.Password.Length >= 4)
    {
        if(!Regex.IsMatch(UName.Text, "^[a-zA-Z0-9]*$")) //condition checks if username is numbers and letters only.
        {
            MessageBox.Show("User name should contain only letters and numbers");
        }
        else
        {
            //tries to log in:
            user = srv.Login(UName.Text, UPassword.Password);
            if (user != null && user.ID > 0)
            {
                //successful login
                MainWindow main = new MainWindow(user);
                main.Show();
                this.Close();
            }
            else
            {
                //failed to login
                MessageBox.Show("Failed to login. Check Your User name and password!");
            }
        }
    }
    else
    {
        MessageBox.Show("Username should contain at least 3 characters and password at least 4");
    }
}
```

הסבר על שימוש בRegex:

הפעולה Regex.IsMatch בודקת אם מחזורת עונה על דפוס של תווים מוגדר כלשהו. בפעולה EnterB יש שימוש בפעולה זו על מנת לבדוק אם במחזורת יש תווים שאינם אות או מספר, ובמידה וכן, מזרקת הודעת שגיאה. על מנת להשתמש בספריה, הצהרתי עליה בראשית התוכנית.

```
using System.Text.RegularExpressions;
```

הפעולה EnterB מתבצעת בלחיצת כפתור, והיא מנסה להתחבר למשתמש בעזרת שם המשתמש והסיסמא שהזין המשתמש. החלון ישלח שגיאה עבור שגיאה בהזנת נתונים בצורה נכונה.

חלון הרשמה: Sign Up

# Sign Up

User Name

Password

Password again

First Name

Last Name

Email

הרשמה בהתאם לתנאים בשדות  
על מנת להיכנס לתוכנה, יש תחילה להירשם אליה.  
בהרשמה על השם משתמש, השם הפרטי, ושם המשפחה להכיל רק אותיות ו/או מספרים.  
סיסמא יכולה להכיל כל תו אך אורכה לפחות ארבעה תווים  
כתובת דוא"ל חייבת להכיל את התו '@' כשלפניו לפחות שני תווים של אותיות/מספרים ואחריו  
לפחות 2 תווים, נקודה ואחריה עוד תו אחד לפחות.  
בדיקת תקינות:

```
/// <summary>
/// method checks if entered data is valid.
/// </summary>
/// <param name="user">users data</param>
/// <returns>true if valid, false if not</returns>
1 reference
private bool IsValidUser(User user)
{
```

```
private bool IsValidUser(User user)
{
    bool valid = true;
    if (!Regex.IsMatch(user.Username, "^[a-zA-Z0-9]*$") && user.Username.Length > 3)
    {
        valid = false;
        MessageBox.Show("User name Should contain only letters and/or numbers\n" +
            "and the length is 3 characters minimum");
    }
    if (!Regex.IsMatch(user.Lname, "^[a-zA-Z0-9]*$") && user.Lname.Length > 3)
    {
        valid = false;
        MessageBox.Show("last name Should contain only letters and/or numbers\n" +
            "and the length is 3 characters minimum");
    }
    if (!Regex.IsMatch(user.Fname, "^[a-zA-Z0-9]*$") && user.Fname.Length > 3)
    {
        MessageBox.Show("first name Should contain only letters and/or numbers\n" +
            "and the length is 3 characters minimum");
        valid = false;
    }
}
```

```
try//check if email is in valid format by trying create MailAddress object.
{
    MailAddress mailAddress = new MailAddress(user.Email);
}
catch (FormatException)
{
    valid = false;
    MessageBox.Show("Not valid Email, check again your mail adress");
}
return valid;
```

```

/// <summary>
/// adds user to the server if valid data was entered. than logins and moves
/// to the main window
/// </summary>
/// <param name="sender">button</param>
/// <param name="e"></param>
1 reference
private void EnterBtn(object sender, RoutedEventArgs e)
{
    User user = new User();
    if (UPassword1.Password == UPassword2.Password)
    {
        //add info to a User object
        user.Uname = UName.Text;
        user.Lname = Lname.Text;
        user.Fname = FName.Text;
        user.Password = UPassword1.Password;
        user.EntrDate = DateTime.Now;
        user.Description = user.Uname;
        user.Email = EMail.Text;
    }
}

```

```

    if (IsValidUser(user))
    {
        srv.InsertUser(user);
        user = srv.Login(user.Uname, user.Password);
        if (user != null)
        {
            MainWindow main = new MainWindow(user);
            main.Show();
            this.Close();
        }
        else
        {
            MessageBox.Show("Failed to enter, try another username");
        }
    }
    else
    {
        MessageBox.Show("The First password and the Second password don't match!");
    }
}

```

הפעולה מתבצעת בלחיצת כפתור, היא בודקת האם המידע שהוזן נכנס באופן תקין, במידה ולא, נשלחת הודעת שגיאה רלוונטית. אם הכל תקין, המשתמש מתווסף לשרת, ולאחר מכן, מתבצע ניסיון התחברות על מנת להיכנס. אם ניסיון ההתחברות נכשל, ככל הנראה משתמש עם שם משתמש זה כבר קיים, לכן השרת אינו מאפשר להתחבר.

ישנו שימוש ב-ResourceDictionary בחלונות שונים על מנת להגדיר תכונות לאובייקטים המופיעים על המסך ב-XAML מבלי לחזור על כתיבת אותן תכונות שוב ושוב  
מהו ResourceDictionary ?



פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
ResourceDictionary הוא מאגר תכונות עבור משאבים ב XAML, כמו styles, שהאפליקציה משתמשת בהם. ניתן להגדיר את משאבים בעלי תכונות חוזרות ב-XAML ולאחר מכן תוכל להשתמש באותה תבנית באובייקטים באמצעות {StaticResource} לדוגמא:

```
<Window.Resources>
  <ResourceDictionary>
    <Style x:Key="Menu_Button" TargetType="Button">
      <Style.Resources>
        <Style TargetType="Border">
          <Setter Property="CornerRadius" Value="5"/>
        </Style>
      </Style.Resources>
      <Setter Property="Foreground" Value="LightYellow" />
      <Setter Property="FontSize" Value="30" />
      <Setter Property="Height" Value="50" />
      <Setter Property="Width" Value="120" />
      <Setter Property="Margin" Value="10" />
      <Setter Property="BorderThickness" Value="2" />
      <Setter Property="Background" Value="#00B1F9" />
    </Style>
  </ResourceDictionary>
</Window.Resources>
```

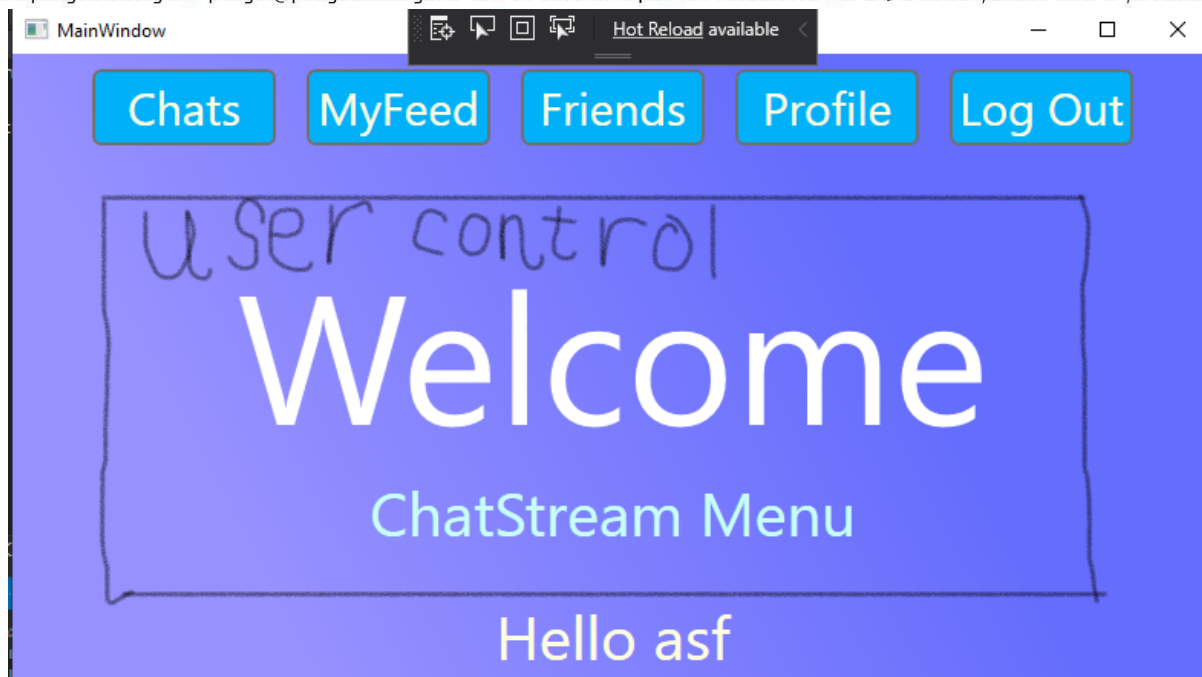
```
<WrapPanel Grid.Row="0" Grid.Column="1" VerticalAlignment="Top" HorizontalAlignment="Center">
  <Button Content="Chats" Style="{StaticResource Menu_Button}" Click="Chat_Click" />
  <Button Content="MyFeed" Style="{StaticResource Menu_Button}" Click="FeedB" />
  <Button Content="Friends" Style="{StaticResource Menu_Button}" Click="FriendB_click"/>
  <Button Content="Profile" Style="{StaticResource Menu_Button}" Click="ProfileB" />
  <Button Content="Log Out" Style="{StaticResource Menu_Button}" Click="OutB" />
</WrapPanel>
```

:Content controle

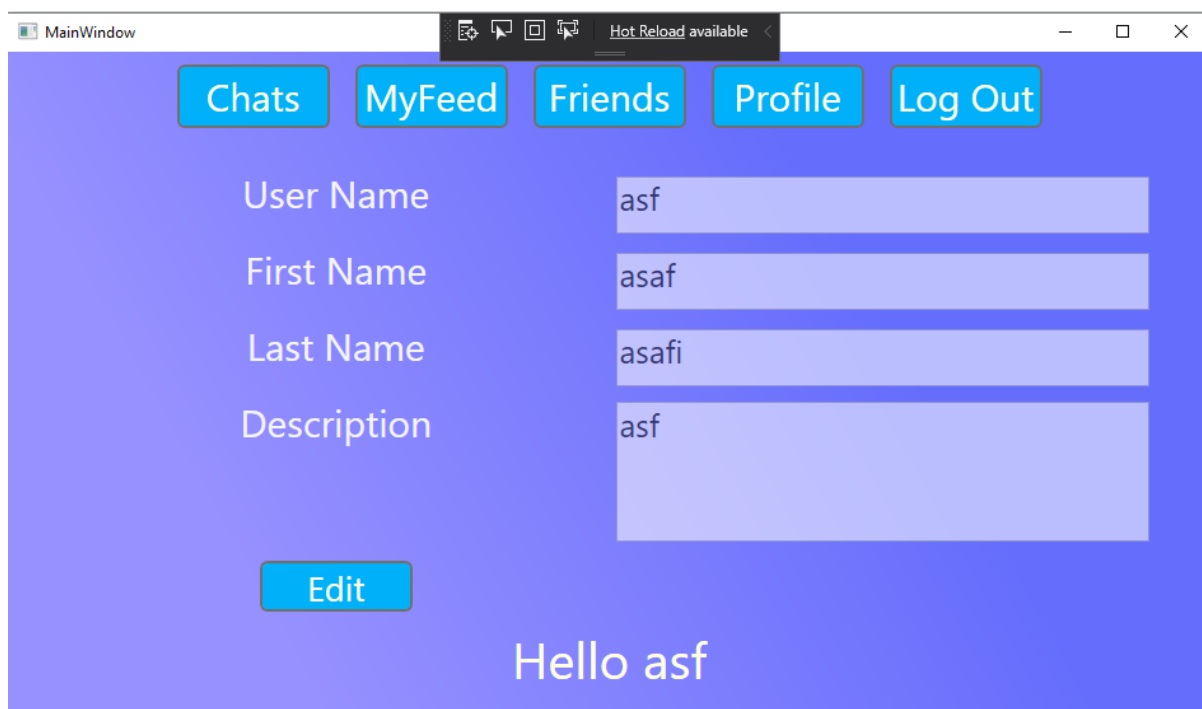
במסך הראשי mainwindow ישנו Content controle שבעזרתו ניתן להחליף את התוכן המוצג בתוך החלון הראשי. בלחיצת כפתור לכניסה לפרופיל למשל המסך המרכזי יתחלף ל UserControl של פרופיל.

```
ContentControl x:Name="ChangeContent" Grid.Row="1" Grid.Column="1"/>
```

```
1 reference
private void ProfileB(object sender, RoutedEventArgs e)
{
    this.ChangeContent.Content = new Profile(this.user);
}
```



||  
v



חלק מהאמא של החלון הראשי

```
<Grid>
<Rectangle Style="{StaticResource MainBackground}" />
<Grid>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="*" />
<ColumnDefinition Width="20*" />
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
<RowDefinition Height="5*" />
<RowDefinition Height="20*" />
<RowDefinition Height="4*" />
</Grid.RowDefinitions>
<WrapPanel Grid.Row="0" Grid.Column="1" VerticalAlignment="Top" HorizontalAlignment="Center">
<Button Content="Chats" Style="{StaticResource Menu_Button}" Click="Chat_Click" />
<Button Content="MyFeed" Style="{StaticResource Menu_Button}" Click="FeedB" />
<Button Content="Friends" Style="{StaticResource Menu_Button}" Click="FriendB_click"/>
<Button Content="Profile" Style="{StaticResource Menu_Button}" Click="ProfileB" />
<Button Content="Log Out" Style="{StaticResource Menu_Button}" Click="OutB" />
</WrapPanel>
<TextBlock Text="" Grid.Column="1" Grid.Row="2" FontSize="40" Foreground="LightYellow" VerticalAlignment="Top" />
<ContentControl x:Name="ChangeContent" Grid.Row="1" Grid.Column="1"/>
</Grid>
</Grid>
```

קוד #C של אותו חלון:

```
public partial class MainWindow : Window
{
    Service1Client srv = new Service1Client();

    private User user;

    public User User { get => user; set => user = value; }

    /// <summary>
    /// initialize main window
    /// </summary>

    /// <param name="user">user that has been logged</param>

    public MainWindow(User user)
    {
        InitializeComponent();

        this.ChangeContent.Content = new OpenMenu();

        this.user = user;

        HelloT.Text += "Hello " + user.Username;
    }
}
```

- מתאחל את החלון עם שם המשתמש מוצג בתחתית העמוד

//Exit MainWindow, Log out, moves to login window

```
private void OutB(object sender, RoutedEventArgs e)
{
    (new Login()).Show();
    this.Close();
}
```

- התנתקות ומעבר לחלון התחברות

// opens profile's data with edit options

```
private void ProfileB(object sender, RoutedEventArgs e)
{
    this.ChangeContent.Content = new Profile(this.user);
}
```

- מעבר ל UserControl של מידע על הפרופיל עם אפשרות עריכה.

//opens this main user's personal feed

```
private void FeedB(object sender, RoutedEventArgs e)
{
    this.ChangeContent.Content = new Feed(this);
}
```

- פתיחת תצוגת הפוסטים האישית הראשית של המשתמש, שאותם המשתמש הגדיר.

//opens friend list with option to add friends

```
private void FriendB_click(object sender, RoutedEventArgs e)
{
    this.ChangeContent.Content = new FriendListShow(this);
}
```

- פתיחת תצוגת חברים, עם אפשרות להוסיף חברים ולאשר בקשות מעקב.

//opens chats menu

```
{
    this.ChangeContent.Content = new ChatsShow(this);
}
```

- פתיחת תפריט התכתבויות

```
//gets current UserControl Displayed
public UserControl GetCurrentControle()
{
    UserControl control = null;

    Application.Current.Dispatcher.Invoke(() =>
    {
        if (ChangeContent.Content is UserControl userControl)
        {
            control = userControl;

            // Do something with 'control'
        }
    });

    return control;
}
```

{

- קבלת UserControl המוצג על פני המסך הראשי ברגע זה. הסבר על  
Application.Current.Dispatcher.Invoke בהמשך בעמוד \*\*73 ביחד עם ההסבר על  
Threads

App.xaml

ניתן גם פה להגדיר ResourceDictionary שניתן לגשת אליו מכל מקום באפליקציה.

```
<Application.Resources>
  <ResourceDictionary>
    <Style x:Key="SR_Buttons" TargetType="Button">
      <Style.Resources>
        <Style TargetType="Border">
          <Setter Property="CornerRadius" Value="5"/>
        </Style>
      </Style.Resources>
      <Setter Property="BorderBrush" Value="Transparent" />
      <Setter Property="Foreground" Value="Black" />
      <Setter Property="FontSize" Value="25" />
      <Setter Property="Height" Value="46" />
      <Setter Property="Margin" Value="10, 2" />
      <Setter Property="BorderThickness" Value="2" />
      <Setter Property="Background" Value="Transparent" />
    </Style>
  </ResourceDictionary>
</Application.Resources>
```

ReasourceDictionary מגדיר מאגר לתכונות לאובייקטים המוצגים בUI, כאשר style הם התכונות לאובייקט מסויים מסוג המוגדר בTargetType.

לדוגמא בStyle מסוג SR\_Buttons עבור אובייקטים של תצוגה מסוג Button מוגדר:

גבול מעוגל ברדיוס פינות 5. עבור התכונה Border

צבע גבול "Property="BorderBrush", שקוף "Value="Transparent"

גודל פונט של טקסט - "Property="FontSize" Value="25 25"

וכן

```
<Style x:Key="AVR_Button" TargetType="Button">
  <Style.Resources>
    <Style TargetType="Border">
      <Setter Property="CornerRadius" Value="5"/>
    </Style>
  </Style.Resources>
  <Setter Property="Foreground" Value="LightYellow" />
  <Setter Property="FontSize" Value="30" />
  <Setter Property="Height" Value="60" />
  <Setter Property="Width" Value="120" />
  <Setter Property="Margin" Value="10" />
  <Setter Property="BorderThickness" Value="2" />
  <Setter Property="Background" Value="#00B1F9" />
</Style>
```

```
<Style x:Key="MainBackground" TargetType="Rectangle">
  <Setter Property="Fill">
    <Setter.Value>
      <LinearGradientBrush StartPoint="0,1" EndPoint="1,0">
        <GradientStop Color="#9791ff" Offset="0.2" />
        <GradientStop Color="#656dfc" Offset="0.7" />
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
</Style>
<Style x:Key="RButtons" TargetType="Button">
  <Style.Resources>
    <Style TargetType="Border">
      <Setter Property="CornerRadius" Value="5" />
    </Style>
  </Style.Resources>
  <Setter Property="BorderBrush" Value="Transparent" />
  <Setter Property="Foreground" Value="Black" />
  <Setter Property="FontSize" Value="30" />
  <Setter Property="Height" Value="46" />
  <Setter Property="Margin" Value="10, 2" />
  <Setter Property="BorderThickness" Value="2" />
</Style>
</ResourceDictionary>
```

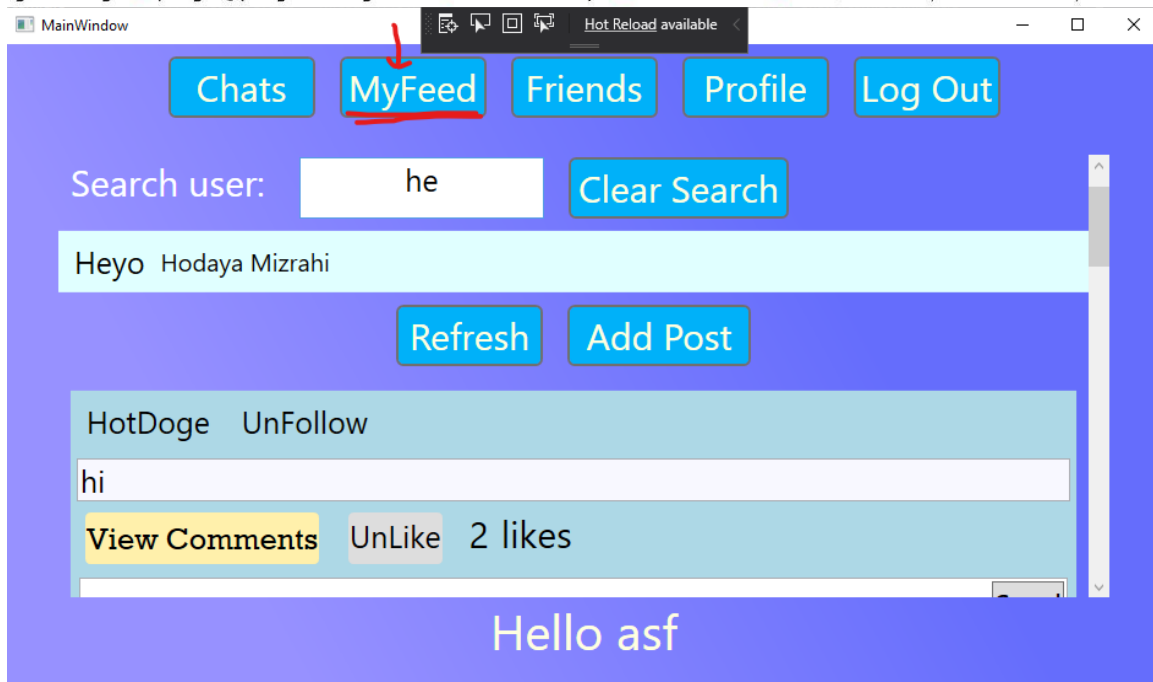
בהדרגה במילוי זה. LinearGradientBrush הוא מילוי צבעים אלכסוני של הרקע שאותו הגדרתי. הצבעים משתנים

הFeed שלי:

כל הפוסטים שהמשתמש מעוניין לראות. הפוסטים של כל המשתמשים שהמשתמש עוקב אחריהם. ניתן לחפש משתמשים ולהיכנס לUserFeed של הפוסטים שלהם ובכך להחליט אם לעקוב אחריהם או להיות חברים שלהם.

בפיד ניתן לחפש משתמשים, לרענן את התוצאות, להוסיף פוסט, וכמובן לראות פוסטים כאשר כל האופציות בUserContol של אותם פוסטים זמינות.





```

<ScrollView Grid.Row="1" VerticalAlignment="Top">
  <StackPanel>
    <TextBlock Text="My Feed" Style="{StaticResource textS}"/>
    <WrapPanel>
      <TextBlock Text="Search user: " Margin="10" FontSize="30" TextAlignment="Center" Foreground="White"/>
      <TextBox Width="200" FontSize="25" TextAlignment="Center" x:Name="USerach_TB" Margin="10" TextChanged="
      <Button Style="{StaticResource AVR_Button}" Height="50" Width="180" Content="Clear Search" Click="Butto
    </WrapPanel>
    <StackPanel Background="LightCyan" x:Name="Search_Results">
    </StackPanel>
    <WrapPanel HorizontalAlignment="Center">
      <Button Style="{StaticResource AVR_Button}" Height="50" Content="Refresh" Click="Refresh_Click" />
      <Button x:Name="AddPost" Style="{StaticResource AVR_Button}" Width="150" Height="50" Content="Add Post"
    </WrapPanel>
    <ContentControl Grid.Row="4" x:Name="CreatePostV" Visibility="Visible"/>
    <StackPanel x:Name="FeedView">
    </StackPanel>
  </StackPanel>
</ScrollView>

```

ScrollView שבתוכו StackPanel מאפשר לגלול בתוכו שנמצא בתוך ה-StackPanel. FeedView הוא StackPanel שמטרתו להציג פוסטים שנודעו לתצוגה. כל פוסט מתווסף לאותו StackPanel, ולאחר מכן במידה ונעשה רענון, כל הפוסטים בתוך ה-StackPanel ימחקו ויעלו מחדש.

```
public partial class Feed : UserControl
{
    private MainWindow main;
    Service1Client srv = new Service1Client();
    1 reference
    public Feed(MainWindow main)
    {
        this.main = main;
        InitializeComponent();
        //create a UserControl of CreatePost, where user can createpost
        CreatePostV.Content = new CreatePost(main.User, AddPost);
        ((CreatePost)CreatePostV.Content).Visibility = Visibility.Collapsed;
        //receive all the posts that user should see in the feed
        List<Post> posts = srv.SelectMyFeed(this.main.User).ToList();

        foreach (Post post in posts) //show each post
        {
            UserControl userControl = new UserControl();
            userControl.Content = new PostView(post, main, this);
            userControl.Margin = new Thickness(10);
            FeedView.Children.Add(userControl);
        }
    }
}
```

כפתור שפותח את תצוגת יצירת הפוסט. נשלח רפרנס אליו ביצירת UserControl של AddPost על מנת להחזיר את הכפתור כאשר מסיימים להכין פוסט. הפעולה מקבלת מהשרת את כל הפוסטים שמתאימים לתצוגת הפיד. עבור כל פוסט בלולאה, הפעולה מוסיפה אותו לתצוגה של כל הפוסטים.

```
/// <summary>
/// the method runs when an event of changing the text box content occurs
/// the method shows all the users that start with what was typed in text box
/// </summary>
/// <param name="sender">textbox</param>
/// <param name="e"></param>
1 reference
private void USerach_TB_TextChanged(object sender, TextChangedEventArgs e)
```

```
{
    TextBox textBox = sender as TextBox;
    Search_Results.Children.Clear();
    if (textBox.Text.Length > 0) //if user searched uname
    {
        //select all the usernames starts with what was typed in the search bar
        List<User> userlist = srv.SelectUserByUName(textBox.Text).ToList();
        foreach (User user in userlist)
        {
            WrapPanel wrapPanel = new WrapPanel();
            EntityButton uButton = new EntityButton();
            TextBlock name = new TextBlock();
            //button with username that shows user's personal feed and info when clicked
            uButton.Style = FindResource("SR_Buttons") as Style;
            uButton.Content = user.Uname;
            uButton.Entity = user;
            uButton.Click += UFeed;
            //Text Block that presents the full name of the user
            name.Style = FindResource("textS") as Style;
            name.Foreground = Brushes.Black;
            name.Text = user.Fname + " " + user.Lname;
            name.FontSize = 20;
            wrapPanel.Children.Add(uButton as Button);
            wrapPanel.Children.Add(name);
            Search_Results.Children.Add(wrapPanel); // add the search results to the stackpanel
        }
    }
}
```

הפעולה הזו מחפשת משתמש קיים על מנת להיכנס לפיד שלו. הפעולה מקבלת רשימת משתמשים שמתחילים בתווים שהמשתמש רשם ב־TextBox, ולאחר מכן מציגה אותם. SelectByUName מחפשת את כל המשתמשים שהשם משתמש מתחיל ברצף התווים המוקלד. לאחר ייצור WrapPanel שבתוכו כפתור שקוף ועגול עם שם המשתמש שלחיצה בו - תוביל לדף הפיד של המשתמש עם המידע עליו, והשם המלא של המשתמש. FindResource מחפשת resource שהוגדר ב-XAML על מנת להוסיף תבנית עיצוב שכבר קיימת.

```
/// <summary>
/// opens user's feed by clicking the button named by username.
/// </summary>
/// <param name="sender">button</param>
/// <param name="e"></param>
1 reference
private void UFeed(object sender, RoutedEventArgs e)
{
    EntityButton userButton = sender as EntityButton;
    this.main.ChangeContent.Content = new UserFeed((User)userButton.Entity, this.main, this);
}
```

כפתור UFeed פותח את ה־feed עם המידע והפוסטים של משתמש כלשהו. UserFeed מקבל את הפרמטר של המשתמש כדי להציג את המידע שלו ולקבל את כל הפוסטים שלו, הוא מקבל את החלון הראשי כדי שיוכל החליף UserControl, והוא שולח מצביע ל־usercontrol הנוכחי על מנת לאפשר לחיצה של חזרה אחורה, שיש בחלון UserFeed.

```

/// <summary>
/// this button shows the add post UserControl in order to allow the user
/// to add a post.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
1 reference
private void AddPost_Click(object sender, RoutedEventArgs e)
{
    Button button = sender as Button;
    button.Visibility = Visibility.Collapsed;
    ((CreatePost)CreatePostV.Content).Visibility = Visibility.Visible;
}
/// <summary>
/// the clear Search button clears the content searched in the textbox
/// </summary>
/// <param name="sender">button</param>
/// <param name="e"></param>
0 references
private void ClearSearch_Click(object sender, RoutedEventArgs e)
{
    USerach_TB.Text = "";
}

```

הפעולה AddPost מציגה את UserControl שמציג את כל השדות למילוי המידע על הפוסט החדש.

```

/// <summary>
/// refreshes all the posts in the feed
/// </summary>
/// <param name="sender">refresh button </param>
/// <param name="e"></param>
2 references
public void Refresh_Click(object sender, RoutedEventArgs e)
{
    FeedView.Children.Clear();
    List<Post> posts = srv.SelectMyFeed(this.main.User).ToList();
    foreach (Post post in posts)
    {
        UserControl userControl = new UserControl();
        userControl.Content = new PostView(post, main, this);
        userControl.Margin = new Thickness(10);
        FeedView.Children.Add(userControl);
    }
}

```

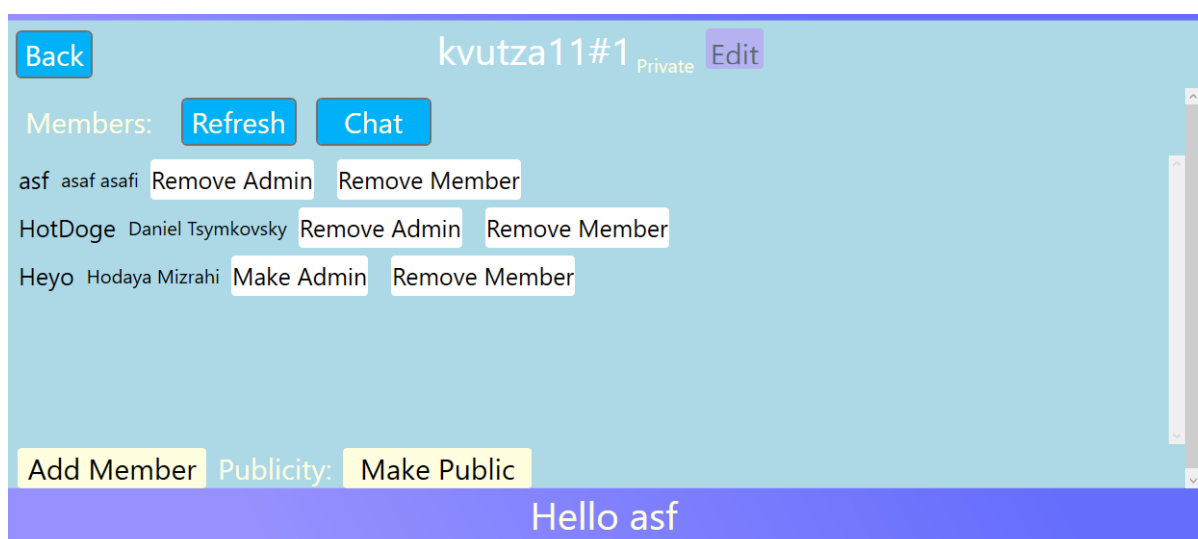
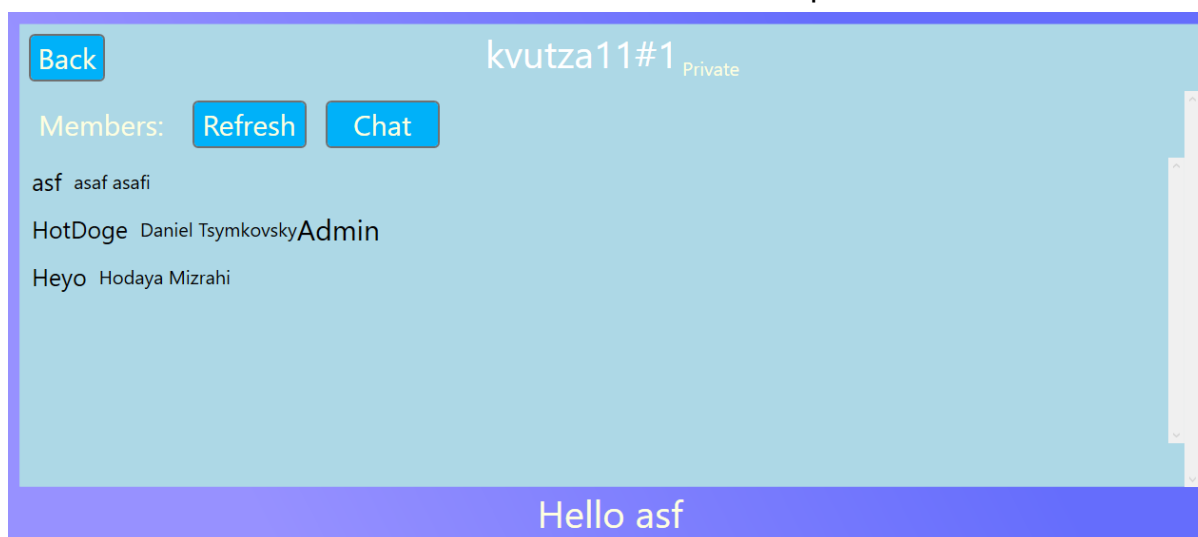
הפעולה Refresh\_Click טוענת את כל הפוסטים עבור הפיד של המשתמש ומציגה אותם. היא טוענת את כל הפוסטים שאמורים להיות מוצגים בפיד מהמסד נתונים, ומציגה אותם בפיד. הפעולה מוסיפה כל פוסט שהתקבל ל-FeedView.

```
18 references
public class EntityButton : Button
{
    private BaseEntity entity;
    4 references
    public EntityButton() : base() { this.entity = null; }
    0 references
    public EntityButton(BaseEntity entity) : base() { this.entity = entity; }

    8 references
    public BaseEntity Entity { get => entity; set => entity = value; }
}
```

EntityButton הינו כפתור מיוחד שהגדרתי שיכול להכיל אובייקט מסוג Entity על מנת שהכפתור יוכל לאחסן מידע על מנת להשתמש בו בזמן לחיצה.

חלון מידע על קבוצה - ישנם שני סוגי משתמשים בחלון זה. מנהל ומשתמש רגיל. מנהל יכול לשנות את שם הקבוצה, להוסיף או למחוק משתמשים, ולהפוך אותה לפרטית או ציבורית. משתמש יכול לראות את פרטי הקבוצה.



```
public partial class GroupInfo : UserControl

{

    //properties

    Service1Client srv = new Service1Client();

    private MainWindow main;

    private Group group;

    private UserControl previous;

    private bool isAdmin;

    public GroupInfo(MainWindow main, GMember member, UserControl previous)
    {

        this.main = main;

        this.group = srv.GetGroupByID(member.Group.ID); //get group data

        this.previous = previous;

        member.User = main.User;

        InitializeComponent();

        GNameTxt.Text = this.group.GNname + "#" + group.ID; //show group name with ID

        this.isAdmin = srv.IsGroupAdmin(member);

        RefreshGMembers();//show all group members

        //display if group is private or public

        PublicityDisplayText.Text = this.group.Publicity ? "Public" : "Private";

        if(this.isAdmin)

        {

            AdminsOptionPanel.Visibility = Visibility.Visible;

            EditGroupName.Visibility = Visibility.Visible;

            //display the right button to change the group's publicity:

            (this.group.Publicity ? MakePublic_btn : MakePrivate_btn).Visibility = Visibility.Collapsed;
```

- הפעולה יוצרת את העצם של GroupInfo. היא מציגה את כל המידע על הקבוצה, טוענת את חבריה, ונותנת למשתמשים מנהלים אפשרויות עריכה, הוספת ומחיקת משתמשים.

}

/// <summary>

/// refresh the members list of the group. add editing options

/// for admins

/// </summary>

private void RefreshGMembers()

{

GMember gMember = new GMember();

gMember.User = this.main.User;

gMember.Group = this.group;

this.isAdmin = srv.IsGroupAdmin(gMember);

List<GMember> gMembers = srv.GetGroupMembers(this.group).ToList();

- קבלת כל המשתמשים של הקבוצה.

MemberListStack.Children.Clear();

foreach (GMember member in gMembers)

{

TextBlock name = new TextBlock();//user's name

uButton.Style = FindResource("SR\_Buttons") as Style;

uButton.Content = member.User.Username;

uButton.Entity = member.User;

uButton.Click += UFeed;

name.Style = FindResource("textS") as Style;

name.Foreground = Brushes.Black;

name.Text = member.User.Fname + " " + member.User.Lname;

name.FontSize = 20;



```
wrapPanel.Children.Add(name);
```

- אם משתמש הראשי מנהל, יש לתת לו אפשרויות מנהל כמו מחיקת משתמש.

```
if (this.isAdmin)
{
    EntityButton makeAdmin = new EntityButton();

    EntityButton removeAdmin = new EntityButton();

    EntityButton removeMember = new EntityButton();

    makeAdmin.Style = removeAdmin.Style = removeMember.Style =
FindResource("RW_Buttons") as Style;

    makeAdmin.Content = "Make Admin";

    removeAdmin.Content = "Remove Admin";

    removeMember.Content = "Remove Member";

    makeAdmin.Entity = removeAdmin.Entity = removeMember.Entity= member;

    makeAdmin.Click += MakeAdmin_Click;

    removeAdmin.Click += RemoveAdmin_Click;

    removeMember.Click += RemoveMember_Click;

    (member.Admin ? makeAdmin : removeAdmin).Visibility = Visibility.Collapsed;

    wrapPanel.Children.Add(makeAdmin as Button);

    wrapPanel.Children.Add(removeAdmin as Button);

    wrapPanel.Children.Add(removeMember as Button);

}

else
{

    if(member.Admin)
    {

        TextBlock adminText = new TextBlock();

        adminText.Style = FindResource("textS") as Style;

        adminText.Foreground = Brushes.Black;
```

```
adminText.Text = "Admin";
```

```
adminText.FontSize = 30;
```

```
wrapPanel.Children.Add(adminText);
```

```
}
```

```
}
```

```
MemberListStack.Children.Add(wrapPanel);
```

```
}
```

```
}
```

//make a user admin in a group button click

```
private void MakeAdmin_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
EntityButton eButton = sender as EntityButton;
```

```
GMember member = eButton.Entity as GMember;
```

```
member.Admin = true;
```

```
srv.UpdateGMemberStatus(member);
```

```
}
```

- הפעולה מעדכנת את סטטוס המשתמש למנהל.

//remove admin from a user button click

```
private void RemoveAdmin_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
EntityButton eButton = sender as EntityButton;
```

```
GMember member = eButton.Entity as GMember;
```

```
member.Admin = false;
```

```
srv.UpdateGMemberStatus(member);
```

```
}
```

- הפעולה מבטלת סטטוס מנהל, ביחד עם אפשרויות מנהל למנהל מסויים.

// delete group member button click

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
private void RemoveMember\_Click(object sender, RoutedEventArgs e)

```
{
    EntityButton eButton = sender as EntityButton;

    GMember member = eButton.Entity as GMember;

    srv.DeleteGMember(member);
}
```

- פעולה זו שולחת הוראה אל השרת למחוק את המשתמש מהקבוצה

//add member to the group button click

private void AddMemberBtn\_Click(object sender, RoutedEventArgs e)

```
{
    MemberAddStack.Visibility = Visibility.Visible;

    ((Button)sender).Visibility = Visibility.Collapsed;
}
```

- הצגת אפשרויות הוספה משתמש, חיפוש והוספה.

//clear user search for adding button click

private void Clear\_Click(object sender, RoutedEventArgs e)

```
{
    USerach_TB.Text = "";
}
```

//make group public button click

private void MakePublic\_btn\_Click(object sender, RoutedEventArgs e)

```
{
    this.group.Publicity = true;

    srv.UpdateGroup(this.group);

    Thread.Sleep(2000); //wait for while, to make sure this would became public before the update is
done
```

this.group = srv.GetGroupByID(this.group.ID);

MakePublic\_btn.Visibility = group.Publicity ? Visibility.Collapsed : Visibility.Visible;

MakePrivate\_btn.Visibility = group.Publicity ? Visibility.Visible : Visibility.Collapsed;

}

- לחיצה על הכפתור זה תגרום לקבוצה להיות ציבורית בכך שתשלח בקשה להפוך את הקבוצה לציבורית. ולאחר מכן תראה שינוי בסטטוס קבוצה רק אם על פי השרת, הקבוצה הפכה לציבורית.

//make the group private button click

private void MakePrivate\_btn\_Click(object sender, RoutedEventArgs e)

{

this.group.Publicity = false;

srv.UpdateGroup(this.group);

Thread.Sleep(2000);

this.group = srv.GetGroupByID(this.group.ID);

MakePublic\_btn.Visibility = group.Publicity ? Visibility.Collapsed : Visibility.Visible;

MakePrivate\_btn.Visibility = group.Publicity ? Visibility.Visible : Visibility.Collapsed;

PublicityDisplayText.Text = this.group.Publicity ? "Public" : "Private";

}

- לחיצה על הכפתור זה תגרום לקבוצה להיות פרטית, בכך שתשלח בקשה להפוך את הקבוצה פרטית. ולאחר מכן תראה שינוי בסטטוס קבוצה רק אם על פי השרת, הקבוצה הפכה לפרטית.

private void USerach\_TB\_TextChanged(object sender, TextChangedEventArgs e)

{

TextBox textBox = sender as TextBox;

MemberAddStack\_Results.Children.Clear();

if (textBox.Text.Length > 0)

{

List<User> userlist = srv.SelectUserByUName(textBox.Text).ToList();

foreach (User user in userlist)

{

WrapPanel wrapPanel = new WrapPanel();

EntityButton = new EntityButton ();

```
EntityButton addAsMember = new EntityButton ();
```

```
TextBlock name = new TextBlock();
```

```
uButton.Style = FindResource("SR_Buttons") as Style;
```

```
uButton.Content = user.Uname;
```

```
uButton.User = user;
```

```
uButton.Click += UFeed;
```

```
name.Style = FindResource("textS") as Style;
```

```
name.Foreground = Brushes.Black;
```

```
name.Text = user.Fname + " " + user.Lname;
```

```
name.FontSize = 20;
```

```
addAsMember.Style = FindResource("SR_Buttons") as Style;
```

```
addAsMember.Content = "Add";
```

```
addAsMember.User = user;
```

```
addAsMember.Click += AddAsMember_Click;
```

```
wrapPanel.Children.Add(uButton as Button);
```

```
wrapPanel.Children.Add(name);
```

```
wrapPanel.Children.Add(addAsMember as Button);
```

```
MemberAddStack_Results.Children.Add(wrapPanel);
```

```
}
```

```
}
```

```
}
```

- חיפוש משתמש בדומה לחיפוש ב-UserControls אחרים רק שהכפתורים שמוצאים פה הם מחיקת המשתמש, הפיכתו למנהל או הסרת מנהל.

```
private void UFeed(object sender, RoutedEventArgs e)
```

```
{
```

```
EntityButton userButton = sender as EntityButton ;
```

```
this.main.ChangeContent.Content = new UserFeed((User)userButton.Entity, this.main, this);
```

```
}
```

```
private void AddAsMember_Click(object sender, RoutedEventArgs e)

{

    EntityButton userButton = sender as EntityButton;

    srv.InsertGMemeber(this.group, (User)userButton.Entity);

    Thread.Sleep(2000); //wait 2 seconds before refreshing

    RefreshGMembers();

}
```

- כפתור זה מוסיף משתמש לקבוצה. במידה והוא התווסף, ניתן יהיה לראות את השינוי לאחר שתי שניות.

//stop showing the adding options. stop showing the user search

```
private void CloseAdd_Click(object sender, RoutedEventArgs e)

{

    MemberAddStack.Visibility = Visibility.Collapsed;

    AddMemberBtn.Visibility = Visibility.Visible;

    Clear_Click(sender, e);

}
```

//open editing the group name

```
private void EditGroupName_Click(object sender, RoutedEventArgs e)

{

    GNameTxt.Text = this.group.GNname;

    GNameTxt.IsReadOnly = false;

    GNameTxt.BorderBrush = Brushes.Black;

    SaveGroupName.Visibility = Visibility.Visible;

    EditGroupName.Visibility = Visibility.Collapsed;

}
```

- עריכת שם הקבוצה. בזמן העריכה הID של הקבוצה אינו מוצג כחלק משם הקבוצה כדי שלא יחשב כחלק מהשם.

//save group's name

```
private void SaveGroupName_Click(object sender, RoutedEventArgs e)
```

```
GNameTxt.IsReadOnly = true;
```

```
this.group.GNname = GNameTxt.Name;
```

```
GNameTxt.Text = this.group.GNname + "#" + this.group.ID;
```

```
GNameTxt.BorderBrush = Brushes.Transparent;
```

```
srv.UpdateGroup(group);
```

```
SaveGroupName.Visibility = Visibility.Collapsed;
```

```
EditGroupName.Visibility = Visibility.Visible;
```

```
}
```

- שמירת שם הקבוצה החדש, וביטול היכולת לערוך את TextBox של שם הקבוצה.

```
//refresh all members
```

```
private void RefreshMembers_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
RefreshGMembers();
```

```
}
```

- רענון חברי קבוצה.

```
//exit this UserControl and move to the previous one
```

```
private void Back_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
this.main.ChangeContent.Content = this.previous;
```

```
}
```

```
private void OpenChat_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
this.main.ChangeContent.Content = new ChatWithGroup(this.main, this.group, this);
```

```
}
```

```
}
```



```
public partial class ChatWithUser : UserControl
```

```
{
```

```
private MainWindow main;
```

```
private User user;
```

```
private UserControl sourceCont;
```

```
Service1Client srv = new Service1Client();
```

```
private bool isAutoUpdate;
```

```
private Thread autoUpdate;
```

```
public ChatWithUser(MainWindow main, User user, UserControl sourceCont)
```

```
{
```

```
this.main = main;
```

```
this.sourceCont = sourceCont;
```

```
this.user = user;
```

```
this.isAutoUpdate = true;
```

```
UserNameChat.Text = user.Uname + "#id:" + user.ID;
```

```
InitializeComponent();
```

```
this.autoUpdate = new Thread(this.AutoRefresh); //this thread will update messages every 2  
seconds
```

• יצירת Thread עבור עדכון אוטומטי של הצ'אט בזמן אמת.

```
List<DMessage> messages = srv.SelectAllDMsChat(this.main.User, this.user).ToList();
```

```
foreach (DMessage dMessage in messages)
```

```
{
```

```
AppendMessage(dMessage);
```

```
this.autoUpdate.Start();//start update messages every 2 seconds
}
```

Thread הינו תהליך בתוכנית שרץ במקביל לתהליכים אחרים ואינו תלוי בהם. התהליך רץ בנפרד מה שמאפשר לתוכנית לבצע כמה פעולות באותו הזמן. בצ'אט הקבוצתי יש צורך בתהליך ברקע שיבדוק כל הזמן אם יש הודעות חדשות על מנת למנוע תקיעות של התוכנה כל כמה זמן כדי להוריד הודעות. דבר זה מאפשר לבצע כמה פעולות באותו הזמן.

בצד שרת גם השרת הוא Multithreaded, כלומר מבצע מספר תהליכים באותו הזמן. השרת משרת כל פעם שלקוח מבקש ממנו שירות. כאשר כמה לקוחות מבקשים שירות, השרת מבצע את השירות עבור כל לקוח בנפרד. מה שמאפשר גמישות ויכולת לשרת כמה לקוחות באותו הזמן.

```
//appends certain message to the visible chat

private void AppendMessage(DMessage message)
{
    Application.Current.Dispatcher.Invoke(() => ///this made to allow using this function from a thread
    {
        var horizontalAI = HorizontalAlignment.Left;

        var messageColour = new SolidColorBrush(Color.FromRgb(0x00, 0x5d, 0x6e));

        if (message.DestUser.ID == this.user.ID)
        {
            horizontalAI = HorizontalAlignment.Right;

            messageColour = new SolidColorBrush(Color.FromRgb(0x4e, 0xad, 0x51));
        }

        WrapPanel wrapPanel = new WrapPanel()
        {
            Width = double.NaN,

            HorizontalAlignment = horizontalAI
        };
    });
}
```

```
Border border = new Border()// message's circular border

{

    Margin = new Thickness(5),

    Width = double.NaN,

    Padding = new Thickness(5),

    BorderThickness = new Thickness(1),

    BorderBrush = Brushes.Black,

    Background = messageColour,

    CornerRadius = new CornerRadius(10)

};

WrapPanel wrapMessage = new WrapPanel()

{

    Width = double.NaN,

    HorizontalAlignment = horizontalAl

};

TextBlock contentTextBlock = new TextBlock()//message content

{

    Foreground = Brushes.White,

    Width = double.NaN,

    TextWrapping = TextWrapping.Wrap,

    Text = message.Content,

    Padding = new Thickness(5, 0, 5, 0)

};

TextBlock dateBlock = new TextBlock() //message time sent

{

    Foreground = Brushes.White,

    Width = double.NaN,

    TextWrapping = TextWrapping.Wrap,
```

Padding = new Thickness(5, 0, 5, 0)

};

dateBlock.FontSize = 10;

contentTextBlock.FontSize = 20;

wrapMessage.Children.Add(contentTextBlock);

wrapMessage.Children.Add(dateBlock);

border.Child = wrapMessage;

wrapPanel.Children.Add(border);

ChatMessages.Children.Add(wrapPanel);

});

}

•

private void AutoRefresh() //refreash all messages every 2 seconds.

{

while(this.main.GetCurrentControle() == this)

{

Thread.Sleep(2000);

List<DMessage> messages = srv.SelectUnSeenDM(this.main.User, this.user).ToList();

foreach(DMessage dMessage in messages)

{

AppendMessage(dMessage);

}

}

}

//sends a message to the second user

private void SendMessage\_Click(object sender, RoutedEventArgs e)

{

```
DMessage message = new DMessage();
```

```
message.Content = textedMessage.Text;
```

```
message.SourceUser = this.main.User;
```

```
message.DestUser = this.user;
```

```
message.Time = DateTime.Now;
```

```
AppendMessage(message);
```

```
srv.InsertDM(message);
```

```
textedMessage.Text = "";
```

```
}
```

```
//returns to the previos UserControl shown by mainwindow
```

```
private void Back_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.main.ChangeContent.Content = this.sourseCont;
```

```
}
```

```
{
```

מדריך למשתמש:

האפליקציה ChatStream היא "רשת חברתית" שבה ניתן להתכתב עם אנשים בקבוצות או בזוגות, וניתן לפרסם פוסטים (מודעות) בפרופיל של המשתמש. משתמשים אחרים יכולים לתת לייק לפוסט ולהגיב.

כאשר פותחים את האפליקציה, נפתח חלון התחברות למשתמש - Login. בחלון זה המשתמש מזין שם משתמש וסיסמא על מנת לבצע התחברות וכניסה למערכת. במידה ולמשתמש לא רשום במערכת, הוא יכול לעבור לחלון ההרשמה על ידי לחיצה על הכפתור SignUp.

חלון התחברות- הכנסת שם משתמש וסיסמא תקינים. שם משתמש תקין הוא שם משתמש שאורכו לפחות 3 תווים, ומכיל רק אותיות ומספרים. סיסמא תקינה מכילה לפחות ארבעה תווים.

במידה והוכנס שם משתמש או סיסמא קצרים מדיי תקפוץ השגיאה הבאה:

במידה ושתי השדות מכילים מספיק תווים אבל שם המשתמש מכיל תו שאינו אות או מספר, תקפוץ השגיאה הבאה:

במידה והשדות תקינים אך שם המשתמש או הסיסמא שגויים, וההתבחרות נכשלה, תקפוצ ההודעה הבאה:

במידה והמשתמש אינו רשום ומחזיק במשתמש, ניתן להירשם למערכת בחלון SignUp. מעבר לחלון זה יתבצע בלחיצה על הכפתור SignUp.

חלון ההרשמה:

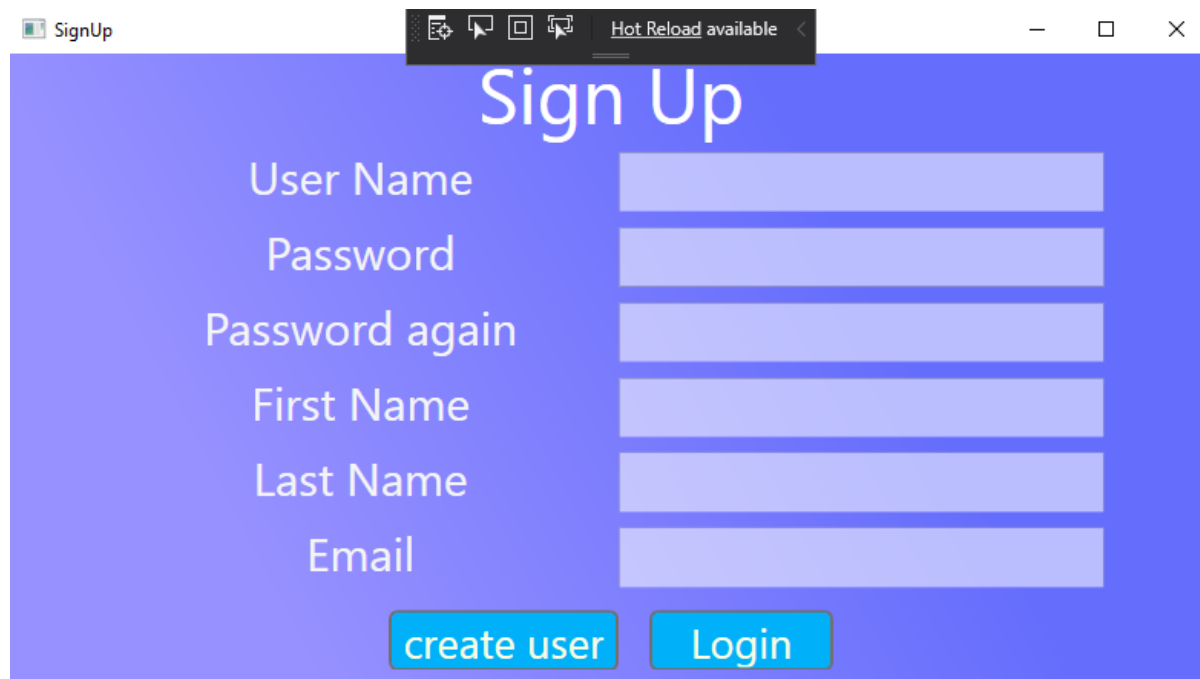
חלון זה נועד לאפשר למשתמש להירשם לאפליקציה - ליצור משתמש חדש. כדי להירשם עם משתמש חדש, על כל השדות להיות תקינים כדי לבצע את ההרשמה.

השדות User Name, First Name, Last name צריכים להכיל 3 תווים לפחות, כאשר התווים הם רק מספרים או אותיות.



השדה Email צריך להכיל כתובת email תקינה.

שני הסיסמאות צריכות להיות קודם כל זהות לפני שמתבצעות שער בדיקות התקינות.



Sign Up

User Name

Password

Password again

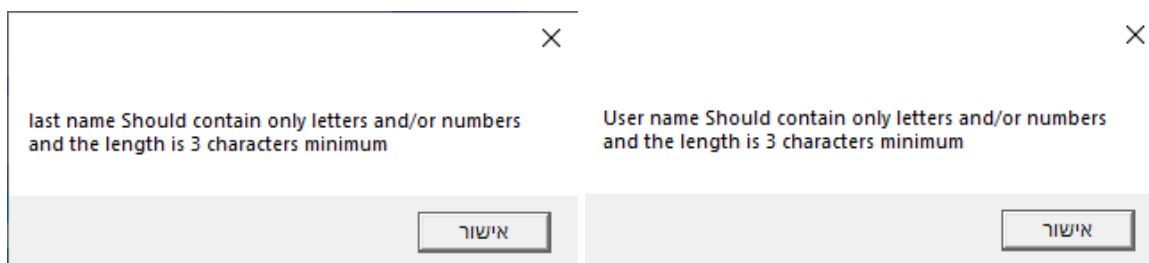
First Name

Last Name

Email

create user Login

שגיאות:



last name Should contain only letters and/or numbers and the length is 3 characters minimum

User name Should contain only letters and/or numbers and the length is 3 characters minimum

אישור

אישור

×

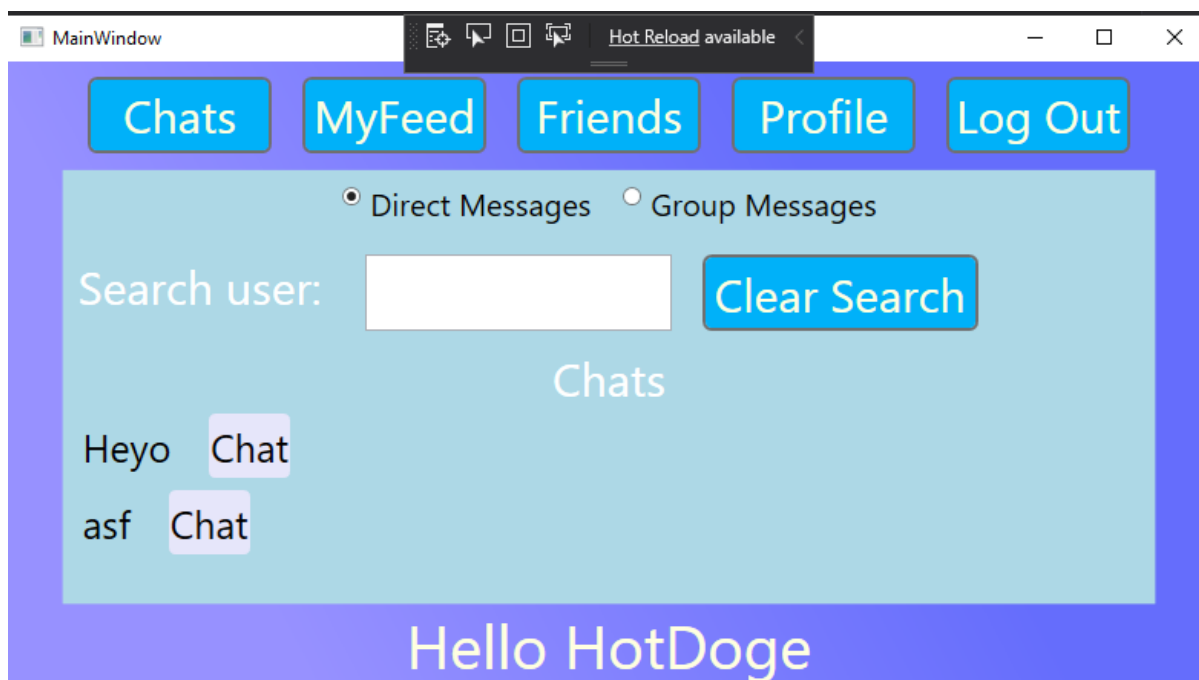
×



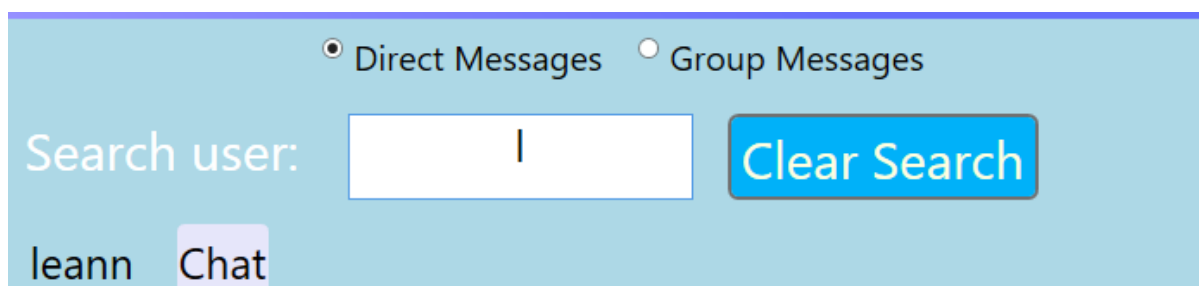
Log Out - יציאה מהמערכת

## Chats:

החלון שנפתח מכיל שני אופציות להתכתבויות- אחת התכתבות ישירה בין שני משתמשים, השנייה התכתבות בקבוצה שאליה מכניס המשתמש שיוצר אותה, או אם הקבוצה ציבורית, ניתן להיכנס אליה על ידי חיפוש בשדה Search:



ניתן להקליד שמות של משתמשים או קבוצות ציבוריות על מנת למצוא קבוצות שהמשתמש/ת רוצה להצטרף, או אנשים שהמשתמש/ת רוצה להתכתב איתם.



החלפה בין סוג ההתכתבות, קבוצתית או אישית תחליף את התוצאות שתפריט ההתכתבות מציג:

☐ Direct Messages
 ☒ Group Messages

Search Group:

Chats

kvutza11#1

על מנת להיכנס להתכתבות יש ללחוץ על Chat. על מנת לראות עוד פרטים על המשתמש או הקבוצה יש ללחוץ על שם המשתמש או הקבוצה.

תפריט מידע על הקבוצה:

MainWindow

Hot Reload available

Chats MyFeed Friends Profile Log Out

Back kvutza11#1 Private Edit

Members: Refresh Chat

asf asaf asafi Make Admin Remove Member

HotDoge Daniel Tsymkovsky Remove Admin Remove Member

Heyo Hodaya Mizrahi Make Admin Remove Member

Add Member Publicity: Make Public

Hello HotDoge

לחיצה על הוספת משתמש תפתח חיפוש משתמשים:

Publicity:

Search user:

yuval yuval gvbhbh Add

לחיצה על שם המשתמש יוביל לעמוד UserFeed עם המידע על המשתמש/ת.

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
לחיצה על Add תוסיף את המשתמש לקבוצה.

לחיצה על Close תסגור את אזור החיפוש.

לחיצה על MakePublic תהפוך את הקבוצה לציבורית, ותאפשר למשתמשים אחרים לחפש אותה.

ClearSearch תנקה את החיפוש.

בחלון עם שמות המשתמשים:

HotDoge	Daniel Tsymkovsky	Remove Admin	Remove Member
Heyo	Hodaya Mizrahi	Make Admin	Remove Member

לחיצה על Remove Admin תוריד למשתמש/ת אפשרויות מנהל, Make Admin תתן למשתמש/ת סטטוס מנהל/ת.

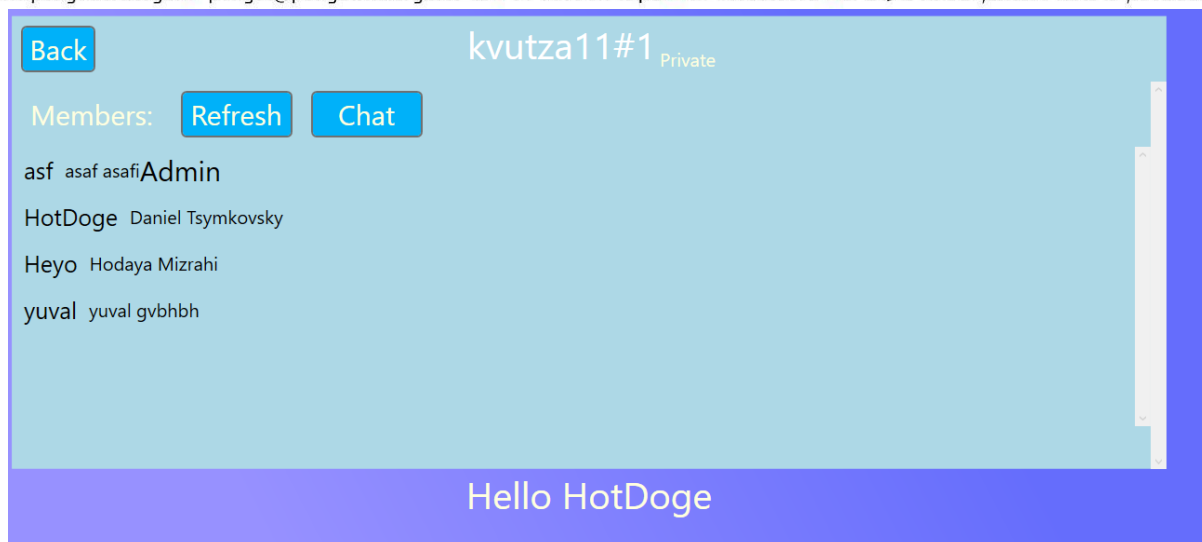
עריכה של שם קבוצה יעשה בלחיצת על הכפתור Edit שליד שם הקבוצה.

kvutza11#1 Private Edit

kvutza21 Private Save

לחיצה על Save תשמור את שם הקבוצה החדש.

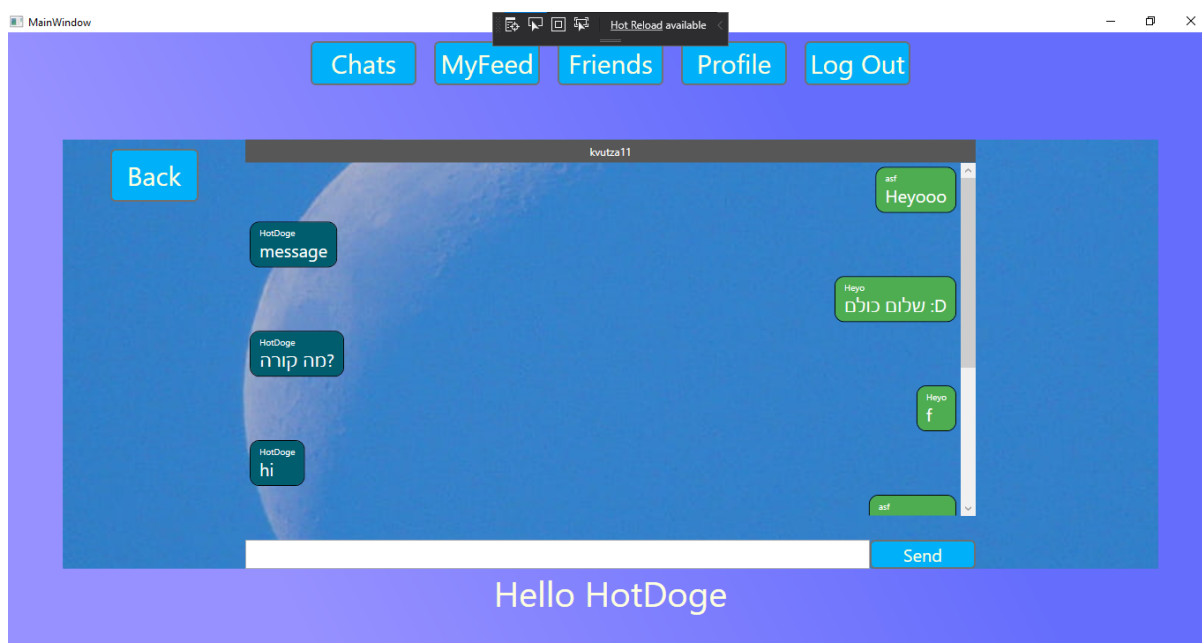
תפריט משתמש שאינו מנהל נראה כך:



לא ניתן לשנות מידע על הקבוצה. ניתן רק להיכנס לצ'אט בכפתור Chat או לרענן את המסך.

לחיצה על Back תצא מהחלון ותחזור לחלון הקודם.

חלון התכתבות:



שם הקבוצה ניתן לראות בראש ההתכתבות.

ניתן להכניס את תוכן ההודעה בשדה התחתון ליד הכפתור Send ולשלוח את ההודעה בלחיצה על Send.

בלחיצה על כפתור Back ניתן לחזור אחורה לחלון הקודם.

בלחיצה על MyFeed ניתן לחזור אל חלון הפוסטים שהמשתמש רוצה לראות:

עבור על פוסט שניתן לראות ניתן לכתוב תגובה, לתת לייק לפוסט שהמשתמש אהב, וללחוץ על כפתור view Comments כדי לראות את התגובות. אם הפוסט נכתב על ידי המשתמש, המשתמש יכול לערוך את תוכן הפוסט, ולשנות את ה"ציבוריות" שלו (רק חברים רואים או כולם).

לחיצה על refresh תרענן ותעדכן את תצוגת הפוסטים.

לחיצה על Add Post תיצור פוסט של המשתמש שניתן יהיה לראות אותו גם ב UserFeed שלו, וגם ב Feed של כל העוקבים שלו. במידה והפוסט פרטי רק החברים של אותו משתמש יראו את הפוסט.

בלחיצה על הוספת פוסט יפתח בתוך החלון את התצוגה הזו:

באזור הלבן ניתן להקליד את תוכן הפוסט, ניתן לבחור אם הוא ציבורי או פרטי, וללחוץ על create בשביל ליצור אותו. במידה והמשתמש התחרט ואינו רוצה ליצור פוסט, המשתמש ילחץ על cancel על מנת לסגור את התצוגה של יצירת פוסט.

בלחיצה על תגובות ניתן לראות את כל התגובות על הפוסט. תגובות שנכתבו על ידי המשתמש ניתן לערוך בכפתור Edit, המופיע רק עבור כותב התגובה:



Heyo UnFollow

Shalom anashimm

^^stop comments view^^

UnLike

2 likes

HotDoge Edit

Shalommmmm

חיפוש משתמש:

My Feed

Search user:

ניתן להקליד שם משתמש, ולמצוא משתמשים על פי האותיות הראשונות של שם המשתמש.

בלחיצה על שם המשתמש, יפתח חלון של מידע על המשתמש, והפוסטים שאותו משתמש/ת כתב/ה.

UserFeed: חלון הפוסטים והמידע על המשתמש

User's Feed

User Name: Heyo  
Name: Hodaya Mizrahi  
Description: Hola Hevre

Heyo UnFollow

Shalom anashimm

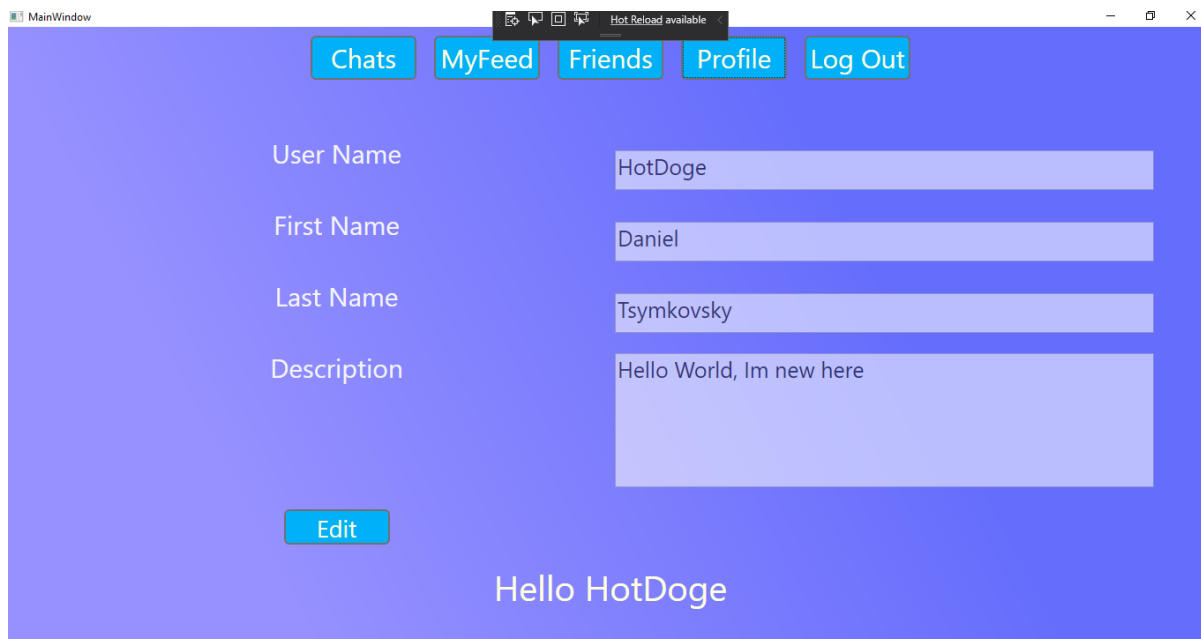
2 likes

Hello HotDoge

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
בחלון זה ניתן לראות את כל הפוסטים שנוצרו על ידי המשתמש המוצג, שנגישים למשתמש הצופה אותם. ניתן "לעקוב" אחרי המשתמש כדי לראות את הפוסטים שלו בFeed, ולבקש/לאשר/לדחות/לבטל בקשת חברות אל אותו משתמש.

ניתן לראות בחלון זה מידע על המשתמש.

חלון פרופיל:

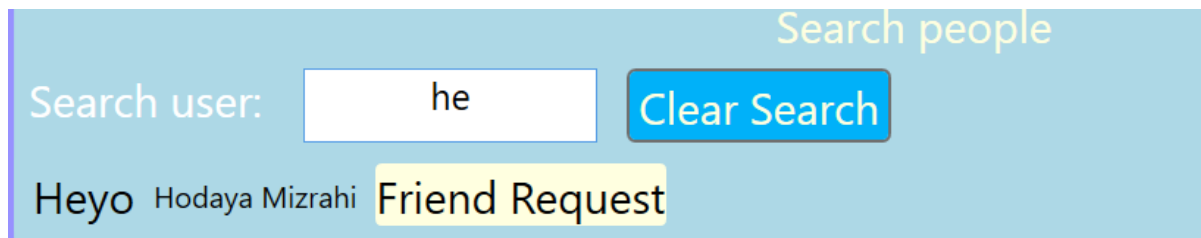


ניתן לראות את פרופיל של המשתמש - המידע עליו, על ידי לחיצה על Profile.

ניתן לערוך אותו בלחיצה על edit ולשמור בלחיצה על save.

Friends:

בלחיצה על Friends ניתן לראות את החברים של אותו משתמש. לאשר את האנשים שרוצים להיות חברים של המשתמש בPending, ולשלוח בקשות חברות על ידי חיפוש המשתמש בהזנת שם משתמש בשורת החיפוש



## נספחים:

## צד שרת:

HOST:

App.Xaml:

```
<Application x:Class="Host.App"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    xmlns:local="clr-namespace:Host"
```

```
    StartupUri="MainWindow.xaml">
```

```
<Application.Resources>
```

```
</Application.Resources>
```

```
</Application>
```

MainWindow.Xaml:

```
<Window x:Class="Host.MainWindow"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Host"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid>

</Grid>

</Window>

MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using WcfServiceWCF;

namespace Host

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

ServiceHost srv = new ServiceHost(typeof(WcfServiceWCF.Service1));

srv.Open();

}

}

}

Base.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

## namespace Model

```
{  
  
    public abstract class BaseEntity  
    {  
  
    }  
  
    public class EntityList : BaseEntity  
    {  
  
        private List<BaseEntity> entities;  
  
        public EntityList()  
        {  
            this.Entities = new List<BaseEntity>();  
        }  
  
        public EntityList(List<BaseEntity> entities)  
        {  
            this.entities = entities;  
        }  
  
        public List<BaseEntity> Entities { get => entities; set => entities = value; }  
    }  
}
```

## Comment.cs

```
using System;
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Model

{

public class Comment : BaseEntity

{

public User User { get => user; set => user = value; }

public Post Post { get => post; set => post = value; }

public string Content { get => content; set => content = value; }

public int ID { get => iD; set => iD = value; }

private int iD;

private User user;

private Post post;

private string content;

public Comment(User user, Post post, string content)

{

this.user = user;

this.post = post;

this.content = content;



```
public Comment() { }
```

```
public Comment(int id, User user, Post post, string content)
```

```
{
```

```
    this.ID = id;
```

```
    this.user = user;
```

```
    this.post = post;
```

```
    this.content = content;
```

```
}
```

```
}
```

```
public class CommentList : List<Comment>
```

```
{
```

```
    public CommentList() { }
```

```
    public CommentList(IEnumerable<Comment> list) : base(list) { }
```

```
    public CommentList(IEnumerable<BaseEntity> list) :  
base(list.Cast<Comment>().ToList()) { }
```

```
}
```

```
}
```

```
DMessage.cs:
```

```
using System;
```

```
using System.Collections.Generic;
```

using System.Text;

using System.Threading.Tasks;

namespace Model

{

public class Comment : BaseEntity

{

public User User { get => user; set => user = value; }

public Post Post { get => post; set => post = value; }

public string Content { get => content; set => content = value; }

public int ID { get => iD; set => iD = value; }

private int iD;

private User user;

private Post post;

private string content;

public Comment(User user, Post post, string content)

{

this.user = user;

this.post = post;

this.content = content;

}

```
public Comment() { }
```

```

public Comment(int id, User user, Post post, string content)
{
    this.ID = id;

    this.user = user;

    this.post = post;

    this.content = content;
}
}
    
```

```

public class CommentList : List<Comment>
{
    public CommentList() { }

    public CommentList(IEnumerable<Comment> list) : base(list) { }

    public CommentList(IEnumerable<BaseEntity> list) :
base(list.Cast<Comment>().ToList()) { }

}
}
    
```

Follow.cs:

```

using System;

using System.Collections.Generic;

using System.Linq;
    
```

using System.Threading.Tasks;

namespace Model

{

public class Follow : BaseEntity

{

private User follower;

private User following;

public Follow(User follower, User following)

{

this.follower = follower;

this.following = following;

}

public Follow() { }

public User Follower { get => follower; set => follower = value; }

public User Following { get => following; set => following = value; }

}

public class FollowList : List<Follow>

{

public FollowList() { }

public FollowList(IEnumerable<Follow> list) : base(list) { }

```
public FollowList(IEnumerable<BaseEntity> list) :
    base(list.Cast<Follow>().ToList()) { }

    }

}
```

Friend.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Model

{

    public class Friend : BaseEntity

    {

        //properties

        private User user1; //first user who initiated friend request

        private User user2; //second user

        private bool approved; //is the friendship is approved by the second user.

        //constructors

        public Friend(User user1, User user2, bool approved)

        {

            this.user1 = user1;

            this.user2 = user2;
```

this.approved = approved;

}

public Friend() { this.approved = false; }

public Friend(User user1, User user2)

{

    this.user1 = user1;

    this.user2 = user2;

    this.approved = false;

}

//getters and setters

public User User1 { get => user1; set => user1 = value; }

public User User2 { get => user2; set => user2 = value; }

public bool Approved { get => approved; set => approved = value; }

}

public class FriendList : List<Friend>

{

    public FriendList() { }

    public FriendList(IEnumerable<Friend> list) : base(list) { }

    public FriendList(IEnumerable<BaseEntity> list) :  
base(list.Cast<Friend>().ToList()) { }

}

}

GMember.cs:

using System;

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Model

{

public class GMember : BaseEntity

{

private Group group;

private User user;

private bool admin;

public Group Group { get => group; set => group = value; }

public User User { get => user; set => user = value; }

public bool Admin { get => admin; set => admin = value; }

public GMember() { }

public GMember(Group group, User user, bool admin)

{

this.group = group;

this.user = user;

this.admin = admin;

}

}



```
public class GMemberList : List<GMember>

{

    public GMemberList() { }

    public GMemberList(IEnumerable<GMember> list) : base(list) { }

    public GMemberList(IEnumerable<BaseEntity> list) :
base(list.Cast<GMember>().ToList()) { }

}

}
```

GMessage.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Model

{

    public class GMessage : BaseEntity

    {

        private int iD;

        private Group group;

        private User user;

        private string content;

        public GMessage() { }
```

```
public GMessage(Group group, User user, string content)

{

    this.group = group;

    this.user = user;

    this.content = content;

}


public int ID { get => iD; set => iD = value; }

public Group Group { get => group; set => group = value; }

public User User { get => user; set => user = value; }

public string Content { get => content; set => content = value; }

}

public class GMessageList : List<GMessage>

{

    public GMessageList() { }

    public GMessageList(IEnumerable<GMessage> list) : base(list) { }

    public GMessageList(IEnumerable<BaseEntity> list) :
base(list.Cast<GMessage>().ToList()) { }

}

}
```

Group.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;
```

using System.Threading.Tasks;

namespace Model

{

public class Group : BaseEntity

{

//properties

private int iD; //מספר מזהה של הקבוצה

private string gNname; //שם הקבוצה

private string description; //תיאור הקבוצה

private bool publicity; //האם הקבוצה ציבורית או פרטית

//getters and setters

public int ID { get => iD; set => iD = value; }

public string GNname { get => gNname; set => gNname = value; }

public string Description { get => description; set => description = value; }

public bool Publicity { get => publicity; set => publicity = value; }

}

public class GroupList : List<Group>

{

public GroupList() { }

public GroupList(IEnumerable<Group> list) : base(list) { }

public GroupList(IEnumerable<BaseEntity> list) :  
base(list.Cast<Group>().ToList()) { }

}

Like.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace Model
```

```
{
```

```
    public class Like : BaseEntity
```

```
    {
```

```
        private Post post;
```

```
        private User user;
```

```
        public Like()
```

```
        { }
```

```
        public Like(Post post, User user)
```

```
        {
```

```
            this.post = post;
```

```
            this.user = user;
```

```
        }
```

```
        public Post Post { get => post; set => post = value; }
```

```
        public User User { get => user; set => user = value; }
```

```
    }
```

```
public class LikeList : List<Like>
{
    public LikeList() { }
    public LikeList(IEnumerable<Like> list) : base(list) { }
    public LikeList(IEnumerable<BaseEntity> list) : base(list.Cast<Like>().ToList()) {
    }
    }
}
```

Post.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

namespace Model

```
{
    public class Post : BaseEntity
    {
        private int iD;
        private User user;
        private string content;
        private bool isPrivate;
```

```
public int ID { get => iD; set => iD = value; }

public User User { get => user; set => user = value; }

public string Content { get => content; set => content = value; }

public bool IsPrivate { get => isPrivate; set => isPrivate = value; }

}

public class PostList : List<Post>

{

    public PostList() { }

    public PostList(IEnumerable<Post> list) : base(list) { }

    public PostList(IEnumerable<BaseEntity> list) : base(list.Cast<Post>().ToList())
    {}

    }

}
```

UnSeenGM.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;
```

namespace Model

```
{

    public class UnSeenGM : BaseEntity

    {
```

```

private GMessage message; //group message

private int user; // user ID to reduce memory usage

private int group; // group ID to reduce memory usage

//constructorss

public UnSeenGM() { }

public UnSeenGM(GMessage message, int user, int group)

{

    this.message = message;

    this.user = user;

    this.group = group;

}

//getters and setters

public GMessage Message { get => message; set => message = value; }

public int User { get => user; set => user = value; }

public int Group { get => group; set => group = value; }

}

public class UnSeenGMList : List<UnSeenGM>

{

    public UnSeenGMList() { }

    public UnSeenGMList(IEnumerable<UnSeenGM> list) : base(list) { }

    public UnSeenGMList(IEnumerable<BaseEntity> list) :
base(list.Cast<UnSeenGM>().ToList()) { }

}

}
    
```

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Text.RegularExpressions;
```

```
namespace Model
```

```
{

    /// <summary>

    /// Removes all non-alphanumeric characters from the input string.

    /// </summary>

    /// <param name="input">The string to remove non-alphanumeric characters
    from.</param>

    /// <returns>A new string that contains only alphanumeric characters.</returns>
```

```
public class User : BaseEntity
```

```
{

    //properties

    private int iD; // מספר מזהה של המשתמש

    private string email; // כתובת דוא"ל

    private string uname; // שם משתמש

    private string fname; // שם פרטי

    private string lname; // שם משפחה
```



private string description; //ביוגרפיה או תיאור של המשתמש

private DateTime entrDate; //תאריך יצירת משתמש

private string password; //סימת המשתמש

//getters and setters

public string Description { get => description; set => description = value; }

public DateTime EntrDate { get => entrDate; set => entrDate = value; }

public string Password { get => password; set => password = value; }

public int ID { get => iD; set => iD = value; }

public string Uname { get => uname; set => uname =  
RemoveNonAlphanumeric(value); }

public string Fname { get => fname; set => fname =  
RemoveNonAlphanumeric(value); }

public string Lname { get => lname; set => lname =  
RemoveNonAlphanumeric(value); }

public string Email { get => email; set => email = value; }

private static string RemoveNonAlphanumeric(string input)

{

// Define a regular expression that matches all non-alphanumeric characters

Regex regex = new Regex("[^a-zA-Z0-9]");

// Use the regular expression to remove all non-alphanumeric characters  
from the input string

string result = regex.Replace(input, "");

return result;

}

}

```
public class UserList : List<User>
{
    public UserList() { }
    public UserList(IEnumerable<User> list) : base(list) { }
    public UserList(IEnumerable<BaseEntity> list) : base(list.Cast<User>().ToList())
    {}
}
}
```

## ModelView

BaseDB.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.OleDb;
using Model;
using System.Text.RegularExpressions;
namespace ModelView
{
    public delegate string CreateSql(BaseEntity entity);
    public class ChangeEntity
    {

```

```
private BaseEntity entity;
```

```
private CreateSql createSql;
```

```
public ChangeEntity(CreateSql createSql, BaseEntity entity)
```

```
{
```

```
    this.entity = entity;
```

```
    this.createSql = createSql;
```

```
}
```

```
public BaseEntity Entity { get => entity; set => entity = value; }
```

```
public CreateSql CreateSql { get => createSql; set => createSql = value; }
```

```
}
```

```
public abstract class BaseDB
```

```
{
```

```
    //Properties of BaseDB;
```

```
    protected string connectionString;
```

```
    protected OleDbConnection connection;
```

```
    protected OleDbCommand command;
```

```
    protected OleDbDataReader reader;
```

```
    protected List<ChangeEntity> inserted = new List<ChangeEntity>();
```

```
    protected List<ChangeEntity> deleted = new List<ChangeEntity>();
```

```
    protected List<ChangeEntity> updated = new List<ChangeEntity>();
```

```
    //BaseDB Class abstract Methods:
```

```
    protected abstract BaseEntity NewEntity();
```

```
protected abstract BaseEntity CreateModel(BaseEntity entity);
```

```
public abstract void Insert(BaseEntity entity);
```

```
public abstract void Update(BaseEntity entity);
```

```
public abstract void Delete(BaseEntity entity);
```

```
protected abstract string CreateInsertSQL(BaseEntity entity);
```

```
protected abstract string CreateUpdateSQL(BaseEntity entity);
```

```
protected abstract string CreateDeleteSQL(BaseEntity entity);
```

```
public BaseDB()
```

```
{
```

```
    connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\magshimim\Desktop\chatstreamwcf\ModelView\SocialNet.accdb;P
ersist Security Info=True";
```

```
    connection = new OleDbConnection(connectionString);
```

```
    command = new OleDbCommand();
```

```
}
```

```
/// <summary>
```

```
/// Removes all non-alphanumeric characters from the input string.
```

```
/// </summary>
```

```
/// <param name="input">The string to remove non-alphanumeric characters
from.</param>
```

```
/// <returns>A new string that contains only alphanumeric characters.</returns>
```

```
public static string RemoveNonAlphanumeric(string input)
```

```
{
```

```
    // Define a regular expression that matches all non-alphanumeric characters
```

```
Regex regex = new Regex("[^a-zA-Z0-9]");
```

// Use the regular expression to remove all non-alphanumeric characters from the input string

```
string result = regex.Replace(input, "");
```

```
return result;
```

```
}
```

```
/// <summary>
```

/// Sanitizes the input string for safe use in database queries to prevent SQL injection attacks.

```
/// </summary>
```

```
/// <param name="original">The string to sanitize.</param>
```

/// <returns>A sanitized string that can be safely used in database queries.</returns>

```
protected string ToSqlStr(string original)
```

```
{
```

```
    StringBuilder builder = new StringBuilder(original);
```

```
    builder.Replace("", "");
```

```
    builder.Replace("\\", "\\");
```

```
    return builder.ToString();
```

```
}
```

```
protected List<BaseEntity> Select()
```

```
List<BaseEntity> list = new List<BaseEntity>(); // create list to store the
results of the query
```

```
try
```

```
{
```

```
    command.Connection = connection; //adds the database connection to the
command
```

```
    //command text
```

```
    connection.Open(); //opens the connection
```

```
    reader = command.ExecuteReader(); //executes the query
```

```
    while (reader.Read()) //for each result:
```

```
    {
```

```
        BaseEntity entity = NewEntity(); //create the entity to store the
information.
```

```
        list.Add(CreateModel(entity)); // store the result in the entity created.
```

```
    }
```

```
}
```

```
catch (Exception e)
```

```
{
```

```
    //show the error in case of error
```

```
    System.Diagnostics.Debug.WriteLine(e.Message + "\nSQL: " +
command.CommandText);
```

```
}
```

```
finally
```

```
{
```

```
    //close the reader and the connection in the end.
```

```
    if (reader != null)
```

```

reader.Close();

if (connection.State == System.Data.ConnectionState.Open)

    connection.Close();

}

return list;

}

public int SaveChanges()

{

    int records = 0;

    try

    {

        command.Connection = this.connection; //add consnection reference to
query

        connection.Open(); //open connection

        foreach (var item in updated)

        {

            command.CommandText = item.CreateSql(item.Entity); //receive string
of the query

            records = command.ExecuteNonQuery(); // execute update query

        }

        foreach (var item in inserted)

        {

            command.CommandText = item.CreateSql(item.Entity); //receive string
of the query

            records = command.ExecuteNonQuery(); // execute update query

            if (item.Entity is User)

```

```

        User user = (User)item.Entity;

        command.CommandText = "Select @@Identity"; //if the entity is user,
        get it's identity property

        user.ID = (int)command.ExecuteScalar(); // receive and add property
    }

    if (item.Entity is GMessage)
    {

        GMessage message = (GMessage)item.Entity;

        command.CommandText = "Select @@Identity"; //if the entity is user,
        get it's identity property

        message.ID = (int)command.ExecuteScalar(); // receive and add
        property
    }

    if(item.Entity is Model.Group)
    {

        Model.Group group = (Model.Group)item.Entity;

        command.CommandText = "Select @@Identity"; //if the entity is user,
        get it's identity property

        group.ID = (int)command.ExecuteScalar(); // receive and add property
    }
}

foreach (var item in deleted)
{

    command.CommandText = item.CreateSql(item.Entity); //receive string
    of the query

    records = command.ExecuteNonQuery(); // execute update query

```



```

    }

    catch (Exception e)

    {

        System.Diagnostics.Debug.WriteLine(e.Message + "\nSQL:" +
        command.CommandText); //show error if occurred

    }

    finally

    {

        //clear queries lists:

        inserted.Clear();

        updated.Clear();

        deleted.Clear();

        //close connection:

        if (connection.State == System.Data.ConnectionState.Open)

            connection.Close();

    }

    return records;

}

}

}
    
```

CommentDB.cs:

using Model;

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
 using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ModelView

{

public class CommentDB : BaseDB

{

public override void Delete(BaseEntity entity)

{

Comment comment = entity as Comment;

if (comment != null)

{

deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));

}

}

public CommentList SelectByPost(Post post)

{

command.CommandText = string.Format("SELECT \* FROM CommentTbl  
WHERE [Post] = {0} ", post.ID);

return new CommentList(base.Select());

}

```
public override void Insert(BaseEntity entity)
{
    Comment comment = entity as Comment;
    if (comment != null)
    {
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
    }
}

public override void Update(BaseEntity entity)
{
    Comment comment = entity as Comment;
    if (comment != null)
    {
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
    }
}

protected override string CreateDeleteSQL(BaseEntity entity)
{
    Comment comment = entity as Comment;

    StringBuilder sql_builder = new StringBuilder();

    sql_builder.AppendFormat("DELETE FROM CommentTbl WHERE ID={0}",
comment.ID);
```

```
return sql_builder.ToString();
```

```
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
```

```
{
```

```
    Comment comment = entity as Comment;
```

```
    return $"INSERT INTO CommentTbl ([User], [Post], [Content])  
VALUES({comment.User.ID}, {comment.Post.ID}, '{ToSqlStr(comment.Content)})'";
```

```
}
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
```

```
{
```

```
    UserDB userDB = new UserDB();
```

```
    PostDB postDB = new PostDB();
```

```
    Comment comment = entity as Comment;
```

```
    comment.ID = (int)reader["ID"];
```

```
    comment.User = userDB.SelectByID((int)reader["User"]);
```

```
    comment.Post = postDB.SelectByID((int)reader["Post"]);
```

```
    comment.Content = (string)reader["Content"];
```

```
    return comment;
```

```
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
```

Comment comment = entity as Comment;

```
return $"UPDATE CommentTbl SET
[Content]='{ToSqlStr(comment.Content)}' WHERE ID={comment.ID}";

}
```

```
protected override BaseEntity NewEntity()
{
    return new Comment() as BaseEntity;
}

}

}
```

DMessageDB.cs:

```
using Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

namespace ModelView

```
{
```

```
public class DMessageDB : BaseDB
```

```

public override void Delete(BaseEntity entity)
{
    DMessage message = entity as DMessage;
    if (message != null)
    {
        deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
    }
}
    
```

```

public override void Insert(BaseEntity entity)
{
    DMessage message = entity as DMessage;
    message.Seen = false;
    if (message != null)
    {
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
    }
}
    
```

```

public override void Update(BaseEntity entity)
{
    DMessage message = entity as DMessage;
    if (message != null)
    {
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
    }
}
    
```

```

    }

    }

    public DMessage SelectMessage(int ID)
    {
        command.CommandText = "SELECT * FROM DMessageTbl" +
            $" WHERE [ID] = {ID}";

        DMessageList list = new DMessageList(base.Select()); //recive results

        return list.Count > 0 ? list[0] : null; // return chosen value by condition if true
        return list[0] else return null
    }

    private bool IsMessagePermition(User main, int ID)
    {
        command.CommandText = "SELECT * FROM DMessageTbl" +
            $" WHERE [ID] = {ID} AND ([DestUser] = {main.ID} OR [SourceUser] = {main.ID}) ";

        DMessageList list = new DMessageList(base.Select());

        return list.Count > 0;
    }

    private bool IsUserInUist(UserList list, User userToAdd)
    {
        { //the method gets UserList and user wanted to add and returns true if the user
        is already in the list

        // else false
    }

```

foreach (User userDM in list)

{

if (userToAdd.ID == userDM.ID )

return true;

}

return false;

}

/// <summary>

/// the method gets the current user and returns all the chats

/// with that user.

/// </summary>

/// <param name="user"></param>

/// <returns>User's list</returns>

public UserList GetAllDMChats(User user)

{

DMessageList sourceUsers = null;

DMessageList destUsers = null;

UserList users = new UserList();

command.CommandText = "SELECT \* FROM DMessageTbl" +

" WHERE [DestUser] = {user.ID}"; //get all Messages that the main user  
recived sql

sourceUsers = new DMessageList(base.Select());

command.CommandText = "SELECT \* FROM DMessageTbl" +

" WHERE [SourceUser] = {user.ID}"; //get all Messages that the main  
user sent sql

destUsers = new DMessageList(base.Select());



foreach (DMessage userS in sourceUsers)//for each user that the main user  
recived

if (!(IsUserInUist(users, userS.SourceUser))) //if the second user is not in  
list

users.Add(userS.SourceUser); //add the user

foreach (DMessage userD in destUsers) //for each user that the main user  
recived

if (!(IsUserInUist(users, userD.DestUser)))//if the second user is not in list

users.Add(userD.DestUser);//add the user

return users;

}

//returns a value that symbolizes the state of all chats

//with that user the purpose is to check this update state value

//to know if the the chats list should be refreshed

public int GetChatsDMUpdateState(User user)

{

    UserList users = GetAllDMChats(user);

    int checksum = 0, count = 0;

    foreach (User userTemp in users)

    {

        checksum += (userTemp.ID \* (int)userTemp.Uname[0]);

        count++;

    }

    checksum \*= count;

}

public DMessageList SelectAllDMsChat(User main, User chattedU)

{

UpdateSeen(main, chattedU);

command.CommandText = "SELECT \* FROM DMessageTbl" +

\$" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID}) " +

\$"OR ([DestUser] = {chattedU.ID} AND [SourceUser] = {main.ID})";

return (new DMessageList(base.Select()));

}

public bool AreChatting(User main, User chattedU)

{

command.CommandText = "SELECT \* FROM DMessageTbl" +

\$" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID}) " +

\$"OR ([DestUser] = {chattedU.ID} AND [SourceUser] = {main.ID}) LIMIT  
1";

return (new DMessageList(base.Select())).Count > 0;

}

public DMessageList SelectUnSeen(User main, User chattedU)

{

```
command.CommandText = "SELECT * FROM DMessageTbl" +
```

```
$" WHERE ([DestUser] = {main.ID} AND [SourceUser] = {chattedU.ID}  
AND [Seen] = False) ";
```

```
DMessageList dMessages = new DMessageList(base.Select());
```

```
UpdateSeen(main, chattedU);
```

```
foreach (DMessage message in dMessages)
```

```
{
```

```
    message.Seen = true;
```

```
}
```

```
return dMessages;
```

```
}
```

```
private void UpdateSeen(User main, User chattedU)
```

```
{
```

```
    EntityList pair = new EntityList();
```

```
    pair.Entities.Add(chattedU);
```

```
    pair.Entities.Add(main);
```

```
    updated.Add(new ChangeEntity(this.CreateUpdateSeenSQL, pair));
```

```
    this.SaveChanges();
```

```
}
```

```
protected override string CreateDeleteSQL(BaseEntity entity)
```

```
{
```

```
    DMessage message = entity as DMessage;
```

```
    StringBuilder sql_builder = new StringBuilder();
```

```
    sql_builder.AppendFormat("DELETE FROM DMessageTbl WHERE ID={0}",  
message.ID);
```

```
return sql_builder.ToString();
```

```
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
```

```
{
```

```
    DMessage message = entity as DMessage;
```

```
    return string.Format("INSERT INTO DMessageTbl ([SourceUser],  
[DestUser]," +
```

```
        " [Content], [Time], [Seen]) VALUES( {0}, {1}, '{2}', '{3}', {4})",
```

```
        message.SourceUser.ID, message.DestUser.ID,  
ToSqlStr(message.Content),
```

```
        DateTime.Now.ToString(), false.ToString());
```

```
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
```

```
    DMessage message = entity as DMessage;
```

```
    return $"UPDATE DMessageTbl SET [Content]='{message.Content}',  
[Seen]='{message.Seen}'" +
```

```
        $" WHERE ID={message.ID}";
```

```
}
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
```

```
{
```

```
UserDB userDB = new UserDB();
```

```
DMessage message = entity as DMessage;
```

```
message.ID = (int)reader["ID"];
```

```
message.DestUser = userDB.SelectByID((int)reader["DestUser"]);
```

```
message.SourceUser= userDB.SelectByID((int)reader["SourceUser"]);
```

```
message.Content = (string)reader["Content"];
```

```
message.Seen = (bool)reader["Seen"];
```

```
message.Time =(DateTime)reader["Time"];
```

```
return message;
```

```
}
```

```
protected string CreateUpdateSeenSQL(BaseEntity entity)
```

```
{
```

```
EntityList entityList = entity as EntityList;
```

```
User source = (User)entityList.Entities[0];
```

```
User dest = (User)entityList.Entities[1];
```

```
return $"UPDATE DMessageTbl SET [Seen]= True WHERE [DestUser]=  
{dest.ID} AND [SourceUser] = {source.ID}";
```

```
}
```

```
protected override BaseEntity NewEntity()
```

```
{
```

}

}

}

FollowDB.cs:

using Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ModelView

{

public class FollowDB : BaseDB

{

public override void Delete(BaseEntity entity)

{

Follow follow = entity as Follow;

if (follow != null)

{

deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));

```

    }

    public override void Insert(BaseEntity entity)
    {
        Follow follow = entity as Follow;

        if (follow != null)
        {
            inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
        }
    }

    public FollowList SelectFollowed(User follower)
    {
        command.CommandText = string.Format("SELECT * FROM FollowTbl
        WHERE User = {0}", follower.ID);

        return new FollowList(base.Select());
    }

    public bool IsFollowing (int follower, int followed)
    {
        command.CommandText = string.Format("SELECT * FROM FollowTbl
        WHERE User = {0} AND Followed = {1} ", follower, followed);

        return (new FollowList(base.Select())).Count > 0;
    }

    public bool IsFollowing(Follow follow)
    {
        return IsFollowing(follow.Follower.ID, follow.Following.ID);
    }

```

}

public override void Update(BaseEntity entity)

{

Follow follow = entity as Follow;

if (follow != null)

{

updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));

}

}

protected override string CreateDeleteSQL(BaseEntity entity)

{

Follow follow = entity as Follow;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM FollowTbl WHERE [User]={0}  
AND [Followed]={1}", follow.Follower.ID, follow.Following.ID);

return sql\_builder.ToString();

}

protected override string CreateInsertSQL(BaseEntity entity)

{

Follow follow = entity as Follow;

return string.Format("INSERT INTO FollowTbl ([User], [Followed])  
VALUES({0}, {1})",

follow.Follower.ID, follow.Following.ID);

}



```
protected override BaseEntity CreateModel(BaseEntity entity)
{
    UserDB userDB = new UserDB();

    Follow follow = entity as Follow;

    follow.Follower = userDB.SelectByID((int)reader["User"]);

    follow.Following = userDB.SelectByID((int)reader["Followed"]); ;

    return follow;
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
{
    return "";
}
```

```
protected override BaseEntity NewEntity()
{
    return new Follow() as BaseEntity;
}

}
```

FriendDB.cs:

using System;

using System.Collections.Generic;

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Model;

namespace ModelView

{

public class FriendDB : BaseDB

{

public override void Delete(BaseEntity entity)

{

Friend friend = entity as Friend;

if (friend != null)

{

deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));

}

}

public override void Insert(BaseEntity entity)

{

Friend friend = entity as Friend;

if (friend != null)

{

inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));

}

```

    }

    public FriendList SelectRecivedReq (User user)

    {

        command.CommandText = string.Format("SELECT * FROM FriendTbl
WHERE [User2] = {0} AND [Approved] = false", user.ID);

        return new FriendList(base.Select());

    }

    public FriendList SelectUserFriends(User user)

    {

        command.CommandText = string.Format("SELECT * FROM FriendTbl
WHERE ([User2] = {0} OR [User1] = {0}) AND [Approved] = True", user.ID);

        return new FriendList(base.Select());

    }

    public bool AreFriends(int user1, int user2)

    {

        command.CommandText = string.Format("SELECT * FROM FriendTbl
WHERE (([User1] = {0} AND [User2] = {1}) OR ([User1] = {1} AND [User2] = {0}))
AND [Approved] = true", user1, user2);

        return (new FriendList(base.Select())).Count > 0;

    }

    public Friend SelectFriend(User user1, User user2)

    {

        command.CommandText = string.Format("SELECT * FROM FriendTbl
WHERE ([User1] = {0} AND [User2] = {1}) OR ([User1] = {1} AND [User2] = {0})",
user1.ID, user2.ID);

        FriendList results = new FriendList(base.Select());

        if (results.Count() > 0)
    
```

```

    {

        return results[0];

    }

    else

        return null;

    }

    public bool AreFriends(Friend friend)

    {

        return AreFriends(friend.User1.ID, friend.User2.ID);

    }

    public override void Update(BaseEntity entity)

    {

        Friend friend = entity as Friend;

        if (friend != null)

        {

            updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));

        }

    }

    protected override string CreateDeleteSQL(BaseEntity entity)

    {

        Friend friend = entity as Friend;

        StringBuilder sql_builder = new StringBuilder();

        sql_builder.AppendFormat("DELETE FROM FriendTbl WHERE ([User1]={0} AND [User2]={1}) OR ([User1]={1} AND [User2]={0})", friend.User1.ID, friend.User2.ID);
    
```

```
return sql_builder.ToString();
```

```
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
```

```
{
```

```
    Friend friend = entity as Friend;
```

```
    return string.Format("INSERT INTO FriendTbl ([User1], [User2], [Approved])  
VALUES({0}, {1}, false)",
```

```
        friend.User1.ID, friend.User2.ID);
```

```
}
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
```

```
{
```

```
    UserDB userDB = new UserDB();
```

```
    Friend friend = entity as Friend;
```

```
    friend.User1 = userDB.SelectByID((int)reader["User1"]);
```

```
    friend.User2 = userDB.SelectByID((int)reader["User2"]);
```

```
    friend.Approved = (bool)reader["Approved"];
```

```
    return friend;
```

```
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
```

```
    Friend friend = entity as Friend;
```

```
return $"UPDATE FriendTbl SET [Approved]={friend.Approved} WHERE
([User1]={friend.User1.ID} AND [User2]={friend.User2.ID}) OR
([User1]={friend.User2.ID} AND [User2]={friend.User1.ID})";

}
```

```
protected override BaseEntity NewEntity()
{
    return new Friend() as BaseEntity;
}
}
```

GMemberDB.cs:

```
using Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

namespace ModelView

```
{
    public class GMemberDB : BaseDB
    {
        public override void Delete(BaseEntity entity)
        {
```

```
GMember gMember = entity as GMember;
```

```
if (gMember != null)
```

```
{
```

```
    deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
```

```
}
```

```
}
```

```
public override void Insert(BaseEntity entity)
```

```
{
```

```
    GMember gMember = entity as GMember;
```

```
    if (gMember != null)
```

```
    {
```

```
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
```

```
    }
```

```
}
```

```
public override void Update(BaseEntity entity)
```

```
{
```

```
    GMember gMember = entity as GMember;
```

```
    if (gMember != null)
```

```
    {
```

```
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
```

```
    }
```

```
}
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • פסגה 71, כרמיאל 2167164

```
protected override string CreateDeleteSQL(BaseEntity entity)
{
    GMember member = entity as GMember;

    StringBuilder sql_builder = new StringBuilder();

    sql_builder.AppendFormat("DELETE FROM GMemberTbl WHERE [User]=
{0} AND [Group]= {1}", member.User.ID, member.Group.ID);

    return sql_builder.ToString();
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
{
    GMember member = entity as GMember;

    return string.Format("INSERT INTO GMemberTbl ([User], [Group], [Admin])
VALUES( {0}, {1}, {2} )",
        member.User.ID, member.Group.ID, false.ToString());
}
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
{
    UserDB userDB = new UserDB();

    GroupDB groupDB = new GroupDB();

    GMember gMember = entity as GMember;

    gMember.User = userDB.SelectByID((int)reader["User"]);

    gMember.Group = groupDB.SelectGroup((int)reader["Group"]);
}
```



```
gMember.Admin = (bool)reader["Admin"];
```

```
return gMember;
```

```
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
```

```
    GMember member = entity as GMember;
```

```
    return $"UPDATE GMemberTbl SET [Admin]={member.Admin} WHERE  
[User]={member.User.ID} AND [Group]={member.Group.ID} ";
```

```
}
```

```
protected override BaseEntity NewEntity()
```

```
{
```

```
    return new GMember();
```

```
}
```

```
public GMemberList GetMembers(Group group)
```

```
{
```

```
    command.CommandText = "SELECT * FROM GMemberTbl" +
```

```
        $" WHERE [Group] = {group.ID}";
```

```
    return new GMemberList(base.Select());
```

```
}
```

```
public List<User> GetUserMembers(Group group)
```

```
{
```

```
command.CommandText = "SELECT * FROM GMemberTbl" +
```

```
$" WHERE [Group] = {group.ID}";
```

```
return new GMemberList(base.Select()).Select(member =>
member.User).ToList(); ;
```

```
}
```

```
public List<User> SelectAdmins(Group group)
```

```
{
```

```
command.CommandText = "SELECT * FROM GMemberTbl" +
```

```
$" WHERE [Group] = {group.ID} AND [Admin] = True";
```

```
GMemberList adminMembers = new GMemberList(base.Select());
```

```
return adminMembers.Select(member => member.User).ToList();
```

```
}
```

```
public List<User> SelectNonAdmins(Group group)
```

```
{
```

```
command.CommandText = "SELECT * FROM GMemberTbl" +
```

```
$" WHERE [Group] = {group.ID} AND [Admin] = False";
```

```
GMemberList adminMembers = new GMemberList(base.Select());
```

```
return adminMembers.Select(member => member.User).ToList();
```

```
}
```

```
public bool IsAdmin(GMember member)
```

```
{
```

```
command.CommandText = "SELECT * FROM GMemberTbl" +
```

```
$" WHERE [Group] = {member.Group.ID} AND [User] =
{member.User.ID}";
```

```
GMemberList members = new GMemberList(base.Select());
```

```
return members.Count() > 0 ? members[0].Admin : false;
```

```

    }

    public bool IsMember(GMember member)
    {
        command.CommandText = "SELECT * FROM GMemberTbl" +
            $" WHERE [Group] = {member.Group.ID} AND [User] = {member.User.ID}";

        GMemberList members = new GMemberList(base.Select());

        return members.Count() > 0;
    }

    public List<Group> GetUsersGroup(User user)
    {
        command.CommandText = "SELECT * FROM GMemberTbl" +
            $" WHERE [User] = {user.ID}";

        GMemberList members = new GMemberList(base.Select());

        return members.Select(member => member.Group).ToList();
    }

}
}

```

GMessageDB.cs:

```

using Model;

using System;

using System.Collections.Generic;

using System.Linq;

```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Text;

using System.Threading.Tasks;

namespace ModelView

{

public class GMessageDB : BaseDB

{

/// <summary>

/// deletes a group message

/// </summary>

/// <param name="entity">group message to delete</param>

public override void Delete(BaseEntity entity)

{

GMessage message = entity as GMessage;

if (message != null)

{

deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));

}

}

/// <summary>

/// adds a group message, also marks the message as unseen for each  
member except

/// the author

/// </summary>

/// <param name="entity">the GMessage to add to DataBase/param>

public override void Insert(BaseEntity entity)

{

GMessage message = entity as GMessage;

UnSeenGMDB unSeenGMDB = new UnSeenGMDB();

if (message != null)

{

inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));

SaveChanges();

unSeenGMDB.Insert(new UnSeenGM(message, 0, message.Group.ID));

unSeenGMDB.SaveChanges();

}

}

/// <summary>

/// updates(changes) a group message

/// </summary>

/// <param name="entity">the updated group message</param>

public override void Update(BaseEntity entity)

{

GMessage message = entity as GMessage;

if (message != null)

{

updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));

}

}

/// <summary>

/// makes sql to delete a group message

/// </summary>

/// <param name="entity">the GMessage to delete</param>

/// <returns>sql that deletes the message</returns>

protected override string CreateDeleteSQL(BaseEntity entity)

{

GMessage message = entity as GMessage;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM GMessageTbl WHERE ID= {0}",  
message.ID);

return sql\_builder.ToString();

}

/// <summary>

/// makes sql to inserts a group message

/// </summary>

/// <param name="entity">the GMessage to insert</param>

/// <returns>sql that inserts the message</returns>

protected override string CreateInsertSQL(BaseEntity entity)

{

GMessage message = entity as GMessage;

return string.Format("INSERT INTO GMessageTbl ([User], [Group],  
[Content]) VALUES( {0}, {1}, '{2}' )",

message.User.ID, message.Group.ID, ToSqlStr(message.Content));

}

/// <summary>

/// fills the feilds in the group message with the received information

/// from the database

/// </summary>

/// <param name="entity"></param>

/// <returns>filled entity</returns>

protected override BaseEntity CreateModel(BaseEntity entity)

{

UserDB userDB = new UserDB();

GroupDB groupDB = new GroupDB();

GMessage message = entity as GMessage;

message.ID = (int)reader["ID"];

message.User = userDB.SelectByID((int)reader["User"]);

message.Group = groupDB.SelectGroup((int)reader["Group"]);

message.Content = (string)reader["Content"];

return message;

}

/// <summary>

/// makes sql to updates a group message

/// </summary>

/// <param name="entity">the GMessage to update</param>

/// <returns>sql that updates the message</returns>

protected override string CreateUpdateSQL(BaseEntity entity)

{

GMessage message = entity as GMessage;

```
return $"UPDATE GMessageTbl SET
[Content]='{ToSqlStr(message.Content)}' WHERE [ID]={message.ID}";
```

```
}
```

```
//creates an empty GMessage object
```

```
protected override BaseEntity NewEntity()
```

```
{
```

```
return new GMessage() as BaseEntity;
```

```
}
```

```
/// <summary>
```

```
/// returns the Gmessage with the ID given
```

```
/// </summary>
```

```
/// <param name="ID">id of wanted message</param>
```

```
/// <returns>the group message</returns>
```

```
public GMessage SelectGMessageByID(int ID)
```

```
{
```

```
command.CommandText = "SELECT * FROM GMessageTbl" +
```

```
 $" WHERE [ID] = {ID}";
```

```
GMessageList list = new GMessageList(base.Select());
```

```
return list.Count > 0 ? list[0] : null;
```

```
}
```

```
/// <summary>
```

```
/// receives all the messages in a group. also marks the messages as seen
```

```
/// </summary>
```

```
/// <param name="user">the user in the group</param>
```



```
/// <param name="group">the group</param>
```

```
/// <returns>GMessageList of the messages</returns>
```

```
public GMessageList GetMessages(User user, Group group)
```

```
{
```

```
    UnSeenGMDB unSeenDB = new UnSeenGMDB();
```

```
    unSeenDB.Delete(new UnSeenGM(null, user.ID, group.ID));
```

```
    unSeenDB.SaveChanges();
```

```
    command.CommandText = "SELECT * FROM GMessageTbl" +
```

```
        $" WHERE [Group] = {group.ID}";
```

```
    return new GMessageList(base.Select());
```

```
}
```

```
}
```

```
}
```

GroupDB.cs:

```
using Model;
```

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace ModelView
```

```
public class GroupDB : BaseDB
```

```
{
```

```
    public override void Delete(BaseEntity entity)
```

```
    {
```

```
        Group group = entity as Group;
```

```
        if (group != null)
```

```
        {
```

```
            deleted.Add(new ChangeEntity(CreateDeleteAllUnSeenSQL, entity));
```

```
            deleted.Add(new ChangeEntity(CreateDeleteAllMessagesSQL, entity));
```

```
            deleted.Add(new ChangeEntity(CreateDeleteMembersSQL, entity));
```

```
            deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
```

```
        }
```

```
    }
```

```
    public string CreateDeleteAllUnSeenSQL(BaseEntity entity)
```

```
    {
```

```
        Group group = entity as Group;
```

```
        StringBuilder sql_builder = new StringBuilder();
```

```
        sql_builder.AppendFormat("DELETE FROM UnSeenGMTbl WHERE  
[Group]= {0}", group.ID);
```

```
        return sql_builder.ToString();
```

```
    }
```

```
    public string CreateDeleteAllMessagesSQL(BaseEntity entity)
```

```
    {
```

Group group = entity as Group;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM GMessageTbl WHERE [Group]  
= {0}", group.ID);

return sql\_builder.ToString();

}

public string CreateDeleteMembersSQL(BaseEntity entity)

{

Group group = entity as Group;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM GMemberTbl WHERE [Group]=  
{1}", group.ID);

return sql\_builder.ToString();

}

public override void Insert(BaseEntity entity)

{

Group group = entity as Group;

if (group != null)

{

inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));

}

}

public override void Update(BaseEntity entity)

{

Group group = entity as Group;

if (group != null)

{

updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));

}

}

protected override string CreateDeleteSQL(BaseEntity entity)

{

Group group = entity as Group;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM GroupTbl WHERE ID={0}",  
group.ID);

return sql\_builder.ToString();

}

protected override string CreateInsertSQL(BaseEntity entity)

{

Group group = entity as Group;

return string.Format("INSERT INTO GroupTbl ([GName], [Description],  
[NeedReq]) VALUES( '{0}', '{1}', {2} )",

ToSqlStr(group.GNname), ToSqlStr(group.Description),  
group.Publicity.ToString());

}

protected override BaseEntity CreateModel(BaseEntity entity)

```
{
    Group group = entity as Group;
    group.ID = (int)reader["ID"];
    group.Description = (string)reader["Description"];
    group.GNname = (string)reader["GNName"];
    try
    {
        object obj = reader["Publicity"];
        group.Publicity = (bool)obj;
    }
    catch (Exception e)
    {
        //show the error in case of error
        System.Diagnostics.Debug.WriteLine("expection: " + e.Message);
    }

    return group;
}

public Group SelectGroup(int ID)
{
    command.CommandText = $"SELECT * FROM GroupTbl WHERE
GroupTbl.ID = {ID} ";

    GroupList groups = new GroupList(base.Select());
    return groups.Count() > 0 ? groups[0] : null ;
}
```

```

public GroupList SelectGroupByGName(string gName)
{
    command.CommandText = $"SELECT * FROM GroupTbl WHERE
GroupTbl.GName LIKE '{ToSqlStr(gName)}%";

    GroupList groups = new GroupList(base.Select());

    return groups;
}

protected override string CreateUpdateSQL(BaseEntity entity)
{
    Group group = entity as Group;

    return $"UPDATE GroupTbl SET [GName]='{group.GNname}',
[Description]='{group.Description}', [Publicity]={group.Publicity} WHERE ID =
{group.ID}";

}

protected override BaseEntity NewEntity()
{
    return new Group() as BaseEntity;
}
}
}
}

```

LikeDB.cs:

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
 using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Model;

namespace ModelView

{

public class LikeDB : BaseDB

{

public override void Delete(BaseEntity entity)

{

Like like = entity as Like;

if (like != null)

{

deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));

}

}

public override void Insert(BaseEntity entity)

{

Like like = entity as Like;

if (like != null)

{

```

        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
    }
}

public override void Update(BaseEntity entity)
{
    Like like = entity as Like;

    if (like != null)
    {
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
    }
}

protected override string CreateDeleteSQL(BaseEntity entity)
{
    Like like = entity as Like;

    StringBuilder sql_builder = new StringBuilder();

    sql_builder.AppendFormat("DELETE FROM LikeTbl WHERE [User] = {0} AND [Post] = {1}", like.User.ID, like.Post.ID);

    return sql_builder.ToString();
}

protected override string CreateInsertSQL(BaseEntity entity)
{

```



Like like = entity as Like;

```
return string.Format("INSERT INTO LikeTbl ([User], [Post]) VALUES({0},
{1})",
```

```
like.User.ID, like.Post.ID);
```

```
}
```

protected override BaseEntity CreateModel(BaseEntity entity)

```
{
```

```
UserDB userDB = new UserDB();
```

```
PostDB postDB = new PostDB();
```

```
Like like = entity as Like;
```

```
like.User = userDB.SelectByID((int)reader["User"]);
```

```
like.Post = postDB.SelectByID((int)reader["Post"]);
```

```
return like;
```

```
}
```

public LikeList SelectByPost(Post post)

```
{
```

```
command.CommandText = "SELECT * FROM LikeTbl" +
```

```
$" WHERE [Post] = {post.ID}";
```

```
LikeList likeList = new LikeList(base.Select());
```

```
return likeList;
```

```
}
```

public int CountLikes(Post post)

```
{
```

```
return SelectByPost(post).Count;
```

```
}
```

```
public bool IsLiked(Like like)
```

```
{
```

```
    command.CommandText = "SELECT * FROM LikeTbl" +
```

```
        $" WHERE [Post]={like.Post.ID} AND [User]={like.User.ID}";
```

```
    LikeList likeList = new LikeList(base.Select());
```

```
    return likeList.Count > 0;
```

```
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
```

```
    return "";
```

```
}
```

```
protected override BaseEntity NewEntity()
```

```
{
```

```
    return new Like() as BaseEntity;
```

```
}
```

```
}
```

```
}
```

PostDB.cs:

```
using System;
```

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
 using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Model;

namespace ModelView

{

public class PostDB: BaseDB

{

protected override BaseEntity NewEntity()

{

return new Post() as BaseEntity;

}

public PostList SelectAll()

{

command.CommandText = "SELECT \* FROM PostTbl";

return new PostList(base.Select());

}

public PostList SelectMyFeed(User user)

{

FollowDB followDB = new FollowDB();

PostList posts = new PostList();

```
foreach (Follow follow in followDB.SelectFollowed(user))
```

```
{
    foreach(Post post in this.SelectUserFeed(user, follow.Following))
        posts.Add(post);
}
posts.Sort(delegate (Post post1, Post post2)
{
    return -post1.ID.CompareTo(post2.ID);
});
return posts;
}
```

```
protected override BaseEntity CreateModel(BaseEntity entity)
```

```
{
    UserDB userDB = new UserDB();
    Post post = entity as Post;
    post.ID = (int)reader["ID"];
    post.User = userDB.SelectById((int)reader["User"]);
    post.Content = (string)reader["Content"];
    post.IsPrivate = (bool)reader["Private"];
    return post;
}
```

```
public PostList SelectUserFeed(User mainUser, User req)
```

```
{
    FriendDB friendDB = new FriendDB();
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
if (mainUser.ID == req.ID || friendDB.AreFriends(mainUser.ID, req.ID))

```
command.CommandText = string.Format("SELECT * FROM PostTbl
WHERE [User] = {0}", req.ID);
```

```
else
```

```
command.CommandText = string.Format("SELECT * FROM PostTbl
WHERE [User] = {0} AND [Private] = false ", req.ID);
```

```
PostList posts = new PostList(base.Select());
```

```
posts.Sort(delegate (Post post1, Post post2)
```

```
{
```

```
return -post1.ID.CompareTo(post2.ID);
```

```
});
```

```
return posts;
```

```
}
```

```
public PostList SelectByUID(int ID)
```

```
{
```

```
command.CommandText = string.Format("SELECT * FROM PostTbl
WHERE User = {0} AND [Private] = false ", ID);
```

```
return new PostList(base.Select());
```

```
}
```

```
public PostList SelectByUser(User user)
```

```
{
```

```
return SelectByUID(user.ID);
```

```
}
```

```
public Post SelectByID(int id)
```

```
{
```

```
command.CommandText = "SELECT * FROM PostTbl" +
```

\$" WHERE PostTbl.ID = {id}";

```
PostList postList = new PostList(base.Select());
```

```
if (postList.Count == 1)
```

```
{
```

```
    return postList[0];
```

```
}
```

```
else
```

```
    return null;
```

```
}
```

```
public override void Insert(BaseEntity entity)
```

```
{
```

```
    Post post = entity as Post;
```

```
    if (post != null)
```

```
{
```

```
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
```

```
}
```

```
}
```

```
public override void Update(BaseEntity entity)
```

```
{
```

```
    Post post = entity as Post;
```

```
    if (post != null)
```

```
{
```

```
        updated.Add(new ChangeEntity(this.CreateUpdateSQL, entity));
```

```
}
```

```
}
```

```
public override void Delete(BaseEntity entity)
```

```
{
    User people = entity as User;
    if (people != null)
    {
        deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
    }
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
```

```
{
    Post post = entity as Post;
    return $"INSERT INTO PostTbl ([User], [Content], [Private])
VALUES({post.User.ID}, '{ToSqlStr(post.Content)}', {post.IsPrivate})";
}
```

```
protected override string CreateUpdateSQL(BaseEntity entity)
```

```
{
    Post post = entity as Post;
    return $"UPDATE PostTbl SET [Private]={post.IsPrivate},
[Content]='{ToSqlStr(post.Content)}' WHERE ID={post.ID}";
}
```

```
protected override string CreateDeleteSQL(BaseEntity entity)
```

```
{
    Post post = entity as Post;
```

```

        StringBuilder sql_builder = new StringBuilder();

        sql_builder.AppendFormat("DELETE FROM PostTbl WHERE ID={0}",
        post.ID);

        return sql_builder.ToString();

    }

}

}
    
```

UnSeenGMDB.cs:

```

using Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ModelView
{
    public class UnSeenGMDB : BaseDB
    {
        /// <summary>

        /// the method inserts the unseen entity into the database

        /// </summary>

        /// <param name="entity"></param>

        public override void Insert(BaseEntity entity)
    }
}
    
```



{

//DataBases needed for the insert

GMemberDB gMemberDB = new GMemberDB();

UnSeenGM unSeen = entity as UnSeenGM;

GMessageDB gMessageDB = new GMessageDB();

if (unSeen != null)

{

//get the group in order to make sure it exists and correct.

Group group =

gMessageDB.SelectGMessageByID(unSeen.Message.ID).Group;

unSeen.Group = group.ID;//add the group id to the unSeen object

List<User> users = gMemberDB.GetUserMembers(group);//get all the  
members if tge group

foreach (User user in users)

{

//if the user isn't the writer of the message, add

//an UnSeen for this user in order to make sure what messages the user

//hasn't received yet.

if (user.ID != unSeen.User)

{

UnSeenGM temp = new UnSeenGM(unSeen.Message, user.ID,  
group.ID);

inserted.Add(new ChangeEntity(this.CreateInsertSQL, temp));

}

}

```

    }

}

/// <summary>
/// the method deletes an UnSeeb entity from the UnSeenGMTbl Database
/// in order to mark that the user saw the message
/// </summary>
/// <param name="entity"></param>
public override void Delete(BaseEntity entity)
{
    UnSeenGM unSeen = entity as UnSeenGM;
    if (unSeen != null)
    {
        deleted.Add(new ChangeEntity(this.CreateDeleteSQL, entity));
    }
}

public override void Update(BaseEntity entity)
{
    UnSeenGM unSeen = entity as UnSeenGM;
    if (unSeen != null)
    {

```

```

    }

}

/// <summary>

/// the method gets all the messages that the user in the group hasn't seen yet

/// </summary>

/// <param name="user"> the user that want to receive unseen
messages</param>

/// <param name="group">the group that the user in</param>

/// <returns>list of GMessages that the user hasn't received yet</returns>
public List<GMessage> GetChatsUnSeenMessages(User user, Group group)
{

    command.CommandText = "SELECT * FROM UnSeenGMTbl" +
        $" WHERE [Group] = {group.ID} AND [USER] = {user.ID}" ;

    List<GMessage> messages =(new
UnSeenGMList(base.Select())).Select(unseen => unseen.Message).ToList();

    Delete(new UnSeenGM(null, user.ID, group.ID)); //delete all messages that
the user receives from unseen database

    SaveChanges();//save the changes in the group unseen messages database

    return messages;

}

/// <summary>

/// makes sql that deletes every message that certain user seen from the list of
didn't unseen messages

```

/// <param name="entity">an unseen message with user and group fluids are used</param>

/// <returns>sql string that deletes all seen messages</returns>

protected override string CreateDeleteSQL(BaseEntity entity)

{

UnSeenGM unSeen = entity as UnSeenGM;

StringBuilder sql\_builder = new StringBuilder();

sql\_builder.AppendFormat("DELETE FROM UnSeenGMTbl WHERE [User]={0} AND [Group]= {1}", unSeen.User, unSeen.Group);

return sql\_builder.ToString();

}

/// <summary>

/// makes sql string that inserts a message to an unseen group messages database

/// in order to mark a message as unseen by user

/// </summary>

/// <param name="entity"></param>

/// <returns></returns>

protected override string CreateInsertSQL(BaseEntity entity)

{

UnSeenGM unSeen = entity as UnSeenGM;

return string.Format("INSERT INTO UnSeenGMTbl ([User], [Message], [Group]) VALUES( {0}, {1}, {2} )",

unSeen.User, unSeen.Message.ID, unSeen.Group);

}

/// <summary>

```

/// receives data from database about unseen message by user

/// </summary>

/// <param name="entity">An empty UnSeenGM object pointer that the method
fills </param>

/// <returns>filled recieved entity entity</returns>

protected override BaseEntity CreateModel(BaseEntity entity)
{
    UnSeenGM unSeen = entity as UnSeenGM;

    GMessageDB gMessageDB = new GMessageDB();

    unSeen.Message =
gMessageDB.SelectGMessageByID((int)reader["Message"]);

    unSeen.User = (int)reader["User"];

    unSeen.Group = (int)reader["Group"];

    return unSeen;
}

protected override string CreateUpdateSQL(BaseEntity entity)
{
    throw new NotImplementedException();
}

/// method returns an empty entity of UnSeenGM

protected override BaseEntity NewEntity()
{
    return new UnSeenGM();
}

```

}

UserDB.cs:

using Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ModelView

{

public class UserDB : BaseDB

{

protected override BaseEntity NewEntity()

{

return new User() as BaseEntity;

}

protected override BaseEntity CreateModel(BaseEntity entity)

{

User user = entity as User;

user.ID = (int)reader["ID"];

user.Fname = (string)reader["Fname"];

```
user.Lname = (string)reader["Lname"];
```

```
user.Uname = (string)reader["Uname"];
```

```
user.Description = (string)reader["Description"];
```

```
if (user.Description == null)//if user doesn't have description
```

```
    user.Description = "";
```

```
user.EntrDate = (DateTime)reader["EntrDate"];
```

```
user.Password = (string)reader["Password"];
```

```
user.Email = (string)reader["Email"];
```

```
return user;
```

```
}
```

```
public override void Insert(BaseEntity entity)
```

```
{
```

```
    User people = entity as User;
```

```
    if (people != null)
```

```
    {
```

```
        inserted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
```

```
    }
```

```
}
```

```
public override void Update(BaseEntity entity)
```

```
{
```

```
    User people = entity as User;
```

```
    if (people != null)
```

```
    {
```

```
}
```

```
}
```

```
public override void Delete(BaseEntity entity)
```

```
{
```

```
    User people = entity as User;
```

```
    if (people != null)
```

```
    {
```

```
        deleted.Add(new ChangeEntity(this.CreateInsertSQL, entity));
```

```
    }
```

```
}
```

```
protected override string CreateInsertSQL(BaseEntity entity)
```

```
{
```

```
    User user = entity as User;
```

```
    return string.Format("INSERT INTO UserTbl (Uname, FName, LName,  
EntrDate, [Password], Description, Email)" +
```

```
        " VALUES('{0}', '{1}', '{2}', '{3}', '{4}', '{5}', '{6}')",
```

```
        RemoveNonAlphanumeric(user.Uname),  
        RemoveNonAlphanumeric(user.Fname), RemoveNonAlphanumeric(user.Lname),
```

```
        user.EntrDate.ToString(), ToSqlStr(user.Password),  
        ToSqlStr(user.Description), ToSqlStr(user.Email));
```

```
}
```

```
protected override string CreateDeleteSQL(BaseEntity entity)
```

```
{
```

```
    User user = entity as User;
```



```

StringBuilder sql_builder = new StringBuilder();

sql_builder.AppendFormat("DELETE FROM UserTbl WHERE [Uname]={0}
AND [Password] = {1}", ToSqlStr(user.Uname), ToSqlStr(user.Password));

return sql_builder.ToString();

}

protected override string CreateUpdateSQL(BaseEntity entity)
{
    User user = entity as User;

    return $"UPDATE UserTbl SET
Uname='{RemoveNonAlphanumeric(user.Uname)}',
FName='{RemoveNonAlphanumeric(user.Fname)}', " +

        $" LName='{RemoveNonAlphanumeric(user.Lname)}', " +

        $" Description='{ToSqlStr(user.Description)}',
[Password]='{ToSqlStr(user.Password)}' WHERE ID={user.ID}";

}

public UserList SelectAll()
{
    command.CommandText = "SELECT * FROM UserTbl";

    return new UserList(base.Select());

}

public UserList SelectUserData(string Uname, string Password)
{
    command.CommandText = string.Format("SELECT * FROM UserTbl
WHERE Uname = '{0}' AND [Password] = '{1}' ", ToSqlStr(Uname),
ToSqlStr(Password));

    return new UserList(base.Select());

}

```

public UserList SelectByUName(string Uname) //select all usernames starts  
with the Uname

```
{
    command.CommandText = string.Format("SELECT * FROM UserTbl
WHERE [Uname] LIKE '{0}%' ", ToSqlStr(Uname));

    UserList userL = new UserList(base.Select());

    foreach (User user in userL)//for each user remove sensitive information.
    {
        user.Password = "";
        user.Email = "";
    }

    return userL;
}
```

public UserList SelectUNameExist(string Uname) //select the User with the  
exact UserName

```
{
    command.CommandText = string.Format("SELECT * FROM UserTbl
WHERE [Uname] = '{0}' ", ToSqlStr(Uname));

    UserList userL = new UserList(base.Select());

    foreach (User user in userL)//for each user remove sensitive information.
    {
        user.Password = "";
        user.Email = "";
    }

    return userL;
}
```

public User SelectByID(int ID) //select user by id

```

        command.CommandText = string.Format("SELECT * FROM UserTbl
WHERE ID = {0}", ID);

        UserList userL = new UserList(base.Select());

        foreach (User user in userL)//for each user remove sensitive information.
        {
            user.Password = "";

            user.Email = "";
        }

        return userL.Count > 0 ? userL[0] : null;
    }
}

```

## WCFServiceWCF:

IService1.cs:

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.ServiceModel;

using System.Text;

using Model;

using ModelView;

```

{

[ServiceContract]

public interface IService1

{

[OperationContract]

string GetData(int value);

[OperationContract]

CompositeType GetDataUsingDataContract(CompositeType composite);

[OperationContract]

void DeleteComment(Comment entity);

[OperationContract]

void InsertComment(Comment entity);

[OperationContract]

void UpdateComment(Comment entity);

[OperationContract]

List<Comment> SelectCommentByPost(Post post);

[OperationContract]

User InsertUser(User entity);

[OperationContract]

User Login(string userName, string password);

[OperationContract]

User SelectByUserID(int ID);

[OperationContract]

List<User> SelectUserByUName(string Uname);

[OperationContract]

void UpdateUser(User user);

[OperationContract]

void DeleteUser(string userName, string password);

[OperationContract]

User SignUp(User user);

[OperationContract]

Follow InsertFollow(Follow follow);

[OperationContract]

void DeleteFollow(Follow follow);

[OperationContract]

List<Follow> SelectFollowed(User follower);

[OperationContract]

bool IsFollowing(Follow follower);

[OperationContract]

Friend InsertFriend(Friend friend);

[OperationContract]

void DeleteFriend(Friend friend);

[OperationContract]

bool AreFriends(Friend friend);

[OperationContract]

Friend SelectFriend(User user1, User user2);

[OperationContract]

```
List<Friend> SelectRecivedReq(User user);
```

```
[OperationContract]
```

```
List<Friend> SelectUserFriends(User user);
```

```
[OperationContract]
```

```
void ApproveFriend(Friend friend);
```

```
[OperationContract]
```

```
Like InsertLike(Like like);
```

```
[OperationContract]
```

```
void DeleteLike(Like like);
```

```
[OperationContract]
```

```
int CountLikes(Post post);
```

```
[OperationContract]
```

```
bool IsLiked(Like like);
```

```
[OperationContract]
```

```
Post InsertPost(Post post);
```

```
[OperationContract]
```

```
void DeletePost(Post post);
```

```
[OperationContract]
```

```
void UpdatePost(Post post);
```

```
[OperationContract]
```

```
List<Post> SelectUserFeed(User mainUser, User req);
```

```
[OperationContract]
```

```
List<Post> SelectMyFeed(User user);
```

```
[OperationContract]
```

```
List<Post> SelectPostsByUser(User user);
```

[OperationContract]

List<Post> SelectPostsByUID(int ID);

[OperationContract]

Post SelectPostByID(int id);

[OperationContract]

void InsertDM(DMessage dMessage);

[OperationContract]

List<User> GetAllDMChats(User main);

[OperationContract]

List<DMessage> SelectAllDMsChat(User main, User chatted);

[OperationContract]

bool AreChatting(User main, User chatted);

[OperationContract]

List<DMessage> SelectUnSeenDM(User main, User chatted);

[OperationContract]

int GetChatsDMUpdateState(User main);

[OperationContract]

Group GetGroupByID(int groupID);

[OperationContract]

List<Group> GetGroupsByName(string name);

[OperationContract]

void InsertGroup(Group group, User creator);

[OperationContract]

void UpdateGroup(Group group);

[OperationContract]

void DeleteGroup(Group group);

[OperationContract]

GMember InsertGMemember(Group group, User user);

[OperationContract]

void DeleteGMember(GMember member);

[OperationContract]

void UpdateGMemberStatus(GMember member);

[OperationContract]

GMemberList GetGroupMembers(Group group);

[OperationContract]

List<User> SelectGroupAdmins(Group group);

[OperationContract]

List<User> SelectGroupNonAdmins(Group group);

[OperationContract]

bool IsGroupAdmin(GMember member);

[OperationContract]

bool IsGroupMember(GMember member);

[OperationContract]

GMessage InsertGMessage(GMessage message);

[OperationContract]

void DeleteGMessage(GMessage message);

[OperationContract]

void UpdateGMessage(GMessage message);

[OperationContract]



```
List<GMessage> SelectGMessages(User user, Group group);
```

```
[OperationContract]
```

```
List<GMessage> GetGMessages(User user, Group group);
```

```
[OperationContract]
```

```
List<GMessage> GetUnSeenGMessages(User user, Group group);
```

```
[OperationContract]
```

```
List<Group> GetMyGroups(User user);
```

```
[OperationContract]
```

```
int GetMyGroupsUpdateState(User user);
```

```
}
```

// Use a data contract as illustrated in the sample below to add composite types to service operations.

// You can add XSD files into the project. After building the project, you can directly use the data types defined there, with the namespace "WcfServiceWCF.ContractType".

```
[DataContract]
```

```
public class CompositeType
```

```
{
```

```
    bool boolValue = true;
```

```
    string stringValue = "Hello ";
```

```
[DataMember]
```

public bool BoolValue

```
{
    get { return boolValue; }
    set { boolValue = value; }
}
```

[DataMember]

public string StringValue

```
{
    get { return stringValue; }
    set { stringValue = value; }
}

}
```

Service1.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.Text;
using Model;
using ModelView;
```

```
{

public class Service1 : IService1

{

    //databases used:

    private CommentDB commentDB = new CommentDB();

    private UserDB userDB = new UserDB();

    private FollowDB followDB = new FollowDB();

    private FriendDB friendDB = new FriendDB();

    private LikeDB likeDB = new LikeDB();

    private PostDB postDB = new PostDB();

    private DMessageDB dMessageDB = new DMessageDB();

    private GroupDB groupDB = new GroupDB();

    private GMemberDB gMemberDB = new GMemberDB();

    private GMessageDB gMessageDB = new GMessageDB();

    private UnSeenGMDB unSeenGMDB = new UnSeenGMDB();

    //operations implementation

    public void DeleteComment(Comment entity)

    {

        commentDB.Delete(entity);

        commentDB.SaveChanges();

    }

    public void InsertComment(Comment entity)

    {

        commentDB.Insert(entity);

    }

}
```

```
commentDB.SaveChanges();
```

```
}
```

```
public void UpdateComment(Comment entity)
```

```
{
```

```
    commentDB.Update(entity);
```

```
    commentDB.SaveChanges();
```

```
}
```

```
public List<Comment> SelectCommentByPost(Post post)
```

```
{
```

```
    return commentDB.SelectByPost(post);
```

```
}
```

```
public User InsertUser(User entity)
```

```
{
```

```
    UserList users = userDB.SelectUNameExist(entity.Uname);
```

```
    if (users.Count == 0)
```

```
    {
```

```
        entity.ID = 0;
```

```
        return entity;
```

```
    }
```

```
    userDB.Insert(entity);
```

```
    userDB.SaveChanges();
```

```
    return entity;
```

```
}
```

```
public bool IsFollowing(Follow follow)
```

```
{
    return followDB.IsFollowing(follow);
}

public User Login(string userName, string password)
{
    UserList user = null;

    if (userName != "" && password != "")
    {
        user = userDB.SelectUserData(userName, password);
    }

    if(user.Count > 0)
    {
        return user[0];
    }

    return null;
}

public User SelectByUserID(int ID)
{
    return userDB.SelectByID(ID);
}

public List<User> SelectUserByUName(string Uname)
{
    return userDB.SelectByUName(Uname);
}

public void UpdateUser(User user)
{

```

```

        userDB.Update(user);

        userDB.SaveChanges();

    }

    public void DeleteUser(string userName, string password)
    {
        User user = new User();

        user.Username = userName;

        user.Password = password;

        userDB.Delete(user);

        userDB.SaveChanges();

    }

    public User SignUp(User user)
    {
        UserList userExist = userDB.SelectUserNameExist(user.Username);

        if (userExist.Count > 0)
        {
            return null;
        }

        userDB.Insert(user);

        userDB.SaveChanges();

        return user;

    }

    public string GetData(int value)
    {

```

```
return string.Format("You entered: {0}", value);
```

```
}
```

```
public CompositeType GetDataUsingDataContract(CompositeType composite)
```

```
{
```

```
    if (composite == null)
```

```
    {
```

```
        throw new ArgumentNullException("composite");
```

```
    }
```

```
    if (composite.BoolValue)
```

```
    {
```

```
        composite.StringValue += "Suffix";
```

```
    }
```

```
    return composite;
```

```
}
```

```
public Follow InsertFollow(Follow follow)
```

```
{
```

```
    followDB.Insert(follow);
```

```
    followDB.SaveChanges();
```

```
    return follow;
```

```
}
```

```
public void DeleteFollow(Follow follow)
```

```
{
```

```
    followDB.Delete(follow);
```

```
    followDB.SaveChanges();
```

}

public List<Follow> SelectFollowed(User follower)

{

return followDB.SelectFollowed(follower);

}

public Friend InsertFriend(Friend friend)

{

friendDB.Insert(friend);

friendDB.SaveChanges();

return friend;

}

public void DeleteFriend(Friend friend)

{

friendDB.Delete(friend);

friendDB.SaveChanges();

}

public bool AreFriends(Friend friend)

{

return friendDB.AreFriends(friend);

}

public Friend SelectFriend(User user1, User user2)

{

return friendDB.SelectFriend(user1, user2);

}



```
public List<Friend> SelectRecivedReq(User user)
```

```
{
```

```
    return friendDB.SelectRecivedReq(user);
```

```
}
```

```
public List<Friend> SelectUserFriends(User user)
```

```
{
```

```
    return friendDB.SelectUserFriends(user);
```

```
}
```

```
public void ApproveFriend(Friend friend)
```

```
{
```

```
    friend.Approved = true;
```

```
    friendDB.Update(friend);
```

```
    friendDB.SaveChanges();
```

```
}
```

```
public Like InsertLike(Like like)
```

```
{
```

```
    likeDB.Insert(like);
```

```
    likeDB.SaveChanges();
```

```
    return like;
```

```
}
```

```
public void DeleteLike(Like like)
```

```
{
```

```
    likeDB.Delete(like);
```

```
    likeDB.SaveChanges();
```

```
}
```

```
public int CountLikes(Post post)
{
    return likeDB.CountLikes(post);
}
```

```
public bool IsLiked(Like like)
{
    return likeDB.IsLiked(like);
}
```

```
public Post InsertPost(Post post)
{
    postDB.Insert(post);
    postDB.SaveChanges();
    return post;
}
```

```
public void DeletePost(Post post)
{
    postDB.Delete(post);
    postDB.SaveChanges();
}
```

```
public void UpdatePost(Post post)
{

```

```
postDB.Update(post);
```

```
postDB.SaveChanges();
```

```
}
```

```
public List<Post> SelectUserFeed(User mainUser, User req)
```

```
{
```

```
    return postDB.SelectUserFeed(mainUser, req);
```

```
}
```

```
public List<Post> SelectMyFeed(User user)
```

```
{
```

```
    return postDB.SelectMyFeed(user);
```

```
}
```

```
public List<Post> SelectPostsByUser(User user)
```

```
{
```

```
    return postDB.SelectByUser(user);
```

```
}
```

```
public List<Post> SelectPostsByUID(int ID)
```

```
{
```

```
    return postDB.SelectByUID(ID);
```

```
}
```

```
public Post SelectPostByID(int id)
```

```
{
```

}

public void InsertDM(DMessage dMessage)

{

dMessageDB.Insert(dMessage);

dMessageDB.SaveChanges();

}

public List<User> GetAllDMChats(User main)

{

return dMessageDB.GetAllDMChats(main);

}

public List<DMessage> SelectAllDMsChat(User main, User chatted)

{

return dMessageDB.SelectAllDMsChat(main, chatted);

}

public bool AreChatting(User main, User chatted)

{

return dMessageDB.AreChatting(main, chatted);

}

public List<DMessage> SelectUnSeenDM(User main, User chatted)

```
{
```

```
    return dMessageDB.SelectUnSeen(main, chatted);
```

```
}
```

```
public int GetChatsDMUpdateState(User main)
```

```
{
```

```
    return dMessageDB.GetChatsDMUpdateState(main);
```

```
}
```

```
public Group GetGroupByID(int groupID)
```

```
{
```

```
    return groupDB.SelectGroup(groupID);
```

```
}
```

```
public List<Group> GetGroupsByName(string name)
```

```
{
```

```
    return groupDB.SelectGroupByGName(name);
```

```
}
```

```
public void InsertGroup(Group group, User creator)
```

```
{
```

```
    groupDB.Insert(group);
```

```
    groupDB.SaveChanges();
```

```
    InsertGMember(group, creator);
```

```
public void UpdateGroup(Group group)
```

```
{
```

```
    groupDB.Update(group);
```

```
    groupDB.SaveChanges();
```

```
}
```

```
public void DeleteGroup(Group group)
```

```
{
```

```
    groupDB.Delete(group);
```

```
    groupDB.SaveChanges();
```

```
}
```

```
public GMember InsertGMember(Group group, User user)
```

```
{
```

```
    GMember gMember = new GMember(group, user, false);
```

```
    gMemberDB.Insert(gMember);
```

```
    gMemberDB.SaveChanges();
```

```
    return gMember;
```

```
}
```

```
public void DeleteGMember(GMember member)
```

```
{
```

```
    gMemberDB.Delete(member);
```

```
gMemberDB.SaveChanges();
```

```
}
```

```
public void UpdateGMemberStatus(GMember member)
```

```
{
```

```
    gMemberDB.Update(member);
```

```
    gMemberDB.SaveChanges();
```

```
}
```

```
public GMemberList GetGroupMembers(Group group)
```

```
{
```

```
    return gMemberDB.GetMembers(group);
```

```
}
```

```
public List<User> SelectGroupAdmins(Group group)
```

```
{
```

```
    return gMemberDB.SelectAdmins(group);
```

```
}
```

```
public List<User> SelectGroupNonAdmins(Group group)
```

```
{
```

```
    return gMemberDB.SelectNonAdmins(group);
```

```
}
```

```
public bool IsGroupAdmin(GMember member)
```

{

return gMemberDB.IsAdmin(member);

}

public bool IsGroupMember(GMember member)

{

return gMemberDB.IsMember(member);

}

public GMessage InsertGMessage(GMessage message)

{

gMessageDB.Insert(message);

gMessageDB.SaveChanges();

return message;

}

public void DeleteGMessage(GMessage message)

{

gMessageDB.Delete(message);

gMessageDB.SaveChanges();

}

public void UpdateGMessage(GMessage message)

{

gMessageDB.Update(message);



```
}
```

```
public List<GMessage> SelectGMessages(User user, Group group)
```

```
{
```

```
    return gMessageDB.GetMessages(user, group);
```

```
}
```

```
public List<GMessage> GetGMessages(User user, Group group)
```

```
{
```

```
    return gMessageDB.GetMessages(user, group);
```

```
}
```

```
public List<GMessage> GetUnSeenGMessages(User user, Group group)
```

```
{
```

```
    return unSeenGMDB.GetChatsUnSeenMessages(user, group);
```

```
}
```

```
public List<Group> GetMyGroups(User user)
```

```
{
```

```
    return gMemberDB.GetUsersGroup(user);
```

```
}
```

```
public int GetMyGroupsUpdateState(User user)
```

```
{
```

```
List<Group> groups = gMemberDB.GetUsersGroup(user);

return groups != null ? groups.Sum(group =>
group.GNname.GetHashCode()) * groups.Count : 0;

}

}

}
```

### צד לקוח:

App.xaml:

```
<Application x:Class="ChatStream.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:ChatStream"

StartupUri="Login.xaml">
```

```
<Application.Resources>

<ResourceDictionary>

<Style x:Key="SR_Buttons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>
```

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="Transparent" />

</Style>

<Style x:Key="AVR\_Button" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="Foreground" Value="LightYellow" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Height" Value="60" />

<Setter Property="Width" Value="120" />

<Setter Property="Margin" Value="10" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="#00B1F9" />

</Style>

<Style x:Key="MainBackground1" TargetType="Rectangle">

<Setter Property="Fill">

<Setter.Value>

<RadialGradientBrush GradientOrigin="1,1"

Center="0.5,0.5" RadiusX="0.9" RadiusY="1.2">

<GradientStop Color="#f2cc63" Offset="0.0" />

<GradientStop Color="#6393f2" Offset="1" />

</RadialGradientBrush>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="MainBackground2" TargetType="Rectangle">

<Setter Property="Fill">

<Setter.Value>

<LinearGradientBrush StartPoint="0,1" EndPoint="1,0">

<GradientStop Color="#FFEB84" Offset="0.0" />

<GradientStop Color="#FFAFAF" Offset="0.50" />

<GradientStop Color="#AFBFFF" Offset="1" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="textS" TargetType="TextBlock">

<Setter Property="FontSize" Value="30" />

<Setter Property="Foreground" Value="LightYellow" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

<Style x:Key="MainBackground" TargetType="Rectangle">

<Setter Property="Fill">

<Setter.Value>

<LinearGradientBrush StartPoint="0,1" EndPoint="1,0">

<GradientStop Color="#9791ff" Offset="0.2" />

<GradientStop Color="#656dfc" Offset="0.7" />

</LinearGradientBrush>

</Setter.Value>

</Setter>

</Style>

<Style x:Key="RButtons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Height" Value="46" />

```
<Setter Property="Margin" Value="10, 2" />
```

```
<Setter Property="BorderThickness" Value="2" />
```

```
</Style>
```

```
</ResourceDictionary>
```

```
</Application.Resources>
```

```
</Application>
```

App.Xaml.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Configuration;
```

```
using System.Data;
```

```
using System.Linq;
```

```
using System.Threading.Tasks;
```

```
using System.Windows;
```

```
namespace ChatStream
```

```
{
```

```
    /// <summary>
```

```
    /// Interaction logic for App.xaml
```

```
    /// </summary>
```

```
    public partial class App : Application
```

```
    {
```

```
    }
```

ChatsShow.xaml:

```
<UserControl x:Class="ChatStream.ChatsShow"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:local="clr-namespace:ChatStream"

    mc:Ignorable="d"

    d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

    <ResourceDictionary>

        <Style x:Key="textS" TargetType="TextBlock">

            <Setter Property="FontSize" Value="30" />

            <Setter Property="Foreground" Value="LightYellow" />

            <Setter Property="VerticalAlignment" Value="Center" />

            <Setter Property="HorizontalAlignment" Value="Center" />

        </Style>

        <Style x:Key="textSW" TargetType="TextBlock">

            <Setter Property="FontSize" Value="30" />

            <Setter Property="Foreground" Value="White" />

            <Setter Property="VerticalAlignment" Value="Center" />

            <Setter Property="HorizontalAlignment" Value="Center" />

        </Style>
```

<Style x:Key="SR\_Buttons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="Transparent" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightBlue">

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<StackPanel>



<WrapPanel HorizontalAlignment="Center">

<RadioButton x:Name="DirectRB" GroupName="messageType"  
IsChecked="True" Content="Direct Messages" FontSize="20" Margin="10"  
VerticalAlignment="Center" Checked="DirectRB\_Checked"/>

<RadioButton GroupName="messageType" Content="Group Messages"  
FontSize="20" Margin="10" Checked="RadioButton\_Checked" />

</WrapPanel>

<WrapPanel>

<TextBlock x:Name="DSerchTextShow" Text="Search user: " Margin="10"  
FontSize="30" TextAlignment="Center" Foreground="White"/>

<TextBlock x:Name="GSerchTextShow" Text="Search Group: "  
Margin="10" FontSize="30" TextAlignment="Center" Foreground="White"  
Visibility="Collapsed"/>

<TextBox Width="200" FontSize="25" TextAlignment="Center"  
x:Name="USerach\_TB" Margin="10" TextChanged="USerach\_TB\_TextChanged"/>

<Button Style="{StaticResource AVR\_Button}" Height="50" Width="180"  
Content="Clear Search" Click="Button\_Click"/>

<Button Style="{StaticResource AVR\_Button}" x:Name="CreateG"  
Height="50" Width="190" Content="Create Group" Click="CreateG\_Click"  
Visibility="Collapsed"/>

</WrapPanel>

<StackPanel x:Name ="Search\_Results">

</StackPanel>

<TextBlock Text="Chats" Style="{StaticResource textSW}" />

<StackPanel x:Name="MyChatsPanel" />

</StackPanel>

</Grid>

</UserControl>

ChatsShow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

using System.Threading;

namespace ChatStream

{

/// <summary>

/// Interaction logic for ChatsShow.xaml

/// </summary>

/// public class UserButton : Button

public partial class ChatsShow : UserControl

{

Service1Client srv = new Service1Client();

private MainWindow main;

private int updateState = 0;

private Thread refreshThread;

private bool typeDM = false;

public ChatsShow(MainWindow main)

{

this.main = main;

InitializeComponent();

RefreshAll();

refreshThread = new Thread(this.RefreshAll);

refreshThread.Start();

}

private void UFeed(object sender, RoutedEventArgs e)

{

EntityButton userButton = sender as EntityButton;

this.main.ChangeContent.Content = new UserFeed((User)userButton.Entity,  
this.main, this);

}

private void GShow(Object sender, RoutedEventArgs e)

```
{

    EntityButton memberBtn = sender as EntityButton;

    this.main.ChangeContent.Content = new
    GroupInfo(main,(GMember)memberBtn.Entity, this);

}

private void ChatUBtn(object sender, RoutedEventArgs e)////////////////////

{

    EntityButton userButton = sender as EntityButton;

    this.main.ChangeContent.Content = new ChatWithUser(this.main,
    (User)userButton.Entity, this);

}

private void ChatGBtn(object sender, RoutedEventArgs e)////////////////////

{

    EntityButton groupButton = sender as EntityButton;

    this.main.ChangeContent.Content = new ChatWithGroup(this.main,
    (Group)groupButton.Entity, this);

}

private void generateUserResults(StackPanel stack, List<User> userlist)

{

    Application.Current.Dispatcher.Invoke(() =>

    {

        stack.Children.Clear();

        foreach (User user in userlist)

        {

            WrapPanel wrapPanel = new WrapPanel();

            EntityButton uButton = new EntityButton();
```

```
EntityButton chatButton = new EntityButton();
```

```
uButton.Style = FindResource("SR_Buttons") as Style;
```

```
uButton.Content = user.Uname;
```

```
uButton.Entity = user;
```

```
uButton.Click += UFeed;
```

```
chatButton.Style = FindResource("SR_Buttons") as Style;
```

```
chatButton.Background = Brushes.Lavender;
```

```
chatButton.Content = "Chat";
```

```
chatButton.Entity = user;
```

```
chatButton.Click += ChatUBtn;
```

```
wrapPanel.Children.Add(uButton as Button);
```

```
wrapPanel.Children.Add(chatButton as Button);
```

```
stack.Children.Add(wrapPanel);
```

```
}
```

```
});
```

```
}
```

```
private void generateGroupResults(StackPanel stack, List<Group> grouplist)
```

```
{
```

```
Application.Current.Dispatcher.Invoke(() =>
```

```
{
```

```
stack.Children.Clear();
```

```
foreach (Group group in grouplist)
```

```
{
```

```
GMember gMember = new GMember();
```

```
gMember.User = this.main.User;
```

```
gMember.Group = group;
```

```
gMember.Admin = srv.IsGroupAdmin(gMember);
```

```
WrapPanel wrapPanel = new WrapPanel();
```

```
EntityButton gButton = new EntityButton();
```

```
EntityButton chatButton = new EntityButton();
```

```
gButton.Style = FindResource("SR_Buttons") as Style;
```

```
gButton.Content = group.GNname + "#" + group.ID;
```

```
gButton.Entity = gMember;
```

```
gButton.Click += GShow;
```

```
chatButton.Style = FindResource("SR_Buttons") as Style;
```

```
chatButton.Background = Brushes.Lavender;
```

```
chatButton.Content = "Chat";
```

```
chatButton.Entity = group;
```

```
chatButton.Click += ChatGBtn;
```

```
wrapPanel.Children.Add(gButton as Button);
```

```
wrapPanel.Children.Add(chatButton as Button);
```

```
stack.Children.Add(wrapPanel);
```

```
}
```

```
});
```

```
}
```

```
private void USerach_TB_TextChanged(object sender, TextChangedEventArgs  
e)
```

```
{
```

```
bool isDirect = (bool)DirectRB.IsChecked;
```

```
Search_Results.Children.Clear();
```

```
if (USerach_TB.Text.Length > 0 && isDirect)
```

```
{
```

```
List<User> userlist = srv.SelectUserByUName(USerach_TB.Text).ToList();
```

```
generateUserResults(Search_Results, userlist);
```

```
}
```

```
}
```

```
private void RadioButton_Checked(object sender, RoutedEventArgs e)
```

```
{
```

```
this.typeDM = false;
```

```
}
```

```
private void DirectRB_Checked(object sender, RoutedEventArgs e)
```

```
{
```

```
this.typeDM = true;
```

```
}
```

```
private void RefreshAll()
```

```
{
```

```
int currentUpdateState;
```

```
bool typeDMPrevious = this.typeDM;
```

```
while (this.main.GetCurrentControle() == this)
```

```
{
    if(this.typeDM)
    {
        currentUpdateState = srv.GetChatsDMUpdateState(this.main.User);
        if (this.updateState != currentUpdateState && typeDMPrevious ==
this.typeDM)
        {
            this.updateState = currentUpdateState;

            List<User> chatsUsers = srv.GetAllDMChats(main.User).ToList();
            generateUserResults(MyChatsPanel, chatsUsers);
        }
    }
    else
    {
        currentUpdateState = srv.GetMyGroupsUpdateState(this.main.User);
        if (this.updateState != currentUpdateState && typeDMPrevious ==
this.typeDM)
        {
            this.updateState = currentUpdateState;

            List<Group> chatsUsers = srv.GetMyGroups(main.User).ToList();
            generateGroupResults(MyChatsPanel, chatsUsers);
        }
    }

    typeDMPrevious = this.typeDM;
```



```
}
```

```
}
```

```
private void Button_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    USerach_TB.Text = "";
```

```
}
```

```
private void CreateG_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
}
```

```
}
```

```
}
```

ChatWithGroup.xaml:

```
<UserControl x:Class="ChatStream.ChatWithGroup"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
    xmlns:local="clr-namespace:ChatStream"
```

```
    mc:Ignorable="d"
```

<Grid>

<Grid.Background>

<ImageBrush ImageSource="/moon-day-sky.jpg" Stretch="UniformToFill"/>

</Grid.Background>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="4\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Button Content="Back" Width ="100" Style="{StaticResource AVR\_Button}"  
VerticalAlignment="Top" Click="Back\_Click"/>

<StackPanel Grid.Column="1">

<TextBlock Foreground="White" Background="#575757" Text="GroupName"  
TextAlignment="Center" Padding="5"/>

<ScrollView VerticalAlignment="Top" Height="400">

<StackPanel x:Name="ChatMessages">

</StackPanel>

</ScrollView>

</StackPanel>

```
<Grid Grid.Row="1" Grid.Column="1">
```

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="*" />
```

```
<ColumnDefinition Width="auto" />
```

```
</Grid.ColumnDefinitions>
```

```
<TextBox x:Name="textedMessage" FontSize="15" Text="" />
```

```
<Button Grid.Column="1" FontSize="20" Margin="0" Content="Send"
Height="auto" Style="{StaticResource AVR_Button}" Click="SendMessage_Click"/>
```

```
</Grid>
```

```
</Grid>
```

```
</UserControl>
```

ChatWithGroup.xaml.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
using System.Windows;
```

```
using System.Windows.Controls;
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Threading;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for ChatWithGroup.xaml

/// </summary>

public partial class ChatWithGroup : UserControl

{

private MainWindow main;

private Group group;

private UserControl sourceCont;

Service1Client srv = new Service1Client();

private bool isAutoUpdate;

private Thread autoUpdate;

public ChatWithGroup(MainWindow main, Group group, UserControl  
sourceCont)

```
this.main = main;

this.sourseCont = sourseCont;

this.group = group;

this.isAutoUpdate = true;

InitializeComponent();

this.autoUpdate = new Thread(this.AutoRefresh);

List<GMessage> messages = srv.GetGMessages(this.main.User,
this.group).ToList();

foreach (GMessage gMessage in messages)
{
    AppendMessage(gMessage);
}

this.autoUpdate.Start();
}

private void Back_Click(object sender, RoutedEventArgs e)
{
    this.main.ChangeContent.Content = this.sourseCont;
}

private void AppendMessage(GMessage message)
{
    Application.Current.Dispatcher.Invoke(() =>
```

```

var horizontalAl = HorizontalAlignment.Left;

var messageColour = new SolidColorBrush(Color.FromRgb(0x00, 0x5d,
0x6e));

if (message.User.ID != this.main.User.ID)
{
    horizontalAl = HorizontalAlignment.Right;

    messageColour = new SolidColorBrush(Color.FromRgb(0x4e, 0xad,
0x51));
}

WrapPanel wrapPanel = new WrapPanel()
{
    Width = double.NaN,

    HorizontalAlignment = horizontalAl
};

Border border = new Border()
{
    Margin = new Thickness(5),

    Width = double.NaN,

    Padding = new Thickness(5),

    BorderThickness = new Thickness(1),

    BorderBrush = Brushes.Black,

    Background = messageColour,

    CornerRadius = new CornerRadius(10)
};
    
```

```
StackPanel stackMessage = new StackPanel()
```

```
{
```

```
    Width = double.NaN,
```

```
    HorizontalAlignment = horizontalAl
```

```
};
```

```
TextBlock contentTextBlock = new TextBlock()
```

```
{
```

```
    Foreground = Brushes.White,
```

```
    Width = double.NaN,
```

```
    TextWrapping = TextWrapping.Wrap,
```

```
    Text = message.Content,
```

```
    Padding = new Thickness(5, 0, 5, 0)
```

```
};
```

```
TextBlock nameBlock = new TextBlock()
```

```
{
```

```
    Foreground = Brushes.White,
```

```
    Width = double.NaN,
```

```
    TextWrapping = TextWrapping.Wrap,
```

```
    Text = message.User.Username,
```

```
    Padding = new Thickness(5, 0, 5, 0)
```

```
};
```

```
nameBlock.FontSize = 10;
```

```
contentTextBlock.FontSize = 20;
```

```
stackMessage.Children.Add(nameBlock);
```

```
stackMessage.Children.Add(contentTextBlock);
```

```

border.Child = stackMessage;

wrapPanel.Children.Add(border);

ChatMessages.Children.Add(wrapPanel);

});

}

private void AutoRefresh()
{
    while (this.main.GetCurrentControle() == this)
    {
        Thread.Sleep(2000);

        List<GMessage> messages = srv.GetUnSeenGMessages(this.main.User,
this.group).ToList();

        foreach (GMessage gMessage in messages)
        {
            AppendMessage(gMessage);
        }
    }
}

private void SendMessage_Click(object sender, RoutedEventArgs e)
{
    GMessage message = new GMessage();

```



```
message.Content = textedMessage.Text;
```

```
message.User = this.main.User;
```

```
message.Group = this.group;
```

```
//message.Date = DateTime.Now;
```

```
AppendMessage(message);
```

```
srv.InsertGMessage(message);
```

```
textedMessage.Text = "";
```

```
}
```

```
}
```

```
}
```

ChatWithUser.xaml:

```
<UserControl x:Class="ChatStream.ChatWithUser"
```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
xmlns:local="clr-namespace:ChatStream"
```

```
mc:Ignorable="d"
```

```
d:DesignHeight="450" d:DesignWidth="800">
```

```
<Grid>
```

```
<Grid.Background>
```

```
<ImageBrush ImageSource="/moon-day-sky.jpg" Stretch="UniformToFill"/>
```

```
</Grid.Background>
```

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="4\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Button Content="Back" Width ="100" Style="{StaticResource AVR\_Button}"  
VerticalAlignment="Top" Click="Back\_Click"/>

<StackPanel Grid.Column="1">

<TextBlock x:Name="UserNameChat" Foreground="White"  
Background="#575757" Text="Username" TextAlignment="Center" Padding="5"/>

<ScrollView VerticalAlignment="Top" >

<StackPanel x:Name="ChatMessages" Grid.Column="1">

<WrapPanel Width="auto"/>

<WrapPanel Width="auto" HorizontalAlignment="Right"/>

</StackPanel>

</ScrollView>

</StackPanel>

<Grid Grid.Row="1" Grid.Column="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="auto" />

</Grid.ColumnDefinitions>

<TextBox x:Name="textedMessage" FontFamily="20" Text="message" />

<Button Grid.Column="1" FontSize="20" Margin="0" Content="Send"  
Height="auto" Style="{StaticResource AVR\_Button}" Click="SendMessage\_Click"/>

</Grid>

</Grid>

</UserControl>

ChatWithUser.Xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Threading;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for ChatWithUser.xaml

/// </summary>

public partial class ChatWithUser : UserControl

{

private MainWindow main;

private User user;

private UserControl sourceCont;

Service1Client srv = new Service1Client();

private bool isAutoUpdate;

private Thread autoUpdate;

public ChatWithUser(MainWindow main, User user, UserControl sourceCont)

{

this.main = main;

```
this.sourseCont = sourceCont;

this.user = user;

this.isAutoUpdate = true;

UserNameChat.Text = user.Uname + "#id:" + user.ID;

InitializeComponent();

this.autoUpdate = new Thread(this.AutoRefresh);//this thread will update
messages every 2 seconds

List<DMessage> messages = srv.SelectAllDMsChat(this.main.User,
this.user).ToList();

foreach (DMessage dMessage in messages)
{
    AppendMessage(dMessage);
}

this.autoUpdate.Start();//start update messages every 2 seconds
}

//appends certain message to the visible chat

private void AppendMessage(DMessage message)
{
    Application.Current.Dispatcher.Invoke(() => ///this made to allow using this
function from a thread

    {
        var horizontalAl = HorizontalAlignment.Left;

        var messageColour = new SolidColorBrush(Color.FromRgb(0x00, 0x5d,
0x6e));

        if (message.DestUser.ID == this.user.ID)
        {
            horizontalAl = HorizontalAlignment.Right;
```

```
messageColour = new SolidColorBrush(Color.FromRgb(0x4e, 0xad,
0x51));
```

```
}
```

```
WrapPanel wrapPanel = new WrapPanel()
```

```
{
```

```
Width = double.NaN,
```

```
HorizontalAlignment = horizontalAl
```

```
};
```

```
Border border = new Border();// message's circular border
```

```
{
```

```
Margin = new Thickness(5),
```

```
Width = double.NaN,
```

```
Padding = new Thickness(5),
```

```
BorderThickness = new Thickness(1),
```

```
BorderBrush = Brushes.Black,
```

```
Background = messageColour,
```

```
CornerRadius = new CornerRadius(10)
```

```
};
```

```
WrapPanel wrapMessage = new WrapPanel()
```

```
{
```

```
Width = double.NaN,
```

```
HorizontalAlignment = horizontalAl
```

```
};
```

```
TextBlock contentTextBlock = new TextBlock();//message content
```

{

Foreground = Brushes.White,

Width = double.NaN,

TextWrapping = TextWrapping.Wrap,

Text = message.Content,

Padding = new Thickness(5, 0, 5, 0)

};

TextBlock dateBlock = new TextBlock() //message time sent

{

Foreground = Brushes.White,

Width = double.NaN,

TextWrapping = TextWrapping.Wrap,

Text = message.Time.ToString(),

Padding = new Thickness(5, 0, 5, 0)

};

dateBlock.FontSize = 10;

contentTextBlock.FontSize = 20;

wrapMessage.Children.Add(contentTextBlock);

wrapMessage.Children.Add(dateBlock);

border.Child = wrapMessage;

wrapPanel.Children.Add(border);

ChatMessages.Children.Add(wrapPanel);

});

```

    }

    private void AutoRefresh() //refresh all messages every 2 seconds.

    {
        while(this.main.GetCurrentControle() == this)
        {
            Thread.Sleep(2000);

            List<DMessage> messages = srv.SelectUnSeenDM(this.main.User,
this.user).ToList();

            foreach(DMessage dMessage in messages)
            {
                AppendMessage(dMessage);
            }
        }
    }

    //sends a message to the second user

    private void SendMessage_Click(object sender, RoutedEventArgs e)
    {
        DMessage message = new DMessage();
        message.Content = textedMessage.Text;
        message.SourceUser = this.main.User;
        message.DestUser = this.user;
        message.Time = DateTime.Now;
        AppendMessage(message);
        srv.InsertDM(message);
    }

```



```

    }

    //returns to the previos UserControl shown by mainwindow

    private void Back_Click(object sender, RoutedEventArgs e)

    {

        this.main.ChangeContent.Content = this.sourseCont;

    }

}

}

```

CommentView.xaml:

```

<UserControl x:Class="ChatStream.CommentView"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:local="clr-namespace:ChatStream"

    mc:Ignorable="d"

    d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

    <ResourceDictionary>

        <Style x:Key="Post_Buttons" TargetType="Button">

            <Style.Resources>

```

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Width" Value="auto" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightCyan">

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<WrapPanel VerticalAlignment="Center">

```
<Button Style="{StaticResource Post_Buttons}" x:Name="CUName"
Content="User Name" Click="UFeed" Background="Transparent"/>
```

```
<Button Style="{StaticResource Post_Buttons}" x:Name="Edit_B"
Content="Edit" Background="Transparent" Click="Edit_B_Click"
Visibility="Collapsed"/>
```

```
<Button Style="{StaticResource Post_Buttons}" x:Name="Save_B"
Content="Save" Background="Transparent" Click="Save_B_Click"
Visibility="Collapsed"/>
```

```
</WrapPanel>
```

```
<TextBox TextWrapping="Wrap" Grid.Row="1" x:Name="Content" Margin="10"
Background="GhostWhite" IsReadOnly="True" FontSize="25"/>
```

```
</Grid>
```

```
</UserControl>
```

CommentView.xaml.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;
```

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for CommentView.xaml

/// </summary>

public partial class CommentView : UserControl

{

Service1Client srv = new Service1Client();

private Comment comment;

private UserControl currentC;

private MainWindow main;

public CommentView(Comment comment, MainWindow main , UserControl  
currentC)

{

this.comment = comment;

this.main = main;

this.currentC = currentC;

InitializeComponent();

if(main.User.ID == comment.User.ID)

{

Edit\_B.Visibility = Visibility.Visible;

}

CUName.Content = comment.User.Uname;

Content.Text = comment.Content;

}

private void Edit\_B\_Click(object sender, RoutedEventArgs e)

{

Edit\_B.Visibility = Visibility.Collapsed;

Save\_B.Visibility = Visibility.Visible;

Content.IsReadOnly = false;

}

private void Save\_B\_Click(object sender, RoutedEventArgs e)

{

Edit\_B.Visibility = Visibility.Visible;

Save\_B.Visibility = Visibility.Collapsed;

Content.IsReadOnly = true;

this.comment.Content = Content.Text;

srv.UpdateComment(this.comment);

}

private void UFeed(object sender, RoutedEventArgs e)

{

```
this.main.ChangeContent.Content = new UserFeed(this.comment.User,
this.main, this.currentC);
```

```
    }
}
}
```

CreateGroup.xaml:

```
<UserControl x:Class="ChatStream.CreateGroup"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
    xmlns:local="clr-namespace:ChatStream"
```

```
    mc:Ignorable="d"
```

```
    d:DesignHeight="450" d:DesignWidth="800">
```

```
<UserControl.Resources>
```

```
<ResourceDictionary>
```

```
<Style x:Key="TextBoxPForm" TargetType="TextBox">
```

```
    <Setter Property="FontSize" Value="25" />
```

```
    <Setter Property="Margin" Value="5" />
```

```
    <Setter Property="Opacity" Value="0.55" />
```

```
    <Setter Property="Height" Value="45" />
```

```
</Style>
```

```
</ResourceDictionary>
```

</UserControl.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="2\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBox Grid.Column="1" Grid.Row="0" x:Name="GName"  
Style="{StaticResource TextBoxPForm}" />

<TextBox Grid.Column="1" Grid.Row="2" x:Name="Description"  
Style="{StaticResource TextBoxPForm}" Height="auto"/>

<TextBlock Grid.Column="0" Grid.Row="0" Margin="1" Text="Group Name"  
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>

<TextBlock Grid.Column="0" Grid.Row="1" Margin="1" Text="Publicity"  
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>

<TextBlock Grid.Column="0" Grid.Row="2" Margin="1" Text="Description"  
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>

<Button Content="Back" Style="{StaticResource AVR\_Button}"  
Grid.Column="0" Grid.Row="3" Click="Back\_Click" />

<Button Content="Save" Style="{StaticResource AVR\_Button}"  
Grid.Column="1" Grid.Row="3" Click="Save\_Click" />

<WrapPanel Grid.Row="1" Grid.Column="1">

```
<RadioButton Content="Public" Foreground="White" FontSize="30"
Margin="20" x:Name="Public"/>
```

```
<RadioButton Content="Private" Foreground="White" FontSize="30"
Margin="20" IsChecked="True"/>
```

```
</WrapPanel>
```

```
</Grid>
```

```
</UserControl>
```

CreateGroup.xaml.cs:

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;
```

```
namespace ChatStream
```



/// <summary>

/// Interaction logic for CreateGroup.xaml

/// </summary>

public partial class CreateGroup : UserControl

{

MainWindow main;

UserControl previous;

Service1Client srv = new Service1Client();

public CreateGroup(MainWindow main, UserControl previous)

{

InitializeComponent();

this.main = main;

this.previous = previous;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

Group group = new Group();

if(GName.Text.Length > 3)

{

group.GName = GName.Text;

group.Publicity = Public.IsChecked.GetValueOrDefault();

group.Description = Description.Text + " ";

srv.InsertGroup(group, this.main.User);

```

    }

}

private void Back_Click(object sender, RoutedEventArgs e)
{
    this.main.ChangeContent.Content = this.previous;
}
}
}
}

```

CreatePost.xaml:

```

<UserControl x:Class="ChatStream.CreatePost"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:ChatStream"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

    <ResourceDictionary>

        <Style x:Key="Post_Buttons" TargetType="Button">

            <Style.Resources>

                <Style TargetType="Border">

```

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightBlue">

<Grid.RowDefinitions>

<RowDefinition Height="5\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0" VerticalAlignment="Top" HorizontalAlignment="Left" />

<TextBox TextWrapping="Wrap" Grid.Row="0" x:Name="Content" Margin="5" Background="White" />

<Grid Grid.Row="2">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<ComboBox Height="30" Width="100" VerticalContentAlignment="Top"  
x:Name="publicity">

<ComboBoxItem Tag="Public" Content="Public" />

<ComboBoxItem Tag="Private" Content="Private" />

</ComboBox>

<WrapPanel Grid.Column="1" HorizontalAlignment="Center">

<Button Click="CreateContent" Grid.Column="1" x:Name="Create1"  
Content=" Create " FontSize="25" Margin="10"/>

<Button Click="Cancel" Grid.Column="1" x:Name="CancelB" Content=" Cancel "  
FontSize="25" Margin="10"/>

</WrapPanel>

<ScrollViewer Grid.ColumnSpan="2" x:Name="CViewer" Grid.Row="1"  
Visibility="Collapsed">

<ListView x:Name="Comments" >

<TextBlock></TextBlock>

</ListView>

</Grid>

</Grid>

</UserControl>

CreatePost.xamk.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for CreatePost.xaml

/// </summary>

```
public partial class CreatePost : UserControl
{
    private User pUser;

    private Button sButton;

    Service1Client srv = new Service1Client();

    public CreatePost(User user, Button button)
    {
        InitializeComponent();

        publicity.SelectedIndex = 0;

        sButton = button;

        pUser = user;
    }
}
```

```
private void CreateContent(object sender, RoutedEventArgs e)
{
    Post post = new Post();

    post.Content = Content.Text;

    post.IsPrivate = publicity.SelectedIndex == 1;

    post.User = pUser;

    srv.InsertPost(post);

    Content.Text = "";
}
```

```
sButton.Visibility = Visibility.Visible;
```

```
this.Visibility = Visibility.Collapsed;
```

```
}
```

```
private void Cancel(object sender, RoutedEventArgs e)
```

```
{
```

```
sButton.Visibility = Visibility.Visible;
```

```
this.Visibility = Visibility.Collapsed;
```

```
}
```

```
}
```

```
}
```

Feed.xaml:

```
<UserControl x:Class="ChatStream.Feed"
```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
xmlns:local="clr-namespace:ChatStream"
```

```
mc:Ignorable="d"
```

```
d:DesignHeight="450" d:DesignWidth="800">
```

```
<UserControl.Resources>
```

<ResourceDictionary>

<Style x:Key="textS" TargetType="TextBlock">

<Setter Property="FontSize" Value="30" />

<Setter Property="Foreground" Value="LightYellow" />

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="HorizontalAlignment" Value="Center" />

</Style>

<Style x:Key="SR\_Buttons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="Transparent" />

</Style>

</ResourceDictionary>

</UserControl.Resources>



<Grid>

<ScrollView Grid.Row="1" VerticalAlignment="Top">

<StackPanel>

<TextBlock Text="My Feed" Style="{StaticResource textS}"/>

<WrapPanel>

<TextBlock Text="Search user: " Margin="10" FontSize="30"  
TextAlignment="Center" Foreground="White"/>

<TextBox Width="200" FontSize="25" TextAlignment="Center"  
x:Name="USerach\_TB" Margin="10" TextChanged="USerach\_TB\_TextChanged"/>

<Button Style="{StaticResource AVR\_Button}" Height="50" Width="180"  
Content="Clear Search" Click="ClearSearch\_Click" />

</WrapPanel>

<StackPanel Background="LightCyan" x:Name="Search\_Results">

</StackPanel>

<WrapPanel HorizontalAlignment="Center">

<Button Style="{StaticResource AVR\_Button}" Height="50"  
Content="Refresh" Click="Refresh\_Click" />

<Button x:Name="AddPost" Style="{StaticResource AVR\_Button}"  
Width="150" Height="50" Content="Add Post" Click="AddPost\_Click" />

</WrapPanel>

<ContentControl Grid.Row="4" x:Name="CreatePostV"  
Visibility="Visible"/>

<StackPanel x:Name="FeedView">

</StackPanel>

</StackPanel>

</Grid>

</UserControl>

Feed.xaml.cs:

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for Feed.xaml

/// </summary>

public partial class Feed : UserControl

{

private MainWindow main;

Service1Client srv = new Service1Client();

public Feed(MainWindow main)

{

this.main = main;

InitializeComponent();

//create a UserControl of CreatePost, where user can createpost

CreatePostV.Content = new CreatePost(main.User, AddPost);

((CreatePost)CreatePostV.Content).Visibility = Visibility.Collapsed;

//receive all the posts that user should see in the feed

List<Post> posts = srv.SelectMyFeed(this.main.User).ToList();

foreach (Post post in posts) //show each post

{

UserControl userControl = new UserControl();

userControl.Content = new PostView(post, main, this);

userControl.Margin = new Thickness(10);

FeedView.Children.Add(userControl);

}

}

```

/// <summary>

/// the method runs when an event of changing the text box content occurs

/// the method shows all the users that start with what was typed in text box

/// </summary>

/// <param name="sender">textbox</param>

/// <param name="e"></param>

private void USerach_TB_TextChanged(object sender, TextChangedEventArgs
e)

{
    TextBox textBox = sender as TextBox;

    Search_Results.Children.Clear();

    if (textBox.Text.Length > 0) //if user searched uname
    {
        //select all the usernames starts with what was typed in the search bar
        List<User> userlist = srv.SelectUserByUName(textBox.Text).ToList();

        foreach (User user in userlist)
        {
            WrapPanel wrapPanel = new WrapPanel();

            EntityButton uButton = new EntityButton();

            TextBlock name = new TextBlock();

            //button with username that shows user's personal feed and info when
            clicked

            uButton.Style = FindResource("SR_Buttons") as Style;

            uButton.Content = user.Uname;

            uButton.Entity = user;

            uButton.Click += UFeed;
        }
    }
}

```

//Text Block that presents the full name of the user

```
name.Style = FindResource("textS") as Style;
```

```
name.Foreground = Brushes.Black;
```

```
name.Text = user.Fname + " " + user.Lname;
```

```
name.FontSize = 20;
```

```
wrapPanel.Children.Add(uButton as Button);
```

```
wrapPanel.Children.Add(name);
```

Search\_Results.Children.Add(wrapPanel);// add the search results to the stackpanel

```
}
```

```
}
```

```
}
```

```
/// <summary>
```

```
/// opens user's feed by clicking the button named by username.
```

```
/// </summary>
```

```
/// <param name="sender">button</param>
```

```
/// <param name="e"></param>
```

```
private void UFeed(object sender, RoutedEventArgs e)
```

```
{
```

```
    EntityButton userButton = sender as EntityButton;
```

```
    this.main.ChangeContent.Content = new UserFeed((User)userButton.Entity, this.main, this);
```

```
}
```

```
/// <summary>
```

```
/// this button shows the add post UserControl in order to allow the user
```

/// to add a post.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void AddPost\_Click(object sender, RoutedEventArgs e)

{

Button button = sender as Button;

button.Visibility = Visibility.Collapsed;

((CreatePost)CreatePostV.Content).Visibility = Visibility.Visible;

}

/// <summary>

/// the clear Search button clears the content searched in the textbox

/// </summary>

/// <param name="sender">button</param>

/// <param name="e"></param>

private void ClearSearch\_Click(object sender, RoutedEventArgs e)

{

USerach\_TB.Text = "";

}

/// <summary>

/// refreshes all the posts in the feed

/// </summary>

/// <param name="sender">refresh button </param>

/// <param name="e"></param>

public void Refresh\_Click(object sender, RoutedEventArgs e)

```
{

    FeedView.Children.Clear();

    List<Post> posts = srv.SelectMyFeed(this.main.User).ToList();

    foreach (Post post in posts)
    {
        UserControl userControl = new UserControl();

        userControl.Content = new PostView(post, main, this);

        userControl.Margin = new Thickness(10);

        FeedView.Children.Add(userControl);
    }
}

/*public class UserButton : Button
{
    private User user;

    public UserButton() : base() { this.user = null; }

    public UserButton(User user) : base() { this.user = user; }

    public User User { get => user; set => user = value; }
}*/

public class EntityButton : Button
{
    private BaseEntity entity;

    public EntityButton() : base() { this.entity = null; }

    public EntityButton(BaseEntity entity) : base() { this.entity = entity; }
```

```
public BaseEntity Entity { get => entity; set => entity = value; }

}

}
```

FriendListShow.xaml:

```
<UserControl x:Class="ChatStream.FriendListShow"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:local="clr-namespace:ChatStream"

    mc:Ignorable="d"

    d:DesignHeight="450" d:DesignWidth="800">

    <UserControl.Resources>

        <ResourceDictionary>

            <Style x:Key="SR_Buttons" TargetType="Button">

                <Style.Resources>

                    <Style TargetType="Border">

                        <Setter Property="CornerRadius" Value="5"/>

                    </Style>

                </Style.Resources>

                <Setter Property="BorderBrush" Value="Transparent" />

            </Style>

        </ResourceDictionary>

    </UserControl.Resources>
```



<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="Transparent" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightBlue">

<ScrollView Grid.Row="1" VerticalAlignment="Top">

<StackPanel>

<TextBlock Text="Search people" Style="{StaticResource textS}"/>

<WrapPanel>

<TextBlock Text="Search user: " Margin="10" FontSize="30"  
TextAlignment="Center" Foreground="White"/>

<TextBox Width="200" FontSize="25" TextAlignment="Center"  
x:Name="USerach\_TB" Margin="10" TextChanged="USerach\_TB\_TextChanged"/>

<Button Style="{StaticResource AVR\_Button}" Height="50" Width="180"  
Content="Clear Search" Click="Clear\_Click" />

</WrapPanel>

<StackPanel x:Name="FriendsSearchView" />

<TextBlock Text="Pending" Style="{StaticResource textS}"/>

<StackPanel x:Name="PendReqView" />

<TextBlock Text="My Friends" Style="{StaticResource textS}"/>

<StackPanel x:Name="MyFriendsView" />

</StackPanel>

</ScrollView>

</Grid>

</UserControl>

FriendListShow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

/// <summary>

/// Interaction logic for FriendListShow.xaml

/// </summary>

public partial class FriendListShow : UserControl

{

Service1Client srv = new Service1Client();

private MainWindow main;

public FriendListShow(MainWindow main)

{

this.main = main;

InitializeComponent();

List<Friend> friendList = srv.SelectUserFriends(this.main.User).ToList();

RefreshListResults(MyFriendsView, friendList);

List<Friend> reqList = srv.SelectRecivedReq(this.main.User).ToList();

RefreshListResults(PendReqView, reqList);

}

e) private void USerach\_TB\_TextChanged(object sender, TextChangedEventArgs

{

TextBox textBox = sender as TextBox;

FriendsSearchView.Children.Clear();

if (textBox.Text.Length > 0)

{

List<User> userList = srv.SelectUserByUName(textBox.Text).ToList();

foreach (User user in userList)

{

FriendShow friendShow = new FriendShow(this.main, user, this);

FriendsSearchView.Children.Add(friendShow);

}

}

}

private void RefreshListResults(StackPanel panel, List<Friend> list)

{

panel.Children.Clear();

foreach (Friend friend in list)

{

if (friend.User1.ID != this.main.User.ID)

{

FriendShow friendShow = new FriendShow(this.main, friend.User1,  
this);

panel.Children.Add(friendShow);

}

if (friend.User2.ID != this.main.User.ID)

{

```
FriendShow friendShow = new FriendShow(this.main, friend.User2,
this);

panel.Children.Add(friendShow);

}

}

}

private void Clear_Click(object sender, RoutedEventArgs e)
{
    USerach_TB.Text = "";
}

}

}
```

FriendShow.xaml:

```
<UserControl x:Class="ChatStream.FriendShow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:ChatStream"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800">
<UserControl.Resources>
<ResourceDictionary>
<Style x:Key="FormTextBox" TargetType="TextBlock">
```

<Setter Property="Margin" Value="1" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Foreground" Value="White" />

<Setter Property="TextAlignment" Value="Center" />

</Style>

<Style x:Key="SR\_Buttons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="Transparent" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightBlue">

<WrapPanel>

<Button FontSize="30" x:Name="UName\_B" Content="UserName"  
Style="{StaticResource SR\_Buttons}" Click="UFeed"/>

<TextBlock FontSize="20" x:Name="FullName" Text="fullname"  
VerticalAlignment="Center"/>

<Button x:Name="FriendB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Friend Request" Click="FriendB\_Click"  
Visibility="Collapsed"/>

<Button x:Name="UnFriendB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Remove Friend" Click="UnFriendB\_Click"  
Visibility="Collapsed"/>

<Button x:Name="AcceptF\_B" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Accept Friend" Click="AcceptF\_B\_Click"  
Visibility="Collapsed"/>

<Button x:Name="DeclineF\_B" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Decline Friendship" Click="UnFriendB\_Click"  
Visibility="Collapsed"/>

</WrapPanel>

</Grid>

</UserControl>

FriendShow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for FriendShow.xaml

/// </summary>

public partial class FriendShow : UserControl

{

Service1Client srv = new Service1Client();

private User mainU;

private MainWindow main;

private User user;

private FriendListShow friendListShow;

public FriendShow(MainWindow main, User user, FriendListShow  
friendListShow)

{



```
this.mainU = main.User;

this.user = user;

this.main = main;

this.friendListShow = friendListShow;

Friend friend = srv.SelectFriend(this.mainU, this.user);

InitializeComponent();

UName_B.Content = user.Uname;

FullName.Text = user.Fname + " " + user.Lname;

if (friend == null)
{
    FriendB.Visibility = Visibility.Visible;
}
else
{
    if (friend.Approved)
    {
        UnFriendB.Visibility = Visibility.Visible;
    }
    else
    {
        DeclineF_B.Visibility = Visibility.Visible;

        if (friend.User2.ID == this.mainU.ID)

            AcceptF_B.Visibility = Visibility.Visible;
    }
}
```

}

```
private void FriendB_Click(object sender, RoutedEventArgs e)
{
```

```
    FriendB.Visibility = Visibility.Collapsed;
```

```
    DeclineF_B.Visibility = Visibility.Visible;
```

```
    Friend friend = new Friend();
```

```
    friend.User1 = this.mainU;
```

```
    friend.User2 = this.user;
```

```
    srv.InsertFriend(friend);
```

```
}
```

```
private void UnFriendB_Click(object sender, RoutedEventArgs e)
{
```

```
    UnFriendB.Visibility = Visibility.Collapsed;
```

```
    DeclineF_B.Visibility = Visibility.Collapsed;
```

```
    AcceptF_B.Visibility = Visibility.Collapsed;
```

```
    FriendB.Visibility = Visibility.Visible;
```

```
    Friend friend = new Friend();
```

```
    friend.User1 = this.mainU;
```

```
    friend.User2 = this.user;
```

}

private void AcceptF\_B\_Click(object sender, RoutedEventArgs e)

{

UnFriendB.Visibility = Visibility.Visible;

DeclineF\_B.Visibility = Visibility.Collapsed;

AcceptF\_B.Visibility = Visibility.Collapsed;

FriendB.Visibility = Visibility.Collapsed;

Friend friend = new Friend();

friend.User1 = this.mainU;

friend.User2 = this.user;

srv.ApproveFriend(friend);

}

private void UFeed(object sender, RoutedEventArgs e)

{

this.main.ChangeContent.Content = new UserFeed(this.user, this.main, this.friendListShow);

}

}

}

GroupInfo.xaml:

<UserControl x:Class="ChatStream.GroupInfo"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:ChatStream"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

<ResourceDictionary>

<Style x:Key="RW\_Buttons" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="White" />

</Style>

</ResourceDictionary>

<Grid Background="LightBlue">

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid Grid.Row="0" >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="auto" />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Button HorizontalAlignment="Left" Content="Back" Click="Back\_Click"  
Style="{StaticResource AVR\_Button}" Height="50" Width="80"/>

<WrapPanel Grid.Column="1">

<TextBox Grid.Column="1" x:Name="GNameTxt" BorderThickness="3"  
IsReadOnly="True" BorderBrush="Transparent" Foreground="White" Text="Group  
Name" FontSize="40" Background="Transparent"/>

<TextBlock Grid.Column="2" x:Name="PublicityDisplayText"  
Text="Publicity" Style="{StaticResource textS}" FontSize="20"  
VerticalAlignment="Bottom" />

<Button Grid.Column="3" x:Name="EditGroupName" Content="Edit"  
Style="{StaticResource RButtons}" Opacity="0.5" Background="#c08bfc"  
Padding="5" Click="EditGroupName\_Click" Visibility="Collapsed"/>

```
<Button Grid.Column="3" x:Name="SaveGroupName" Content="Save"
Style="{StaticResource RButtons}" Opacity="0.5" Background="#c08bfc"
Padding="5" Click="SaveGroupName_Click" Visibility="Collapsed"/>
```

```
</WrapPanel>
```

```
</Grid>
```

```
<ScrollView Grid.Row="1">
```

```
<StackPanel>
```

```
<WrapPanel>
```

```
<TextBlock Style="{StaticResource textS}" Text="Members:"
HorizontalAlignment="Left" Margin="20, 0"/>
```

```
<Button Content="Refresh" Click="RefreshMembers_Click"
Style="{StaticResource AVR_Button}" Height="50" />
```

```
<Button Content="Chat" Click="OpenChat_Click"
Style="{StaticResource AVR_Button}" Height="50" />
```

```
</WrapPanel>
```

```
<ScrollView Height="300">
```

```
<StackPanel x:Name="MemberListStack">
```

```
</StackPanel>
```

```
</ScrollView>
```

```
<WrapPanel x:Name="AdminsOptionPanel" Visibility="Collapsed">
```

```
<Button x:Name="AddMemberBtn" Content="Add Member" Width="200"
Style="{StaticResource RButtons}" Background="LightYellow"
Click="AddMemberBtn_Click"/>
```

```
<TextBlock Style="{StaticResource textS}" Text="Publicity:"
HorizontalAlignment="Left"/>
```

```
<Button x:Name="MakePublic_btn" Content="Make Public" Width="200"
Style="{StaticResource RButtons}" Background="LightYellow"
Click="MakePublic_btn_Click"/>
```

```
<Button x:Name="MakePrivate_btn" Content="Make Private"
Width="200" Style="{StaticResource RButtons}" Background="LightYellow"
Click="MakePrivate_btn_Click"/>
```

```
</WrapPanel>
```

```
<StackPanel x:Name="MemberAddStack" Visibility="Collapsed">
```

```
<WrapPanel>
```

```
<TextBlock x:Name="DSerchTextShow" Text="Search user: "
Margin="10" FontSize="30" TextAlignment="Center" Foreground="White"/>
```

```
<TextBlock x:Name="GSerchTextShow" Text="Search Group: "
Margin="10" FontSize="30" TextAlignment="Center" Foreground="White"
Visibility="Collapsed"/>
```

```
<TextBox Width="200" FontSize="25" TextAlignment="Center"
x:Name="USerach_TB" Margin="10" TextChanged="USerach_TB_TextChanged"/>
```

```
<Button Style="{StaticResource AVR_Button}" Height="50"
Width="180" Content="Clear Search" Click="Clear_Click" />
```

```
<Button Style="{StaticResource AVR_Button}" Height="50"
Width="100" Content="Close" Click="CloseAdd_Click" />
```

```
</WrapPanel>
```

```
<StackPanel x:Name="MemberAddStack_Results">
```

```
</StackPanel>
```

```
</StackPanel>
```

```
</StackPanel>
```

```
</ScrollViewer>
```

</UserControl>

GroupInfo.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

using System.Threading;

namespace ChatStream

{

/// <summary>

/// Interaction logic for GroupInfo.xaml



```
public partial class GroupInfo : UserControl
```

```
{
```

```
//properties
```

```
Service1Client srv = new Service1Client();
```

```
private MainWindow main;
```

```
private Group group;
```

```
private UserControl previous;
```

```
private bool isAdmin;
```

```
public GroupInfo(MainWindow main, GMember member, UserControl previous)
```

```
{
```

```
    this.main = main;
```

```
    this.group = srv.GetGroupByID(member.Group.ID); //get group data
```

```
    this.previous = previous;
```

```
    member.User = main.User;
```

```
    InitializeComponent();
```

```
    GNameTxt.Text = this.group.GNname + "#" + group.ID; //show group name  
with ID
```

```
    this.isAdmin = srv.IsGroupAdmin(member);
```

```
    RefreshGMembers();//show all group members
```

```
    //dispay if group is private or public
```

```
    PublicityDisplayText.Text = this.group.Publicity ? "Public" : "Private";
```

```
    if(this.isAdmin)
```

```
{
```

```
        AdminsOptionPanel.Visibility = Visibility.Visible;
```

```
EditGroupName.Visibility = Visibility.Visible;
```

```
//display the right button to change the group's publicity:
```

```
(this.group.Publicity ? MakePublic_btn : MakePrivate_btn).Visibility =  
Visibility.Collapsed;
```

```
}
```

```
}
```

```
/// <summary>
```

```
/// refresh the members list of the group. add editing options
```

```
/// for admins
```

```
/// </summary>
```

```
private void RefreshGMembers()
```

```
{
```

```
    GMember gMember = new GMember();
```

```
    gMember.User = this.main.User;
```

```
    gMember.Group = this.group;
```

```
    this.isAdmin = srv.IsGroupAdmin(gMember);
```

```
    List<GMember> gMembers = srv.GetGroupMembers(this.group).ToList();
```

```
    MemberListStack.Children.Clear();
```

```
    foreach (GMember member in gMembers)
```

```
    {
```

```
        WrapPanel wrapPanel = new WrapPanel();
```

```
        EntityButton uButton = new EntityButton();//button for member-user's feed  
        with info
```

```
TextBlock name = new TextBlock();//user's name
```

```
uButton.Style = FindResource("SR_Buttons") as Style;
```

```
uButton.Content = member.User.Uname;
```

```
uButton.Entity = member.User;
```

```
uButton.Click += UFeed;
```

```
name.Style = FindResource("textS") as Style;
```

```
name.Foreground = Brushes.Black;
```

```
name.Text = member.User.Fname + " " + member.User.Lname;
```

```
name.FontSize = 20;
```

```
wrapPanel.Children.Add(uButton as Button);
```

```
wrapPanel.Children.Add(name);
```

```
if (this.isAdmin)
```

```
{
```

```
EntityButton makeAdmin = new EntityButton();
```

```
EntityButton removeAdmin = new EntityButton();
```

```
EntityButton removeMember = new EntityButton();
```

```
makeAdmin.Style = removeAdmin.Style = removeMember.Style =  
FindResource("RW_Buttons") as Style;
```

```
makeAdmin.Content = "Make Admin";
```

```
removeAdmin.Content = "Remove Admin";
```

```
removeMember.Content = "Remove Member";
```

```
makeAdmin.Entity = removeAdmin.Entity = removeMember.Entity=  
member;
```

```
makeAdmin.Click += MakeAdmin_Click;
```

```
removeAdmin.Click += RemoveAdmin_Click;
```

```
removeMember.Click += RemoveMember_Click;
```

Visibility.Collapsed;

wrapPanel.Children.Add(makeAdmin as Button);

wrapPanel.Children.Add(removeAdmin as Button);

wrapPanel.Children.Add(removeMember as Button);

}

else

{

if(member.Admin)

{

TextBlock adminText = new TextBlock();

adminText.Style = FindResource("textS") as Style;

adminText.Foreground = Brushes.Black;

adminText.Text = "Admin";

adminText.FontSize = 30;

wrapPanel.Children.Add(adminText);

}

}

MemberListStack.Children.Add(wrapPanel);

}

}

//make a user admin in a group button click

private void MakeAdmin\_Click(object sender, RoutedEventArgs e)

{

```
EntityButton eButton = sender as EntityButton;
```

```
GMember member = eButton.Entity as GMember;
```

```
member.Admin = true;
```

```
srv.UpdateGMemberStatus(member);
```

```
}
```

```
//remove admin from a user button click
```

```
private void RemoveAdmin_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
EntityButton eButton = sender as EntityButton;
```

```
GMember member = eButton.Entity as GMember;
```

```
member.Admin = false;
```

```
srv.UpdateGMemberStatus(member);
```

```
}
```

```
// delete group member button click
```

```
private void RemoveMember_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
EntityButton eButton = sender as EntityButton;
```

```
GMember member = eButton.Entity as GMember;
```

```
srv.DeleteGMember(member);
```

```
}
```

```
//add member to the group button click
```

```
private void AddMemberBtn_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
MemberAddStack.Visibility = Visibility.Visible;
```

```
((Button)sender).Visibility = Visibility.Collapsed;
```

}

//clear user search for adding button click

private void Clear\_Click(object sender, RoutedEventArgs e)

{

USerach\_TB.Text = "";

}

//make group public button click

private void MakePublic\_btn\_Click(object sender, RoutedEventArgs e)

{

this.group.Publicity = true;

srv.UpdateGroup(this.group);

Thread.Sleep(2000); //wait for while, to make sure this would became public  
before the update is done

this.group = srv.GetGroupByID(this.group.ID);

MakePublic\_btn.Visibility = group.Publicity ? Visibility.Collapsed :  
Visibility.Visible;

MakePrivate\_btn.Visibility = group.Publicity ? Visibility.Visible :  
Visibility.Collapsed;

PublicityDisplayText.Text = this.group.Publicity ? "Public" : "Private";

}

//make the group private button click

private void MakePrivate\_btn\_Click(object sender, RoutedEventArgs e)

{

this.group.Publicity = false;

srv.UpdateGroup(this.group);

Thread.Sleep(2000);

```
this.group = srv.GetGroupByID(this.group.ID);
```

```
MakePublic_btn.Visibility = group.Publicity ? Visibility.Collapsed :  
Visibility.Visible;
```

```
MakePrivate_btn.Visibility = group.Publicity ? Visibility.Visible :  
Visibility.Collapsed;
```

```
PublicityDisplayText.Text = this.group.Publicity ? "Public" : "Private";
```

```
}
```

```
private void USerach_TB_TextChanged(object sender, TextChangedEventArgs  
e)
```

```
{
```

```
TextBox textBox = sender as TextBox;
```

```
MemberAddStack_Results.Children.Clear();
```

```
if (textBox.Text.Length > 0)
```

```
{
```

```
List<User> userlist = srv.SelectUserByUName(textBox.Text).ToList();
```

```
foreach (User user in userlist)
```

```
{
```

```
WrapPanel wrapPanel = new WrapPanel();
```

```
EntityButton uButton = new EntityButton();
```

```
EntityButton addAsMember = new EntityButton();
```

```
TextBlock name = new TextBlock();
```

```
uButton.Style = FindResource("SR_Buttons") as Style;
```

```
uButton.Content = user.Uname;
```

```
uButton.Entity = user;
```

```
uButton.Click += UFeed;
```

```
name.Style = FindResource("textS") as Style;
```

```
name.Foreground = Brushes.Black;
```

```
name.Text = user.Fname + " " + user.Lname;
```

```
name.FontSize = 20;
```

```
addAsMember.Style = FindResource("SR_Buttons") as Style;
```

```
addAsMember.Content = "Add";
```

```
addAsMember.Entity = user;
```

```
addAsMember.Click += AddAsMember_Click;
```

```
wrapPanel.Children.Add(uButton as Button);
```

```
wrapPanel.Children.Add(name);
```

```
wrapPanel.Children.Add(addAsMember as Button);
```

```
MemberAddStack_Results.Children.Add(wrapPanel);
```

```
}
```

```
}
```

```
}
```

```
private void UFeed(object sender, RoutedEventArgs e)
```

```
{
```

```
    EntityButton userButton = sender as EntityButton;
```

```
    this.main.ChangeContent.Content = new UserFeed((User)userButton.Entity,  
this.main, this);
```

```
}
```

```
//add user to group
```

```
private void AddAsMember_Click(object sender, RoutedEventArgs e)
```



```
{

    EntityButton userButton = sender as EntityButton;

    srv.InsertGMemember(this.group, (User)userButton.Entity);

    Thread.Sleep(2000); //wait 2 seconds before refreshing

    RefreshGMembers();

}

//stop showing the adding options. stop showing the user search

private void CloseAdd_Click(object sender, RoutedEventArgs e)

{

    MemberAddStack.Visibility = Visibility.Collapsed;

    AddMemberBtn.Visibility = Visibility.Visible;

    Clear_Click(sender, e);

}

//open editing the group name

private void EditGroupName_Click(object sender, RoutedEventArgs e)

{

    GNameTxt.Text = this.group.GNname;

    GNameTxt.IsReadOnly = false;

    GNameTxt.BorderBrush = Brushes.Black;

    SaveGroupName.Visibility = Visibility.Visible;

    EditGroupName.Visibility = Visibility.Collapsed;

}

//save group's name

private void SaveGroupName_Click(object sender, RoutedEventArgs e)

{
```

```
GNameTxt.IsReadOnly = true;
```

```
this.group.GNname = GNameTxt.Name;
```

```
GNameTxt.Text = this.group.GNname + "#" + this.group.ID;
```

```
GNameTxt.BorderBrush = Brushes.Transparent;
```

```
srv.UpdateGroup(group);
```

```
SaveGroupName.Visibility = Visibility.Collapsed;
```

```
EditGroupName.Visibility = Visibility.Visible;
```

```
}
```

```
//refresh all members
```

```
private void RefreshMembers_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    RefreshGMembers();
```

```
}
```

```
//exit this UserControl and move to the previous one
```

```
private void Back_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.main.ChangeContent.Content = this.previous;
```

```
}
```

```
private void OpenChat_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.main.ChangeContent.Content = new ChatWithGroup(this.main,  
this.group, this);
```

```
}  
  
}  
  
}
```

Login.xaml:

```
<Window x:Class="ChatStream.Login"  
  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  
  
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  
  
    xmlns:local="clr-namespace:ChatStream"  
  
    mc:Ignorable="d"  
  
    Title="Login" Height="450" Width="800">  
  
<Window.Resources>  
  
    <ResourceDictionary>  
  
        <Style x:Key="buttonLS" TargetType="Button">  
  
            <Style.Resources>  
  
                <Style TargetType="Border">  
  
                    <Setter Property="CornerRadius" Value="5"/>  
  
                </Style>  
  
            </Style.Resources>  
  
            <Setter Property="Foreground" Value="LightYellow" />  
  
            <Setter Property="FontSize" Value="25" />  
  
            <Setter Property="Height" Value="50" />  
  
            <Setter Property="Width" Value="120" />
```

<Setter Property="Margin" Value="10" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="#00B1F9" />

</Style>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Rectangle Style="{StaticResource MainBackground}" />

<Grid >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="10\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="10\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid x:Name="Change" Grid.Row="1" Grid.Column="1" >

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="3\*" />

<RowDefinition Height="\*" />

```
<TextBlock Text="Login" Grid.Row="0" FontSize="60" Foreground="White"
VerticalAlignment="Bottom" HorizontalAlignment="Center"/>
```

```
<Grid Grid.Row="1">
```

```
<Grid.ColumnDefinitions>
```

```
<ColumnDefinition Width="*" />
```

```
<ColumnDefinition Width="*" />
```

```
</Grid.ColumnDefinitions>
```

```
<Grid.RowDefinitions>
```

```
<RowDefinition Height="*" />
```

```
<RowDefinition Height="*" />
```

```
</Grid.RowDefinitions>
```

```
<TextBox Grid.Column="1" Grid.Row="0" Margin="20" FontSize="35"
x:Name="UName" Opacity="0.55" Height="60" />
```

```
<PasswordBox Grid.Column="1" Grid.Row="1" Margin="20"
FontSize="35" x:Name="UPassword" Opacity="0.55" Height="60" />
```

```
<TextBlock Grid.Column="0" Grid.Row="0" Margin="20" Text="User
Name" FontSize="40" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="1" Margin="20"
Text="Password" FontSize="40" Foreground="WhiteSmoke"
TextAlignment="Center"/>
```

```
</Grid>
```

```
<WrapPanel Grid.Row="2" VerticalAlignment="Top"
HorizontalAlignment="Center">
```

```
<Button Content="Enter" Style="{StaticResource buttonLS}"
Click="EnterB" />
```

```
<Button Content="SignUp" Style="{StaticResource buttonLS}"
Click="SignUpW" />
```

</Grid>

</Grid>

</Grid>

</Window>

Login.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

using System.Text.RegularExpressions;

namespace ChatStream

{

/// <summary>

/// Interaction logic for Login.xaml

/// </summary>

public partial class Login : Window

{

Service1Client srv = new Service1Client();

public Login()

{

InitializeComponent();

}

private void SignUpW(object sender, RoutedEventArgs e)

{

SignUp signUp = new SignUp();

signUp.Show();

this.Close();

}

/// <summary>

/// the button click collects the user's password and username, and logs in if the  
info

/// is valid

/// </summary>

/// <param name="sender">button</param>

/// <param name="e"></param>

private void EnterB(object sender, RoutedEventArgs e)

{

User user = null;

if(UName.Text.Length >= 3 && UPassword.Password.Length >= 4)

{

if(!Regex.IsMatch(UName.Text, "^[a-zA-Z0-9]\*\$")) //condition checks if  
username is numbers and letters only.

{

MessageBox.Show("User name should contain only letters and  
numbers");

}

else

{

//tries to log in:

user = srv.Login(UName.Text, UPassword.Password);

if (user != null && user.ID > 0)

{//succsessful login

MainWindow main = new MainWindow(user);

main.Show();

this.Close();

}

else

{//failed to login

MessageBox.Show("Failed to login. Check Your User name and  
password!");

}

}

}



```
else
{
    MessageBox.Show("Username should contain at least 3 characters and
password at least 4");
}
}
}
}
```

MainWindow.xaml:

```
<Window x:Class="ChatStream.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:ChatStream"
    mc:Ignorable="d"
    Title="MainWindow" Height="450" Width="800">
<Window.Resources>
    <ResourceDictionary>
        <Style x:Key="Menu_Button" TargetType="Button">
            <Style.Resources>
                <Style TargetType="Border">
                    <Setter Property="CornerRadius" Value="5"/>
                </Style>
            </Style.Resources>
        </Style>
    </ResourceDictionary>
</Window.Resources>
```

</Style.Resources>

<Setter Property="Foreground" Value="LightYellow" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Height" Value="50" />

<Setter Property="Width" Value="120" />

<Setter Property="Margin" Value="10" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="#00B1F9" />

</Style>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Rectangle Style="{StaticResource MainBackground}" />

<Grid >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="" />

<ColumnDefinition Width="20\*" />

<ColumnDefinition Width="" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="5\*" />

<RowDefinition Height="20\*" />

<RowDefinition Height="4\*" />

</Grid.RowDefinitions>

```
<WrapPanel Grid.Row="0" Grid.Column="1" VerticalAlignment="Top"
HorizontalAlignment="Center">
```

```
<Button Content="Chats" Style="{StaticResource Menu_Button}"
Click="Chat_Click" />
```

```
<Button Content="MyFeed" Style="{StaticResource Menu_Button}"
Click="FeedB" />
```

```
<Button Content="Friends" Style="{StaticResource Menu_Button}"
Click="FriendB_click"/>
```

```
<Button Content="Profile" Style="{StaticResource Menu_Button}"
Click="ProfileB" />
```

```
<Button Content="Log Out" Style="{StaticResource Menu_Button}"
Click="OutB" />
```

```
</WrapPanel>
```

```
<TextBlock Text="" Grid.Column="1" Grid.Row="2" FontSize="40"
Foreground="LightYellow" VerticalAlignment="Top" HorizontalAlignment="Center"
x:Name="HelloT"/>
```

```
<ContentControl x:Name="ChangeContent" Grid.Row="1" Grid.Column="1"/>
```

```
</Grid>
```

```
</Grid>
```

```
</Window>
```

MainWindow.xaml.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Service1Client srv = new Service1Client();

private User user;

public User User { get => user; set => user = value; }

/// <summary>

```
/// initialize main window
```

```
/// </summary>
```

```
/// <param name="user">user that has been logged</param>
```

```
public MainWindow(User user)
```

```
{
```

```
    InitializeComponent();
```

```
    this.ChangeContent.Content = new OpenMenu();
```

```
    this.user = user;
```

```
    HelloT.Text += "Hello " + user.Username;
```

```
}
```

```
//Exit MainWindow, Log out, moves to login window
```

```
private void OutB(object sender, RoutedEventArgs e)
```

```
{
```

```
    (new Login()).Show();
```

```
    this.Close();
```

```
}
```

```
// opens profile's data with edit options
```

```
private void ProfileB(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.ChangeContent.Content = new Profile(this.user);
```

```
}
```

//opens this main user's personal feed

```
private void FeedB(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.ChangeContent.Content = new Feed(this);
```

```
}
```

//opens friend list with option to add friends

```
private void FriendB_click(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.ChangeContent.Content = new FriendListShow(this);
```

```
}
```

//opens chats menu

```
private void Chat_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    this.ChangeContent.Content = new ChatsShow(this);
```

```
}
```

//gets current UserControl Displayed

```
public UserControl GetCurrentControle()
```

```
{
```

```
    UserControl control = null;
```

```
    Application.Current.Dispatcher.Invoke(() =>
```

```
{
```

```
        if (ChangeContent.Content is UserControl userControl)
```

```
{
```

```
control = userControl;

// Do something with 'control'

}

});

return control;

}

}

}
```

OpenMenu.Xaml:

```
<UserControl x:Class="ChatStream.OpenMenu"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:ChatStream"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="5*" />

<RowDefinition Height="2*" />

</Grid.RowDefinitions>
```

```
<TextBlock Text="Welcome" Grid.Row="0" FontSize="120" Foreground="White"
VerticalAlignment="Bottom" HorizontalAlignment="Center"/>
```

```
<TextBlock Text="ChatStream Menu" Grid.Row="1" FontSize="40"
Foreground="#C9FFF6" VerticalAlignment="Top" HorizontalAlignment="Center"/>
```

```
</Grid>
```

```
</UserControl>
```

PostView.xaml:

```
<UserControl x:Class="ChatStream.PostView"
```

```
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
```

```
xmlns:local="clr-namespace:ChatStream"
```

```
mc:Ignorable="d"
```

```
d:DesignHeight="450" d:DesignWidth="800">
```

```
<UserControl.Resources>
```

```
<ResourceDictionary>
```

```
<Style x:Key="Post_Buttons" TargetType="Button">
```

```
<Style.Resources>
```

```
<Style TargetType="Border">
```

```
<Setter Property="CornerRadius" Value="5"/>
```

```
</Style>
```

```
</Style.Resources>
```



<Setter Property="BorderBrush" Value="Transparent" />

<Setter Property="Foreground" Value="Black" />

<Setter Property="FontSize" Value="25" />

<Setter Property="Height" Value="46" />

<Setter Property="Margin" Value="10, 2" />

<Setter Property="BorderThickness" Value="2" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid Background="LightBlue">

<Grid.RowDefinitions>

<RowDefinition Height="auto" />

<RowDefinition Height="auto" />

<RowDefinition Height="auto" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0" VerticalAlignment="Top" HorizontalAlignment="Left">

<Button Style="{StaticResource Post\_Buttons}" x:Name="PUName" Content="User Name" Click="UFeed" Background="Transparent"/>

<Button Style="{StaticResource Post\_Buttons}" x:Name="FollowB" Content="Follow" Background="Transparent" Click="FollowP"/>

<Button Style="{StaticResource Post\_Buttons}" x:Name="UnFollowB" Content="UnFollow" Background="Transparent" Click="UnFollowP"/>

</WrapPanel>

```
<TextBox TextWrapping="Wrap" Grid.Row="1" x:Name="Content" Margin="5"
Background="GhostWhite" IsReadOnly="True" FontSize="25"/>

<Button Click="EditContent" Grid.Row="1" x:Name="Edit" Content="Edit"
FontSize="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="10"
Visibility="Collapsed"/>

<Button Click="SaveContent" Grid.Row="1" x:Name="Save" Content="Save"
FontSize="25" VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="10"
Visibility="Collapsed"/>

<Grid Grid.Row="2">

    <Grid.RowDefinitions>

        <RowDefinition Height="*" />

        <RowDefinition Height="*" />

    </Grid.RowDefinitions>

    <WrapPanel Grid.Row="0">

        <Button Style="{StaticResource Post_Buttons}" FontFamily="Rockwell"
Content="View Comments" Background="#fff0ab" Click="View_Comments_B" />

        <Button Style="{StaticResource Post_Buttons}" Content="Like"
x:Name="Like_Button" FontFamily="Helvetica" Click="Like_Click"/>

        <Button Style="{StaticResource Post_Buttons}" Content="UnLike"
x:Name="UnLike_Button" FontFamily="Helvetica" Click="UnLike_Click"/>

        <TextBlock x:Name="LCounter" Text="no" FontSize="30" Margin="10,0"/>

        <TextBlock Text="likes" FontSize="30"/>

        <ComboBox Margin="10" Width="100" x:Name="publicity"
SelectionChanged="ComboBox_SelectionChanged" Visibility="Collapsed">

            <ComboBoxItem Tag="Public" Content="Public" />

            <ComboBoxItem Tag="Private" Content="Private" />

        </ComboBox>
```

```
<ScrollView Height="150" Grid.ColumnSpan="2" x:Name="CViewer"
Grid.Row="1" Visibility="Collapsed">
```

```
<StackPanel x:Name="Comments_List_View">
```

```
</StackPanel>
```

```
</ScrollView>
```

```
</Grid>
```

```
<TextBox x:Name="My_Comment_Content" TextWrapping="Wrap"
Grid.Row="4" Margin="7" FontSize="20" />
```

```
<Button Grid.Row="3" Content="Send" FontSize="25"
VerticalAlignment="Bottom" HorizontalAlignment="Right" Margin="10"
Click="SendComment_Click" />
```

```
</Grid>
```

```
</UserControl>
```

PostView.xaml.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
using System.Windows;
```

```
using System.Windows.Controls;
```

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • 2167164, כרמיאל 71, פסגה 71  
using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for PostView.xaml

/// </summary>

public partial class PostView : UserControl

{

Service1Client srv = new Service1Client();

protected MainWindow main;

protected Post post;

private UserControl current;

public PostView(Post post, MainWindow main, UserControl current) : this(post, main)

{

this.current = current;

}

public PostView(Post post, MainWindow main)

{

    this.current = null;

    InitializeComponent();

    int likeCounter = srv.CountLikes(post);

    LCounter.Text = likeCounter.ToString();

    this.main = main;

    this.post = post;

    Like likedPost = new Like();

    likedPost.Post = this.post;

    likedPost.User = this.main.User;

    Follow followedUser = new Follow();

    followedUser.Follower = main.User;

    followedUser.Following = post.User;

    if (srv.IsLiked(likedPost))

    {

        UnLike\_Button.Visibility = Visibility.Visible;

        Like\_Button.Visibility = Visibility.Collapsed;

    }

    else

    {

        UnLike\_Button.Visibility = Visibility.Collapsed;

        Like\_Button.Visibility = Visibility.Visible;

    }

```
{
    Edit.Visibility = Visibility.Visible;

    publicity.Visibility = Visibility.Visible;

    FollowB.Visibility = Visibility.Collapsed;

    UnFollowB.Visibility = Visibility.Collapsed;
}

else if(srv.IsFollowing(followedUser))
{
    FollowB.Visibility = Visibility.Collapsed;

    UnFollowB.Visibility = Visibility.Visible;
}

else
{
    UnFollowB.Visibility = Visibility.Collapsed;

    FollowB.Visibility = Visibility.Visible;
}

Content.Text = post.Content;

PUName.Content = this.post.User.Username;

if (post.IsPrivate)
{
    //publicity.SelectedIndex = 2;

    publicity.SelectedItem = publicity.Items[1];
}
```

else

{

//publicity.SelectedItem = 1;

publicity.SelectedItem = publicity.Items[0];

}

}

private void UFeed(object sender, RoutedEventArgs e)

{

this.main.ChangeContent.Content = new UserFeed(this.post.User, this.main, this.current);

}

private void FollowP(object sender, RoutedEventArgs e)

{

FollowB.Visibility = Visibility.Collapsed;

UnFollowB.Visibility = Visibility.Visible;

Follow follow = new Follow();

follow.Follower = main.User;

follow.Following = post.User;

srv.InsertFollow(follow);

}

private void UnFollowP(object sender, RoutedEventArgs e)

{

UnFollowB.Visibility = Visibility.Collapsed;

FollowB.Visibility = Visibility.Visible;

Follow follow = new Follow();

follow.Follower = main.User;

follow.Following = post.User;

srv.DeleteFollow(follow);

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = sender as ComboBox;

if(comboBox.SelectedIndex == 1 && (!this.post.IsPrivate))

{

this.post.IsPrivate = (!this.post.IsPrivate);

srv.UpdatePost(post);

}

if (comboBox.SelectedIndex == 0 && this.post.IsPrivate)

{

this.post.IsPrivate = (!this.post.IsPrivate);

srv.UpdatePost(post);

}

}

private void EditContent(object sender, RoutedEventArgs e)

{



Content.IsReadOnly = false;

Edit.Visibility = Visibility.Collapsed;

Save.Visibility = Visibility.Visible;

}

private void SaveContent(object sender, RoutedEventArgs e)

{

Content.IsReadOnly = true;

Edit.Visibility = Visibility.Visible;

Save.Visibility = Visibility.Collapsed;

post.Content = Content.Text;

srv.UpdatePost(post);

}

private void Like\_Click(object sender, RoutedEventArgs e)

{

Like like = new Like();

like.Post = this.post;

like.User = this.main.User;

this.srv.InsertLike(like);

LCounter.Text = this.srv.CountLikes(this.post).ToString();

UnLike\_Button.Visibility = Visibility.Visible;

Like\_Button.Visibility = Visibility.Collapsed;

}

```
private void UnLike_Click(object sender, RoutedEventArgs e)
```

```
{
    Like like = new Like();
    like.Post = this.post;
    like.User = this.main.User;
    srv.DeleteLike(like);
    LCounter.Text = srv.CountLikes(this.post).ToString();
    UnLike_Button.Visibility = Visibility.Collapsed;
    Like_Button.Visibility = Visibility.Visible;
}
```

```
private void View_Comments_B(object sender, RoutedEventArgs e)
```

```
{
    Button button = sender as Button;
    if(CViewer.Visibility == Visibility.Collapsed)
    {
        CViewer.Visibility = Visibility.Visible;
        List<Comment> comments = srv.SelectCommentByPost(this.post).ToList();
        foreach (Comment comment in comments)
        {
            UserControl userControl = new UserControl();
            userControl.Content = new CommentView(comment, main, this.current);
            userControl.Margin = new Thickness(5);
            Comments_List_View.Children.Add(userControl);
        }
    }
}
```

button.Content = "^^stop comments view^^";

}

else

{

button.Content = "View Comments";

Comments\_List\_View.Children.Clear();

CViewer.Visibility = Visibility.Collapsed;

}

}

private void SendComment\_Click(object sender, RoutedEventArgs e)

{

if(My\_Comment\_Content.Text.Length >= 1)

{

Comment comment = new Comment();

comment.User = this.main.User;

comment.Post = this.post;

comment.Content = My\_Comment\_Content.Text;

srv.InsertComment(comment);

My\_Comment\_Content.Text = "";

}

}

}

}

<UserControl x:Class="ChatStream.Profile"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:ChatStream"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

<ResourceDictionary>

<Style x:Key="buttonLS" TargetType="Button">

<Style.Resources>

<Style TargetType="Border">

<Setter Property="CornerRadius" Value="5"/>

</Style>

</Style.Resources>

<Setter Property="Foreground" Value="LightYellow" />

<Setter Property="FontSize" Value="27" />

<Setter Property="Height" Value="40" />

<Setter Property="Width" Value="120" />

<Setter Property="Margin" Value="10" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="#00B1F9" />

</Style>

<Style x:Key="TextBoxPForm" TargetType="TextBox">

<Setter Property="FontSize" Value="25" />

<Setter Property="Margin" Value="5" />

<Setter Property="Opacity" Value="0.55" />

<Setter Property="Height" Value="45" />

<Setter Property="IsReadOnly" Value="True" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="2\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBox Grid.Column="1" Grid.Row="0" x:Name="UName"  
Style="{StaticResource TextBoxPForm}" />

```
<TextBox Grid.Column="1" Grid.Row="1" Style="{StaticResource
TextBoxPForm}" x:Name="FName"/>
```

```
<TextBox Grid.Column="1" Grid.Row="2" x:Name="Lname"
Style="{StaticResource TextBoxPForm}"/>
```

```
<TextBox Grid.Column="1" Grid.Row="3" x:Name="Description"
Style="{StaticResource TextBoxPForm}" Height="auto"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="0" Margin="1" Text="User Name"
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="1" Margin="1" Text="First Name"
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="2" Margin="1" Text="Last Name"
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="3" Margin="1" Text="Description"
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<Button x:Name="Edit" Content="Edit" Style="{StaticResource buttonLS}"
Grid.Column="0" Grid.Row="4" Click="Edit_Click" />
```

```
<Button x:Name="Save" Content="Save" Style="{StaticResource buttonLS}"
Grid.Column="1" Grid.Row="4" Visibility="Collapsed" Click="Save_Click" />
```

```
</Grid>
```

```
</UserControl>
```

Profile.xaml.cs:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
using System.Windows;
```

```
using System.Windows.Controls;
```

פסגה 71, כרמיאל 2167164, שכונה מערבית • טל. 04-6668301 • פקס. 04-9880446 • מייל. psagot@psagot.ort.org.il • www.psagot.ort.org.il  
using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for Profile.xaml

/// </summary>

public partial class Profile : UserControl

{

Service1Client srv = new Service1Client();

protected User user;

public Profile(User user)

{

InitializeComponent();

this.user = user;

UName.Text = user.Uname;

Lname.Text = user.Lname;

FName.Text = user.Fname;

Description.Text = user.Description;

}

private void Edit\_Click(object sender, RoutedEventArgs e)

{

Save.Visibility = Visibility.Visible;

Edit.Visibility = Visibility.Collapsed;

UName.IsReadOnly = false;

FName.IsReadOnly = false;

Lname.IsReadOnly = false;

Description.IsReadOnly = false;

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

Save.Visibility = Visibility.Collapsed;

Edit.Visibility = Visibility.Visible;

user.Uname = UName.Text;

user.Lname = Lname.Text;

user.Fname = FName.Text;

user.Description = Description.Text;

srv.UpdateUser(this.user);

UName.IsReadOnly = true;

FName.IsReadOnly = true;

Lname.IsReadOnly = true;



Description.IsReadOnly = true;

```
}
}
}
```

SignUp.xaml:

```
<Window x:Class="ChatStream.SignUp"

    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

    xmlns:local="clr-namespace:ChatStream"

    mc:Ignorable="d"

    Title="SignUp" Height="450" Width="800">

<Window.Resources>

    <ResourceDictionary>

        <Style x:Key="buttonLS" TargetType="Button">

            <Style.Resources>

                <Style TargetType="Border">

                    <Setter Property="CornerRadius" Value="5"/>

                </Style>

            </Style.Resources>

            <Setter Property="Foreground" Value="LightYellow" />

            <Setter Property="FontSize" Value="27" />

            <Setter Property="Height" Value="40" />
```

<Setter Property="Width" Value="120" />

<Setter Property="Margin" Value="10" />

<Setter Property="BorderThickness" Value="2" />

<Setter Property="Background" Value="#00B1F9" />

</Style>

</ResourceDictionary>

</Window.Resources>

<Grid>

<Rectangle Style="{StaticResource MainBackground}" />

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="10\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="20\*" />

</Grid.RowDefinitions>

<Grid x:Name="Change" Grid.Row="0" Grid.Column="1" >

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="5\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBlock Text="Sign Up" Grid.Row="0" FontSize="50"  
Foreground="White" VerticalAlignment="Bottom" HorizontalAlignment="Center"/>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<TextBox Grid.Column="1" Grid.Row="0" Margin="5" FontSize="25"  
x:Name="UName" Opacity="0.55" Height="45"/>

<PasswordBox Grid.Column="1" Grid.Row="1" Margin="5"  
FontSize="25" x:Name="UPassword1" Opacity="0.55" />

<PasswordBox Grid.Column="1" Grid.Row="2" Margin="5"  
FontSize="25" x:Name="UPassword2" Opacity="0.55" Height="45"/>

<TextBox Grid.Column="1" Grid.Row="3" Margin="5" FontSize="25"  
x:Name="FName" Opacity="0.55" Height="45"/>

<TextBox Grid.Column="1" Grid.Row="4" Margin="5" FontSize="25"  
x:Name="Lname" Opacity="0.55" Height="45"/>

```
<TextBox Grid.Column="1" Grid.Row="5" Margin="5" FontSize="25"
x:Name="EMail" Opacity="0.55" Height="45"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="0" Margin="1" Text="User
Name" FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="1" Margin="1"
Text="Password" FontSize="30" Foreground="WhiteSmoke"
TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="2" Margin="1" Text="Password
again" FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="3" Margin="1" Text="First
Name" FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="4" Margin="1" Text="Last
Name" FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
<TextBlock Grid.Column="0" Grid.Row="5" Margin="1" Text="Email"
FontSize="30" Foreground="WhiteSmoke" TextAlignment="Center"/>
```

```
</Grid>
```

```
<WrapPanel Grid.Row="2" VerticalAlignment="Top"
HorizontalAlignment="Center">
```

```
<Button Content="create user" Style="{StaticResource buttonLS}"
Click="EnterBtn" Width="150" />
```

```
<Button Content="Login" Style="{StaticResource buttonLS}"
Click="LoginW" />
```

```
</WrapPanel>
```

```
</Grid>
```

```
</Grid>
```

```
</Grid>
```

```
</Window>
```

SignUp.xaml.cs:

www.psagot.ort.org.il • psagot@psagot.ort.org.il • מייל • 04-9880446 • פקס • 04-6668301 • טל • שכונה מערבית • פסגה 71, כרמיאל 2167164

```
using System;  
  
using System.Collections.Generic;  
  
using System.Linq;  
  
using System.Text;  
  
using System.Threading.Tasks;  
  
using System.Windows;  
  
using System.Windows.Controls;  
  
using System.Windows.Data;  
  
using System.Windows.Documents;  
  
using System.Windows.Input;  
  
using System.Windows.Media;  
  
using System.Windows.Media.Imaging;  
  
using System.Windows.Shapes;  
  
using ChatStream.ServiceReference1;  
  
using System.Text.RegularExpressions;  
  
using System.Net.Mail;
```

```
namespace ChatStream
```

```
{
```

```
    /// <summary>
```

```
    /// Interaction logic for SignUp.xaml
```

```
    /// </summary>
```

```
    public partial class SignUp : Window
```

```
{
```

```
Service1Client srv = new Service1Client();
```

```
public SignUp()
```

```
{
```

```
    InitializeComponent();
```

```
}
```

```
private void LoginW(object sender, RoutedEventArgs e)
```

```
{
```

```
    Login login = new Login();
```

```
    login.Show();
```

```
    this.Close();
```

```
}
```

```
/// <summary>
```

```
/// method checks if entered data is valid.
```

```
/// </summary>
```

```
/// <param name="user">users data</param>
```

```
/// <returns>true if valid, false if not</returns>
```

```
private bool IsUserValid(User user)
```

```
{
```

```
    bool valid = true;
```

```
    if (!Regex.IsMatch(user.Username, "[a-zA-Z0-9]*$") && user.Username.Length >
```

3)

```
{
```

```
    valid = false;
```

```

        MessageBox.Show("User name Should contain only letters and/or
        numbers\n" +

            "and the length is 3 characters minimum");

    }

    if (!Regex.IsMatch(user.Lname, "[a-zA-Z0-9]*$") && user.Lname.Length > 3)

    {

        valid = false;

        MessageBox.Show("last name Should contain only letters and/or
        numbers\n" +

            "and the length is 3 characters minimum");

    }

    if (!Regex.IsMatch(user.Fname, "[a-zA-Z0-9]*$") && user.Fname.Length > 3)

    {

        MessageBox.Show("first name Should contain only letters and/or
        numbers\n" +

            "and the length is 3 characters minimum");

        valid = false;

    }

    try//check if email is in valid format by trying create MailAddress object.

    {

        MailAddress mailAddress = new MailAddress(user.Email);

    }

    catch (FormatException)

    {

        valid = false;

        MessageBox.Show("Not valid Email, check again your mail adress");
    }

```

}

return valid;

}

/// <summary>

/// adds user to the server if valid data was entered. than logins and moves

/// to the main window

/// </summary>

/// <param name="sender">button</param>

/// <param name="e"></param>

private void EnterBtn(object sender, RoutedEventArgs e)

{

User user = new User();

if (UPassword1.Password == UPassword2.Password)

{

//add info to a User object

user.Uname = UName.Text;

user.Lname = Lname.Text;

user.Fname = FName.Text;

user.Password = UPassword1.Password;

user.EntrDate = DateTime.Now;

user.Description = user.Uname;

user.Email = EMail.Text;

if (IsValidUser(user))

{

srv.InsertUser(user);



```
user = srv.Login(user.Uname, user.Password);
```

```
if (user != null)
```

```
{
```

```
    MainWindow main = new MainWindow(user);
```

```
    main.Show();
```

```
    this.Close();
```

```
}
```

```
else
```

```
{
```

```
    MessageBox.Show("Failed to enter, try another username");
```

```
}
```

```
}
```

```
}
```

```
else
```

```
{
```

```
    MessageBox.Show("The First password and the Second password don't  
match!");
```

```
}
```

```
}
```

```
}
```

```
}
```

UserFeed.xaml:

```
<UserControl x:Class="ChatStream.UserFeed"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:ChatStream"

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800">

<UserControl.Resources>

<ResourceDictionary>

<Style x:Key="FormTextBox" TargetType="TextBlock">

<Setter Property="Margin" Value="1" />

<Setter Property="FontSize" Value="30" />

<Setter Property="Foreground" Value="White" />

<Setter Property="TextAlignment" Value="Center" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<ScrollView Grid.Row="1" VerticalAlignment="Top">

<StackPanel>

<TextBlock Text="User's Feed" FontSize="30" TextAlignment="Center"  
Foreground="White"/>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="2\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="auto" />

</Grid.RowDefinitions>

<WrapPanel Grid.Row="0">

<TextBlock x:Name="UName" Text="User Name: "  
Style="{StaticResource FormTextBox}" />

</WrapPanel>

<WrapPanel Grid.Row="1">

<TextBlock x:Name="FullName" Text="Name: "  
Style="{StaticResource FormTextBox}" />

</WrapPanel>

<WrapPanel Grid.Row="2">

<TextBlock x:Name="Description" Text="Description: "  
Style="{StaticResource FormTextBox}" />

</WrapPanel>

<WrapPanel Grid.Row="3">

<Button x:Name="FollowB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Follow" Click="FollowB\_Click"/>

<Button x:Name="UnFollowB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="UnFollow" Click="UnFollowB\_Click"/>

<Button x:Name="FriendB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Friend Request" Click="FriendB\_Click"  
Visibility="Collapsed"/>

<Button x:Name="UnFriendB" Style="{StaticResource RButtons}"  
Background="LightYellow" Content="Remove Friend" Click="UnFriendB\_Click"  
Visibility="Collapsed"/>

```
<Button x:Name="AcceptF_B" Style="{StaticResource RButtons}"
Background="LightYellow" Content="Accept Friend" Click="AcceptF_B_Click"
Visibility="Collapsed"/>
```

```
<Button x:Name="DeclineF_B" Style="{StaticResource RButtons}"
Background="LightYellow" Content="Decline Friendship" Click="UnFriendB_Click"
Visibility="Collapsed"/>
```

```
</WrapPanel>
```

```
<WrapPanel Grid.Row="4">
```

```
<Button x:Name="AddPost" Visibility="Collapsed"
Style="{StaticResource AVR_Button}" Width="200" Content="Add Post"
Click="AddPost_Click" />
```

```
<Button Style="{StaticResource AVR_Button}" Width="150"
Content="Refresh" Click="Refresh_Click" />
```

```
<Button Style="{StaticResource AVR_Button}" Width="100"
Content="Back" Click="Back_Click" />
```

```
</WrapPanel>
```

```
<ContentControl Grid.Row="5" x:Name="CreatePostV" />
```

```
</Grid>
```

```
<TextBlock Height="10" />
```

```
<StackPanel x:Name="FeedView">
```

```
</StackPanel>
```

```
</StackPanel>
```

```
</ScrollView>
```

```
</Grid>
```

```
</UserControl>
```

UserFeed.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using ChatStream.ServiceReference1;

namespace ChatStream

{

/// <summary>

/// Interaction logic for UserFeed.xaml

/// </summary>

public partial class UserFeed : UserControl

{

```
Service1Client srv = new Service1Client();
```

```
protected User user;
```

```
protected MainWindow main;
```

```
private UserControl previous;
```

```
public UserControl Previous { get => previous; set => previous = value; }
```

```
public UserFeed(User user, MainWindow main, UserControl previous) :  
this(user, main)
```

```
{
```

```
    this.previous = previous;
```

```
}
```

```
public UserFeed(User user, MainWindow main)
```

```
{
```

```
    this.previous = null;
```

```
    InitializeComponent();
```

```
    this.user = user;
```

```
    this.main = main;
```

```
    UName.Text += user.Uname;
```

```
    FullName.Text += user.Fname + " " + user.Lname;
```

```
    Description.Text += user.Description;
```

```
    List<Post> posts = srv.SelectUserFeed(this.main.User, this.user).ToList();
```

```
    Follow follow = new Follow();
```

```
    follow.Follower = this.main.User;
```

```
    follow.Following = this.user;
```

```
    if(srv.IsFollowing(follow))
```

```
{
    this.FollowB.Visibility = Visibility.Collapsed;

    this.UnFollowB.Visibility = Visibility.Visible;
}
else
{
    this.FollowB.Visibility = Visibility.Visible;
    this.UnFollowB.Visibility = Visibility.Collapsed;
}
if(user.ID == main.User.ID)
{
    AddPost.Visibility = Visibility.Visible;
    CreatePostV.Content = new CreatePost(this.user, AddPost);
    ((CreatePost)CreatePostV.Content).Visibility = Visibility.Collapsed;
}
foreach (Post post in posts)
{
    UserControl userControl = new UserControl();
    userControl.Content = new PostView(post, main, this);
    userControl.Margin = new Thickness(10);
    FeedView.Children.Add(userControl);
}
Friend friend = srv.SelectFriend(this.main.User, this.user);
```

```

if(friend == null)

{

    FriendB.Visibility = Visibility.Visible;

}

else

{

    if (friend.Approved)

    {

        UnFriendB.Visibility = Visibility.Visible;

    }

    else

    {

        DeclineF_B.Visibility = Visibility.Visible;

        if(friend.User2.ID == this.main.User.ID)

            AcceptF_B.Visibility = Visibility.Visible;

    }

}

}

private void FollowB_Click(object sender, RoutedEventArgs e)

{

    FollowB.Visibility = Visibility.Collapsed;

    UnFollowB.Visibility = Visibility.Visible;

    Follow follow = new Follow();

    follow.Follower = this.main.User;
    
```



```
follow.Following = this.user;
```

```
srv.InsertFollow(follow);
```

```
}
```

```
private void UnFollowB_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
UnFollowB.Visibility = Visibility.Collapsed;
```

```
FollowB.Visibility = Visibility.Visible;
```

```
Follow follow = new Follow();
```

```
follow.Follower = this.main.User;
```

```
follow.Following = this.user;
```

```
srv.DeleteFollow(follow);
```

```
}
```

```
public void Refresh_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
FeedView.Children.Clear();
```

```
List<Post> posts = srv.SelectUserFeed(this.main.User, this.user).ToList();
```

```
foreach (Post post in posts)
```

```
{
```

```
UserControl userControl = new UserControl();
```

```
userControl.Content = new PostView(post, main, this);
```

```
userControl.Margin = new Thickness(10);
```

```
FeedView.Children.Add(userControl);
```

```
}
```

}

```
private void AddPost_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    Button button = sender as Button;
```

```
    button.Visibility = Visibility.Collapsed;
```

```
    ((CreatePost)CreatePostV.Content).Visibility = Visibility.Visible;
```

```
}
```

```
private void Back_Click(object sender, RoutedEventArgs e)
```

```
{
```

```
    if(this.previous != null)
```

```
    {
```

```
        if (this.previous is UserFeed)
```

```
        {
```

```
            UserFeed feed = this.previous as UserFeed;
```

```
            feed.Refresh_Click(sender, e);
```

```
        }
```

```
        else if (this.previous is Feed)
```

```
        {
```

```
            Feed feed = this.previous as Feed;
```

```
            feed.Refresh_Click(sender, e);
```

```
        }
```

```
        this.main.ChangeContent.Content = this.previous;
```

```
}
```

}

private void FriendB\_Click(object sender, RoutedEventArgs e)

{

Friend friend = new Friend();

friend.User1 = this.main.User;

friend.User2 = this.user;

FriendB.Visibility = Visibility.Collapsed;

DeclineF\_B.Visibility = Visibility.Visible;

srv.InsertFriend(friend);

}

private void UnFriendB\_Click(object sender, RoutedEventArgs e)

{

Friend friend = new Friend();

friend.User1 = this.main.User;

friend.User2 = this.user;

UnFriendB.Visibility = Visibility.Collapsed;

DeclineF\_B.Visibility = Visibility.Collapsed;

AcceptF\_B.Visibility = Visibility.Collapsed;

FriendB.Visibility = Visibility.Visible;

srv.DeleteFriend(friend);

}

```
private void AcceptF_B_Click(object sender, RoutedEventArgs e)
{
    Friend friend = new Friend();
    friend.User1 = this.main.User;
    friend.User2 = this.user;
    UnFriendB.Visibility = Visibility.Visible;
    DeclineF_B.Visibility = Visibility.Collapsed;
    AcceptF_B.Visibility = Visibility.Collapsed;
    FriendB.Visibility = Visibility.Collapsed;
    srv.ApproveFriend(friend);
}
}
}
```