

**בית ספר תיכון ע"ש "אילן רמון" הוד השרון**

**Ramon High School – Hod Hasharon**



**עבודת גמר**

**בתכנון ותכנות מערכות**

**במסלול שירותי רשת אינטרנט**

**שנת הלימודים : התשפ"ג**

|  |  |
| --- | --- |
| שם התלמיד/ה: | איתי ברודר |
| ת.ז : | 328142070 |
| שם המורה:  תאריך: | סמדר וכטר  4.5.2023 |

# הוקרת תודה

אני רוצה להודות מכל ליבי למורות שלי סמדר ורחלי.

הגורם העיקרי שבלעדיה לא הייתי מצליח לסיים את הפרויקט הארוך והמורכב הזה, המורה סמדר, שהשקיעה שעות רבות במפגשים איתי, הקשיבה לכל הבעיות השונות בשעות בית הספר וגם לאחר סיום השעות, על חשבון הזמן האישי שלה. תמיד מצאתי אוזן קשבת ועזרה רבה ואני מעריך את זה מאד. קיבלתי הנחיות מפורטות החל מתכנון הפרויקט לפני שהתחלתי, דרך שלבים שונים בפיתוח שלו עם המון פידבקים על כל נקודה גדולה כקטנה, פתרון באגים שונים והסברים רבים עם המון סבלנות על הצורך בכתיבה נכונה של קוד, של תיעוד נרחב ומדויק ואני לוקח את כל העצות האלו לשנים רבות קדימה.

רחלי, השקיעה בנושאים שונים ואפשרה לי להישאר במעבדה ולעבוד על הפרויקט, עזרה לי ברעיונות שונים לפתרון קשיים שונים שעלו מידי פעם ותרמה מהידע הרב שלה תמיד בשמחה ובמקצועיות. אני לוקח איתי את ההקפדה על פרטים קטנים כגדולים, עצות והמון סבלנות כדוגמה לאופן בו צריך לעבוד עם תלמידים ואני מעריך מאד את התרומה שלה לפרויקט.

אני רוצה להודות להורי ובמיוחד לאימי שתמכה בי ודחפה אותי שלא לוותר ותמיד לקדם את הפרויקט הארוך והמורכב הזה כל פעם עוד כמה צעדים קדימה, למרות שיש לי השנה המון משימות רבות, בגרויות אוניברסיטה ומבדקים לצה"ל.

אני רוצה להודות גם לחברתי, שיראל, שעזרה לי להישאר מפוקס וממוקד במהלך הפרויקט, למרות אינסוף שעות שהשקעתי בו על חשבון הזמן המשותף ותמיד הייתה לי לכתף כשהייתי זקוק לה.

ברצוני להודות למשתמשי Stack Overflow הנהדרים שסיפקו מענה לבעיות שלי ולמייקרוסופט שללא האתר שלהם לא הייתי מצליח למצוא את התכונות והפעולות של כל מחלקה.

תוכן עניינים

[הוקרת תודה 2](#_Toc133922417)

[תוכן עניינים 3](#_Toc133922418)

[מבוא 5](#_Toc133922419)

[הרקע לפרויקט 5](#_Toc133922420)

[שם הפרויקט 5](#_Toc133922421)

[תיאור קצר של הפרויקט 5](#_Toc133922422)

[קהל היעד 5](#_Toc133922423)

[הסיבות לבחירת הנושא 5](#_Toc133922424)

[מטרות המערכת 6](#_Toc133922425)

[מטרת על 6](#_Toc133922426)

[מטרות נלוות 6](#_Toc133922427)

[תיאור המערכת 6](#_Toc133922428)

[גבולות המערכת של כל לקוח 6](#_Toc133922429)

[סביבת/סביבות פיתוח 6](#_Toc133922430)

[שפת/שפות תכנות 6](#_Toc133922431)

[שכבות 7](#_Toc133922432)

[שכבת גישה לנתונים (DAL – Data access layer): 7](#_Toc133922433)

[שכבת ההיגיון העסקי (BLL – Business logic layer): 7](#_Toc133922434)

[שכבת ממשק משתמש (UI): 7](#_Toc133922435)

[שכבת שירות הרשת (WS): 8](#_Toc133922436)

[פלטפורמות של הלקוח /ות, כולל הסבר למה נבחרה כל פלטפורמה 9](#_Toc133922437)

[אתגרים מרכזיים 9](#_Toc133922438)

[חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים בפרויקט 10](#_Toc133922439)

[בסיס הנתונים - DataBase 10](#_Toc133922440)

[פירוט הטבלאות 11](#_Toc133922441)

[UserTBL – טבלה השומרת את המידע על כל המשתמשים במערכת. 11](#_Toc133922442)

[11](#_Toc133922443)

[NotebookTBL – טבלה השומרת את המידע על כל המחברות במערכת. 12](#_Toc133922444)

[SubjectTBL – טבלה השומרת את הנושאים למחברות. 14](#_Toc133922445)

[SchoolTBL – טבלה השומרת את המידע על בתי הספר 14](#_Toc133922446)

[LikeTBL – טבלה השומרת את הלייקים למחברות 16](#_Toc133922447)

[CommentTBL – טבלה השומרת את התגובות למחברות 16](#_Toc133922448)

[UserInSchoolTBL – טבלה מקשרת הממפה בין משתמש לבית ספר מסויים 17](#_Toc133922449)

[תרשים DSD (קשרי גומלין) 18](#_Toc133922450)

[צד שרת 20](#_Toc133922451)

[המחלקות המטפלות בנתונים – Models 20](#_Toc133922452)

[Class Diagram 20](#_Toc133922453)

[קוד המחלקות 21](#_Toc133922454)

[שכבת שירותי רשת 28](#_Toc133922455)

[הטכנולוגיה 28](#_Toc133922456)

[שירותי רשת שהאפליקציה מספקת 29](#_Toc133922457)

[שירותי רשת נצרכים ממקור חיצוני 32](#_Toc133922458)

[שכבת Business Logic לוגיקה עסקית 37](#_Toc133922459)

[Class Diagram 37](#_Toc133922460)

[קוד המחלקות 38](#_Toc133922461)

[צד לקוח 70](#_Toc133922462)

[ממשקי משתמש/ת 70](#_Toc133922463)

[מפת אתר 70](#_Toc133922464)

[מדריך למשתמש/ת 71](#_Toc133922465)

[מדריך למשתמש רגיל 71](#_Toc133922466)

[מדריך למשתמש אדמין 80](#_Toc133922467)

[ביבליוגרפיה 87](#_Toc133922468)

[נספחים 88](#_Toc133922469)

[AdminPanel.aspx 88](#_Toc133922470)

[AdminPanel.aspx.cs 91](#_Toc133922471)

[CreateNotebook.aspx 96](#_Toc133922472)

[CreateNotebook.aspx.cs 98](#_Toc133922473)

[Explore.aspx 102](#_Toc133922474)

[Explore.aspx.cs 104](#_Toc133922475)

[Home.aspx 108](#_Toc133922476)

[Home.aspx.cs 110](#_Toc133922477)

[LikedNotebooks.aspx 112](#_Toc133922478)

[LikedNotebooks.aspx.cs 113](#_Toc133922479)

[Login.aspx 115](#_Toc133922480)

[Login.aspx.cs 116](#_Toc133922481)

[Register.aspx 117](#_Toc133922482)

[Register.aspx.cs 119](#_Toc133922483)

[UserInfo.aspx 121](#_Toc133922484)

[UserInfo.aspx.cs 124](#_Toc133922485)

[UserNotebooks.aspx 126](#_Toc133922486)

[UserNotebooks.aspx.cs 127](#_Toc133922487)

[ViewNotebook.aspx 129](#_Toc133922488)

[ViewNotebook.aspx.cs 135](#_Toc133922489)

[WebsiteStats.aspx 141](#_Toc133922490)

[WebsiteStats.aspx.cs 149](#_Toc133922491)

# מבוא

## הרקע לפרויקט

### שם הפרויקט

NoteShare

### תיאור קצר של הפרויקט

פלטפורמה לכתיבת ושיתוף מחברות בין סטודנטים ותלמידים.

### קהל היעד

המערכת מיועדת בעיקר לסטודנטים/תלמידים הרוצים למצוא ולכתוב סיכומים ומחברות בקורסים או מקצועות שונים. בנוסף, המערכת יכולה לשמש לכל סטודנט או כל משתמש לכתוב הערות לעצמו לנהל מחברת בנושאים שונים ולשמור סיכומים.

### הסיבות לבחירת הנושא

היות ואני לומד מכיתה ח' כסטודנט לתואר ראשון במדעי המחשב במקביל ללימודי בתיכון ובנוסף יש לי מספר תוכניות אקדמיות נוספות, היה לי צורך רב בכתיבה נרחבת של סיכומים וכמו כן היה צורך בחיפוש ברשת האינטרנט. מכיוון שנחשפתי לצורך הזה והבנתי שרבים זקוקים לפתרון, חשבתי על רעיון לפלטפורמה שתרכז את כל הצרכים הללו במקום אחד ותנגיש את הכתיבה והחיפוש של הסיכומים והמחברות.

## מטרות המערכת

### מטרת על

המטרה המרכזית של המערכת היא לספק כלי לסטודנטים ותלמידים לחפש לשתף ולכתוב בעצמם מחברות וסיכומים, עם אפשרות להגיב ולשמור.

### מטרות נלוות

מטרת משנה למערכת היא לספק לאנשים דרך לשמור מסמכים שלהם ולכתוב מסמכים.

## תיאור המערכת

הפונקציונליות העיקרית היא מערכת כתיבת ויצירת מחברות. ניתן לבחור בין להעלות מסמך או קובץ לשיתוף או להשתמש בכלי לכתיבת מחברות בתוך האתר. שבו ניתן לערוך את המחברת בתוך האתר בצורה נוחה וקלה. בנוסף שיתוף וחיפוש של המחברות בצורה יעילה ונוחה.

## גבולות המערכת של כל לקוח

המערכת מטפלת בשיתוף מחברות, עריכה online, הורדת והעלת קבצים, חיפוש ועוד..

המערכת אינה מטפלת ב2 אנשים הרוצים לערוך מחברת אחת.

סביבת/סביבות פיתוח  
Microsoft Visual Studio  
Microsoft SQL Server

שפת/שפות תכנות  
ASP.NET, C# וכן javascript, html, css, bootstrap, material UI, jquery.

שכבות

### שכבת גישה לנתונים (DAL – Data access layer):

שכבת הגישה לנתונים אחראית על ניהול התקשורת בין אפליקציית האינטרנט למסד הנתונים. מטרתו העיקרית היא לשלוף ולאחסן נתונים למסד הנתונים וממנו. הוא פועל כמתווך בין שכבת ההיגיון העסקי (BLL) לבין מסד הנתונים, ומסיר את המורכבות של מערכת אחסון הנתונים הבסיסית משכבת ההיגיון העסקי. שכבת הגישה לנתונים מספקת גם פונקציונליות נוספת המאפשרת שיפור ביצועים(Caching, connection pooling). בפרוייקט שלי מדובר במחלקות שבתקיית הmodels.

### שכבת ההיגיון העסקי (BLL – Business logic layer):

שכבת הלוגיקה העסקית אחראית על הטמעת הכללים העסקיים והתהליכים של אפליקציית האינטרנט. הוא מכיל את ההיגיון שקובע כיצד הנתונים מעובדים, מטופלים ומשנים. שכבת ההיגיון העסקי מפקידה את כללי האפליקציה, ומבטיחה שהם מופרדים משכבות הUI וגישה לנתונים. שכבה זו היא המקום שבו מתבצעים חישובים מורכבים, אלגוריתמים ואימותים. זה כולל גם את זרימות העבודה של האפליקציה. שכבת הלוגיקה העסקית אחראית לספק ממשק עקבי ואמין לשכבת הגישה לנתונים ולשכבת ממשק המשתמש. בפרוייקט שלי מדובר במחלקות שבתקיית הmodelsBL. חשוב לציין שהן מרחיבות (partial) את המחלקות בmodels ושם הפונקציונליות המרכזית של המערכת.

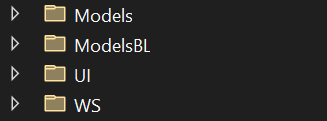
### שכבת ממשק משתמש (UI):

שכבת ממשק המשתמש (UI) היא שכבת המצגת של יישום האינטרנט. מטרתו העיקרית היא לספק ממשק ידידותי למשתמש למשתמשים לאינטראקציה עם האתר. שכבה זו מורכבת מ-HTML, CSS ו-JavaScript שיוצרים את דפי האינטרנט המוצגים למשתמש. שכבת ממשק המשתמש מטפלת באינטראקציות ואירועים של משתמשים, כגון לחיצות עכבר, קלט מקלדת והגשת טפסים. הוא מתקשר עם שכבת ההיגיון העסקי כדי לעדכן את הנתונים ולהגיב לאינטראקציות של משתמשים. שכבת ממשק המשתמש אחראית גם על מתן משוב למשתמש על הפעולות שעשה ועל התוצאות של אותן פעולות. אצלי בפרוייקט, זה נמצא תחת התיקייה UI ששם הweb forms של כל הדפים של האתר, ומחלקות הcode behind המתממשקות עם הmodelsBL.

לסיכום, שלוש השכבות הללו של פרויקט אינטרנט פועלות יחד כדי לספק יישום אינטרנט אמין, מהיר וידידותי למשתמש. שכבת הגישה לנתונים מנהלת אחסון ושליפת נתונים, שכבת הלוגיקה העסקית מיישמת את הכללים והתהליכים, ושכבת ממשק המשתמש מציגה את המידע למשתמש בצורה אינטואיטיבית ונגישה. הבנת תפקידה של כל שכבה היא קריטית לבניית אפליקציית אינטרנט מוצלחת.

### שכבת שירות הרשת (WS):

שכבה זאת אחראית לייצא שירות רשת מהאתר. כלומר אפשור של אפליקציות אחרות להשתמש בפונקציות המוגדרות בשכבה זו. בפרוייקט אנו השתמשנו בקובץ מסוג asmx.

מבנה התיקיות בפרוייקט:

פלטפורמות של הלקוח /ות, כולל הסבר למה נבחרה כל פלטפורמה  
פלטפורמת WEB – ASP.NET

על מנת שהמערכת תהיה נגישה מכל מקום דרך רשת האינטרנט, ללא צורך בהתקנות.

ASP.NET, שפותחה על ידי מיקרוסופט, היא פלטפורמה חזקה הידועה באינטגרציה החלקה שלה עם טכנולוגיות מיקרוסופט, ביצועים גבוהים ותמיכה חזקה. כלי הפיתוח המהירים שלה, תכונות האבטחה המובנות, הארכיטקטורה המודולרית והרבגוניות הופכים אותה לבחירה טובה עבור מפתחים הרוצים להנגיש מערכת.

## אתגרים מרכזיים

בעיה מרכזית שהייתה לי היא בניית המערכת של עריכת המחברות online ושמירתם בצורה נוחה. בשביל העריכה עצמה, השתמשתי בגאווה סקריפט שמשתמש בתגים של html כדי לערוך את הטקסט. למשל, אם ארצה להגדיל טקסט זה יקיף אותו בתג <b>, ושמירת הhtml במסד הנתונים על מנת לשמור את המחברת.

בחנתי כמה פתרונות שונים שעלו בנוגע לשמירת המחברת בינהן, להשתמש בפקד asp.net של טקסט, כדי לשלוח לשרת את הנתונים. או שליחת הנתונים דרך גאווה סקריפט עם jquery. לבסוף החלטתי להשתמש בגאווה סקריפט וכך אני ניתן את המחברת מבלי לטעון את הדף.

בעיה נוספת היא לתת גישה למידע למשתמשי מנהל, החלטתי ליצור מן מערכת שתאפשר לאדמין להתחבר בתור משתמש רגיל ובכך לבצע את כל הפעולות שלו ולראות את המסכים שלו.

Graphical user interface, text, application, email

Description automatically generatedניתן לראות זו בקוד שלהלן אשר פתר את הבעיה:

שליחת תוכן המחברת לBehind Code לאחר לחיצה על כפתור "Save"

Graphical user interface, text, application, email

Description automatically generated

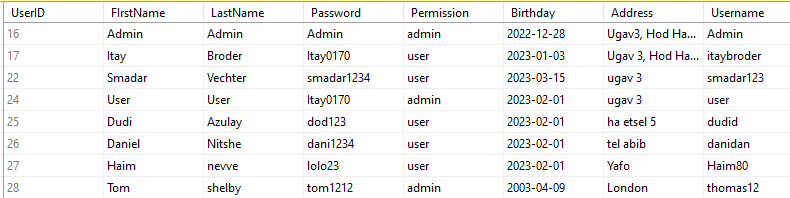
טיפול הבקשה בBehind Code ושמירת תוכן המחברת במסד הנתונים

## חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים בפרויקט

# Graphical user interface, text, application, email Description automatically generatedבסיס הנתונים - DataBase

## פירוט הטבלאות

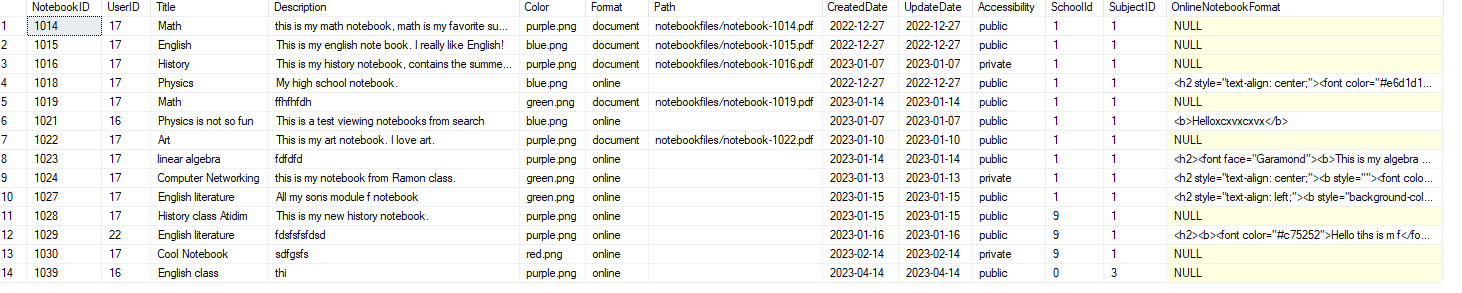
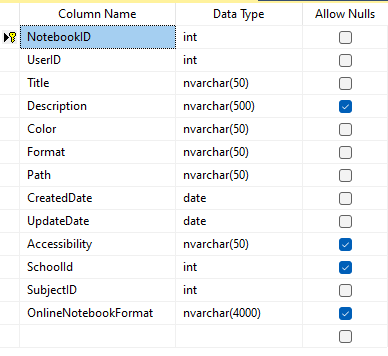
### UserTBL – טבלה השומרת את המידע על כל המשתמשים במערכת.

* UserID – מזהה יחודי למשתמש
* FirstName – השם הפרטי של המשתמש
* LastName – השם משפחה של המשתמש
* Password - הסיסמה של המשתמש
* Permission – הרשאות הגישה של המשתמש, זה הסוגי משתמשים היכולים להיות. כרגע admin להרשאת מנהל וuser להרשאת גישה רגילה.
* Birthday – תאריך לידה של המשתמש
* Address – כתובת המגורים\עיר של המשתמש
* Username – מזהה המשתמש, יוצג כשמו לשאר המשתמשים מערכת.

### 

### NotebookTBL – טבלה השומרת את המידע על כל המחברות במערכת.

* NotebookID – מזהה ייחודי למחברת
* UserID – המשתמש ששייכת לו המחברת
* Title – כותרת למחברת
* Description - תיאור במילים של המחברת
* Color – צבע המחברת
* Format – פורמט המחברת, יכול להיות document, מחברת שמחזיקה קובץ או online שזו מחברת שניתנת לעריכה באתר
* Path – במקרה והמחברת מסוג document, שדה זה מכיל את מיקום הקובץ בשרת
* CreatedDate – התאריך בו נוצרה המחברת
* UpdateDate – התאריך האחרון בו עודכנה המחברת
* Accessibility – הרשאות גישה של המחברת, יכול להיות public במצב זה המחברת זמינה לכל משתמש, או private במצב זה המחברת זמינה אך ורק למשתמש שיצר אותה ולadmins.
* ScoolID – המספר המזהה של בית הספר אליו משוייכת המחברת
* SubjectID – המספר המזהה של הנושא אליו משוייכת המחברת
* OnlineNotebookFormat – במידה והמחברת בפורמט online, שדה זה מכיל את הנתונים על המחברת. מחרוזת של html המתארת את המחברת

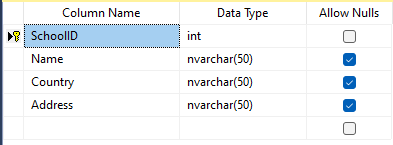


### SubjectTBL – טבלה השומרת את הנושאים למחברות.

* SubjectID - מזהה יחודי לכל נושא.
* Name – שם הנושא. למשל Math, English.

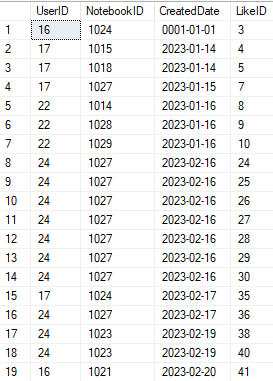
### SchoolTBL – טבלה השומרת את המידע על בתי הספר

* SchoolID – מזהה ייחודי לכל ביתספר
* Name – שם בית הספר
* Country – המדינה שבה בית הספר
* Address – כתובת של בית הספר



### LikeTBL – טבלה השומרת את הלייקים למחברות

* UserID – מזהה ייחודי של המשתמש שעשה את הלייק
* NotebookID – מזהה ייחודי של המחברת שעשו לה לייק
* CreatedDate – התאריך בו נעשה הלייק
* LikeID – מזהה ייחודי לכל לייק



### CommentTBL – טבלה השומרת את התגובות למחברות

* CommentID – מזהה ייחודי לתגובה
* UserID – מזהה המשתמש שהגיב
* NotebookID – מזהה המחברת שעליה התגובה
* Comment – תוכן התגובה
* CreatedDate – תאריך יצירת התגובה

### UserInSchoolTBL – טבלה מקשרת הממפה בין משתמש לבית ספר מסויים

* UserID - מזהה המשתמש
* SchoolID – מזהה בית הספר

## תרשים DSD (קשרי גומלין)

# צד שרת

## המחלקות המטפלות בנתונים – Models

### Class Diagram

### קוד המחלקות

#### UserTbl – מחלקה האחראית על משתמשים במסד הנתונים

/// <summary>

/// A class that represents a user in the database. User's basic information.

/// </summary>

public partial class UserTbl

{

public UserTbl()

{

CommentTbl = new HashSet<CommentTbl>();

LikeTbl = new HashSet<LikeTbl>();

UserInSchoolTbl = new HashSet<UserInSchoolTbl>();

}

public UserTbl(int userId, string firstName, string lastName,

string password, string permission, DateTime birthday, string

address, string username)

{

UserId = userId;

FirstName = firstName;

LastName = lastName;

Password = password;

Permission = permission;

Birthday = birthday;

Address = address;

Username = username;

}

public UserTbl(int userId)

{

UserTbl user = UserTbl.GetUserByUserId(userId);

UserId = user.UserId;

FirstName = user.FirstName;

LastName = user.LastName;

Password = user.Password;

Permission = user.Permission;

Birthday = user.Birthday;

Address = user.Address;

Username = user.Username;

}

public int UserId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Password { get; set; }

public string Permission { get; set; }

public DateTime Birthday { get; set; }

public string Address { get; set; }

public string Username { get; set; }

public virtual ICollection<CommentTbl> CommentTbl { get; set; }

public virtual ICollection<LikeTbl> LikeTbl { get; set; }

public virtual ICollection<UserInSchoolTbl> UserInSchoolTbl

{ get; set; }

}

#### NotebookTBL – מחלקה האחראית על ניהול המחברות

/// <summary>

/// A class that represents the NotebookTbl table in the database.

/// </summary>

public partial class NotebookTbl

{

public int NotebookId { get; set; }

public int UserId { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string Color { get; set; }

public string Format { get; set; }

public string Path { get; set; }

public DateTime CreatedDate { get; set; }

public DateTime UpdateDate { get; set; }

public string Accessibility { get; set; }

public int? SchoolId { get; set; }

public int SubjectId { get; set; }

public string OnlineNotebookFormat { get; set; }

public virtual SubjectTbl Subject { get; set; }

#### }

#### SubjectTBL - מחלקה האחראית על ניהול הנושאים במסד הנתונים

/// <summary>

/// A class that represents a subject in the database.

/// </summary>

public partial class SubjectTbl

{

public SubjectTbl()

{

NotebookTbl = new HashSet<NotebookTbl>();

}

public SubjectTbl(int subjectId)

{

NoteShareContext context = new NoteShareContext();

SubjectTbl subject = context.SubjectTbl.ToList().Find((SubjectTbl a) => a.SubjectId == subjectId);

this.SubjectId = subjectId;

this.Name = subject.Name;

}

public int SubjectId { get; set; }

public string Name { get; set; }

public virtual ICollection<NotebookTbl> NotebookTbl { get; set; }

}

#### CommentTBL – מחלקה האחראית לתגובות על מחברת

/// <summary>

/// This class represents the CommentTbl table in the database.

/// </summary>

public partial class CommentTbl

{

public int UserId { get; set; }

public int NotebookId { get; set; }

public string Comment { get; set; }

public DateTime CreatedDate { get; set; }

public int CommentId { get; set; }

public virtual UserTbl User { get; set; }

}

#### LikeTBL – מחלקה השומרת מידע על הלייקים של מחברות

/// <summary>

/// This class represents the LikeTbl table in the database.

/// </summary>

public partial class LikeTbl

{

public int UserId { get; set; }

public int NotebookId { get; set; }

public DateTime CreatedDate { get; set; }

public int LikeId { get; set; }

public virtual UserTbl User { get; set; }

#### }

#### SchoolTBL – טבלה המייצגת בית ספר במסד הנתונים

/// <summary>

/// This class represents the SchoolTbl table in the database.

/// </summary>

public partial class SchoolTbl

{

public SchoolTbl()

{

UserInSchoolTbl = new HashSet<UserInSchoolTbl>();

}

public int SchoolId { get; set; }

public string Name { get; set; }

public string Country { get; set; }

public string Address { get; set; }

public virtual ICollection<UserInSchoolTbl> UserInSchoolTbl { get; set; }

}

#### UserInSchoolTBL – מחלקה המייצגת משתמש בבית הספר

/// <summary>

/// This class represents the UserInSchoolTbl table in the database.

/// </summary>

public partial class UserInSchoolTbl

{

public int UserId { get; set; }

public int SchoolId { get; set; }

public virtual SchoolTbl School { get; set; }

public virtual UserTbl User { get; set; }

}

## שכבת שירותי רשת

### הטכנולוגיה

Simple Object Access Protocol (SOAP) הוא פרוטוקול העברת הודעות המשמש להחלפת נתונים מובנים בין שירותי רשת. זהו פרוטוקול סטנדרטי המגדיר את מבנה ההודעות, כולל פורמט ההודעה, קשירת הפרוטוקול ופרוטוקול ההעברה הבסיסי. ב-.NET Framework, ניתן ליצור שירותי רשת באמצעות קבצי, ASMX המכילים את הטמעת פונקציות ושיטות בשירות הרשת. לאחר מכן, לקוחות יכולים לקרוא לשיטות אלה באמצעות הודעות . SOAP קובץ ASMX פועל מתווך להודעות SOAP נכנסות, ו-.NET Framework מספק מחסנית פרוטוקול SOAP המטפלת בניתוח ובעיבוד של הודעות. SOAP פרוטוקול SOAP מספק דרך לשירותי רשת להיות עצמאיים בפלטפורמה ובלתי תלויים בשפה, מה שהופך אותו לבחירה פופולרית לבניית מערכות אינטרנט.

### שירותי רשת שהאפליקציה מספקת

#### קהל היעד

את הפעולות הנ"ל יצרוך האתר של התיכון (מיני-פרויקט). כך, בכניסה לאתר יוכל הגולש לראות מידע אודות התלמידים הלומדים בתיכון. כמו כן, יוכל הגולש לקבל מידע נוסף על המחברות של אותו התלמיד.

#### הפעולות

public class NoteShareWS : System.Web.Services.WebService

{

/// <summary>

/// Returns all the notebooks.

/// </summary>

/// <returns></returns>

[WebMethod]

public List<NotebookWS> GetAllNotebooks()

{

List<NotebookTbl> notebooks = NotebookTbl.getAllNotebooks();

return notebooks.Select(x=> new NotebookWS(x)).ToList();

}

/// <summary>

/// Returns a notebook by id.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[WebMethod]

public NotebookWS GetNotebookById(int id)

{

NotebookWS notebook = new NotebookWS(NotebookTbl.GetNotebookByNotebookID(id));

return notebook;

}

/// <summary>

/// returns the user that owns the notebook.

/// </summary>

/// <param name="notebook"></param>

/// <returns></returns>

[WebMethod]

public UserWS GetUserFromNotebook(NotebookWS notebook)

{

return new UserWS(UserTbl.GetUserByUserId(notebook.UserId));

}

/// <summary>

/// Likes a notebook.

/// </summary>

/// <param name="notebook"></param>

[WebMethod]

public void LikeNotebook(NotebookWS notebook)

{

//24 is a user which allows anonymos likes and comments.

NotebookTbl notebookTbl = new NotebookTbl(notebook);

notebookTbl.Like(new UserTbl(24));

}

/// <summary>

/// Unlikes a notebook.

/// </summary>

/// <param name="notebook"></param>

[WebMethod]

public void UnLikeNotebook(NotebookWS notebook)

{

//24 is a user which allows anonymos likes and comments.

NotebookTbl notebookTbl = new NotebookTbl(notebook);

List<LikeTbl> likes = notebookTbl.GetAllLikes();

if(likes.Count > 0)

{

LikeTbl.UnLikeByLikeId(likes.First().LikeId);

}

}

/// <summary>

/// Adds a comment to a notebook.

/// </summary>

/// <param name="comment"></param>

/// <param name="notebook"></param>

[WebMethod]

public void CommentOnNotebook(string comment, NotebookWS notebook)

{

//24 is a user which allows anonymos likes and comments.

NotebookTbl notebookTbl = new NotebookTbl(notebook);

notebookTbl.AddComment(new UserTbl(24), comment);

}

/// <summary>

/// Returns all the comments of a notebook.

/// </summary>

/// <param name="notebook"></param>

/// <returns></returns>

[WebMethod]

public List<CommentWS> GetCommentsOfNotebook(NotebookWS notebook)

{

List<CommentTbl> comments = CommentTbl.GetCommentsByNotebookId(notebook.NotebookId);

return comments.Select(x => new CommentWS(x)).ToList();

}

/// <summary>

/// returns a subject by a subject Id.

/// </summary>

/// <param name="subjectId"></param>

/// <returns></returns>

[WebMethod]

public string GetSubjectById(int subjectId)

{

return SubjectTbl.GetSubjectByID(subjectId).Name;

}

/// <summary>

/// Returns the like count of a notebook.

/// </summary>

/// <param name="notebook"></param>

/// <returns></returns>

[WebMethod]

public int GetLikeCount(NotebookWS notebook)

{

NotebookTbl notebookTbl = new NotebookTbl(notebook);

return notebookTbl.GetLikeCount();

}

}

### שירותי רשת נצרכים ממקור חיצוני

#### מחלקת Notebooks

##### צד שרת של המחלקה – אחראי על הצגת המחברות באתר תיכון רמון

public partial class Notebooks : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

NoteShareWS context = new NoteShareWS();

NotebookWS[] notebooksArray = context.GetAllNotebooks();

List<NotebookWS> notebooks = new List<NotebookWS>();

foreach (NotebookWS notebook in notebooksArray)

{

notebooks.Add(notebook);

}

if (!IsPostBack)

{

//show only public notebooks for regular users.

NotebooksDataList.DataSource = notebooks.FindAll(x => x.Accessibility == "public" && x.Format == "online" && x.SchoolId == 1);

NotebooksDataList.DataBind();

}

}

protected void NotebooksDataList\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument);

}

}

}

##### A picture containing graphical user interface Description automatically generatedהUI של המחלקה בשירות ברשת

#### מחלקת ViewNotebooks

##### צד שרת של המחלקה – אחראי על הצגת המחברת באתר תיכון רמון

public partial class ViewNotebook : System.Web.UI.Page

{

public static NotebookWS notebook;

protected void Page\_Load(object sender, EventArgs e)

{

NoteShareWS context = new NoteShareWS();

if (!IsPostBack)

{

string desiredValue = "";

foreach (string item in HttpContext.Current.Request.Url.Query.Split('&'))

{

string[] parts = item.Replace("?", "").Split('=');

if (parts[0] == "id")

{

desiredValue = parts[1];

break;

}

}

//update mode

if (desiredValue != "")

{

notebook = context.GetNotebookById((Int32.Parse(desiredValue)));

NoteTitle.Text = notebook.Title;

DescriptionLabel.Text = notebook.Description;

DateLBL.Text = notebook.UpdateDate.ToString("dd/MM/yyyy");

CantEditDiv.InnerHtml = notebook.OnlineNotebookFormat;

NotebookSubject.Text = context.GetSubjectById(notebook.SubjectId);

UserLinkl.Text = context.GetUserFromNotebook(notebook).Username;

CommentWS[] commentsArray = context.GetCommentsOfNotebook(notebook);

List<CommentWS> comments = new List<CommentWS>();

foreach (CommentWS comment in commentsArray)

{

Comments.Add(comment);

}

comments.Reverse();

CommentsDataList.DataSource = comments;

CommentsDataList.DataBind();

}

}

LikeCountLabel.Text = "" + context.GetLikeCount(notebook);

}

protected void LikeButton\_Click(object sender, ImageClickEventArgse)

{

NoteShareWS context = new NoteShareWS();

if (LikeButton.ImageUrl == "./Assets/LikeBefore.png")

{

//like

context.LikeNotebook(notebook);

LikeButton.ImageUrl = "./Assets/LikeAfter.png";

}

else

{

//unlike

context.UnLikeNotebook(notebook);

LikeButton.ImageUrl = "./Assets/LikeBefore.png";

}

LikeCountLabel.Text = "" + context.GetLikeCount(notebook);

}

protected void AddCommentBTN\_Click(object sender, EventArgs e)

{

NoteShareWS context = new NoteShareWS();

string comment = CommentTextBox.Text;

context.CommentOnNotebook(comment, notebook);

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

}

##### הUI של עמוד המחברת בשירות הרשת

## שכבת Business Logic לוגיקה עסקית

### Class Diagram

### קוד המחלקות

#### UserTbl – מחלקה שאחראית על המשתמש במערכת ועל הפעולות שיכול לבצע

/// <summary>

/// A class that represents a user in the database.

/// </summary>

public partial class UserTbl

{

/// <summary>

/// Login a user with username and password. Then returns the loged in user.

/// </summary>

/// <param name="username"></param>

/// <param name="pass"></param>

/// <returns></returns>

public static UserTbl UserLogin(string username, string pass)

{

NoteShareContext db = new NoteShareContext();

try

{

return db.UserTbl.ToList().Find(x => x.Username == username && x.Password == pass);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

return null;

}

/// <summary>

/// Registers a user. Returning the new user.

/// </summary>

/// <param name="username"></param>

/// <param name="password"></param>

/// <param name="firstName"></param>

/// <param name="LastName"></param>

/// <param name="birthday"></param>

/// <param name="schools"></param>

/// <param name="address"></param>

/// <returns></returns>

public static UserTbl Register(string username, string password, string firstName, string LastName,  
 DateTime birthday, List<SchoolTbl> schools, string address)

{

NoteShareContext db = new NoteShareContext();

if (GetUserByUsername(username) == null)

{

UserTbl user = new UserTbl();

{

user.Username = username;

user.Password = password;

user.FirstName = firstName;

user.LastName = LastName;

user.Birthday = birthday;

user.Address = address;

user.Permission = "user";

}

db.Entry(user).State = Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

foreach (SchoolTbl school in schools)

{

UserInSchoolTbl userInSchool = new UserInSchoolTbl();

{

userInSchool.UserId = user.UserId;

userInSchool.SchoolId = school.SchoolId;

}

db.Entry(userInSchool).State = Microsoft.EntityFrameworkCore.EntityState.Added;

}

db.SaveChanges();

return user;

}

return null;

}

/// <summary>

/// Updates user with properties.

/// </summary>

/// <param name="permission"></param>

/// <param name="username"></param>

/// <param name="password"></param>

/// <param name="firstName"></param>

/// <param name="LastName"></param>

/// <param name="birthday"></param>

/// <param name="schools"></param>

/// <param name="address"></param>

public void UpdateUser(string permission, string username, string password, string firstName, string LastName,  
 DateTime birthday, List<SchoolTbl> schools, string address)

{

NoteShareContext db = new NoteShareContext();

if (GetUserByUsername(username) == null || GetUserByUsername(username).Username == username)

{

UserTbl user = new UserTbl();

{

user.UserId = this.UserId;

user.Username = username;

user.Password = password;

user.FirstName = firstName;

user.LastName = LastName;

user.Birthday = birthday;

user.Permission = permission;

user.Address = address;

}

db.Update(user).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

foreach (SchoolTbl school in schools)

{

UserInSchoolTbl userInSchool = new UserInSchoolTbl();

{

userInSchool.UserId = user.UserId;

userInSchool.SchoolId = school.SchoolId;

}

db.Update(userInSchool).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

}

db.SaveChanges();

}

}

/// <summary>

/// Returns a user by username.

/// </summary>

/// <param name="username"></param>

/// <returns></returns>

public static UserTbl GetUserByUsername(string username)

{

NoteShareContext db = new NoteShareContext();

return db.UserTbl.ToList().Find(x => x.Username == username);

}

/// <summary>

/// Returns a user by userId

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static UserTbl GetUserByUserId(int id)

{

NoteShareContext db = new NoteShareContext();

return db.UserTbl.ToList().Find(x => x.UserId == id);

}

/// <summary>

/// Returns a list of all the users.

/// </summary>

/// <returns></returns>

public static List<UserTbl> GetAllUsers()

{

NoteShareContext db = new NoteShareContext();

return db.UserTbl.ToList();

}

/// <summary>

/// Deletes all Comments from the notebook.

/// </summary>

public void DeleteAllComments()

{

NoteShareContext db = new NoteShareContext();

foreach (CommentTbl comment in db.CommentTbl.ToList().FindAll(x => x.UserId == this.UserId))

{

NoteShare.Models.CommentTbl.DeleteComment(comment.CommentId);

}

}

/// <summary>

/// Deletes All the notebooks that the user of this notebooks wrote.

/// </summary>

public void DeleteAllNotebooks()

{

foreach (NotebookTbl notebook in NotebookTbl.GetNotebooksByUserID(this.UserId))

{

notebook.DeleteNotebook();

}

}

/// <summary>

/// Deletes a user from the database.

/// </summary>

public void Delete()

{

DeleteAllLikes();

DeleteAllComments();

DeleteAllNotebooks();

NoteShareContext db = new NoteShareContext();

NoteShare.Models.UserInSchoolTbl.DeleteUserInSchool(this.UserId);

UserTbl user = new UserTbl();

{

user.UserId = this.UserId;

}

db.Remove(user).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Deletes all likes from user.

/// </summary>

public void DeleteAllLikes()

{

NoteShareContext db = new NoteShareContext();

foreach (LikeTbl like in db.LikeTbl.ToList().FindAll(x => x.UserId == this.UserId))

{

db.Remove(like).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

}

/// <summary>

/// Gets all the liked notebooks of the user.

/// </summary>

/// <returns></returns>

public List<NotebookTbl> GetLikedNotebooks()

{

NoteShareContext db = new NoteShareContext();

List<LikeTbl> likes = db.LikeTbl.ToList().FindAll(x => x.UserId == this.UserId);

List<NotebookTbl> notebooks = new List<NotebookTbl>();

foreach (LikeTbl like in likes)

{

notebooks.Add(NotebookTbl.GetNotebookByNotebookID(like.NotebookId));

}

return notebooks;

}

}

#### NotebookTBL – מחלקה האחראית על ניהול המחברות במערכת

/// <summary>

/// A class that represents the NotebookTbl table in the database.

/// </summary>

public partial class NotebookTbl

{

// a database context object.

public static NoteShareContext db = new NoteShareContext();

/// <summary>

/// a copy constructor that copys NotebookWS to NotebookTBl object.

/// </summary>

/// <param name="notebook"></param>

public NotebookTbl(NotebookWS notebook)

{

this.NotebookId = notebook.NotebookId;

this.UserId = notebook.UserId;

this.Title = notebook.Title;

this.Description = notebook.Description;

this.Color = notebook.Color;

this.Format = notebook.Format;

this.Path = notebook.Path;

this.CreatedDate = notebook.CreatedDate;

this.UpdateDate = notebook.UpdateDate;

this.Accessibility = notebook.Accessibility;

this.SchoolId = notebook.SchoolId;

this.SubjectId = notebook.SubjectId;

this.OnlineNotebookFormat = notebook.OnlineNotebookFormat;

this.SubjectId = notebook.SubjectId;

}

public NotebookTbl()

{

}

/// <summary>

/// Creates a notebook with all the properties, then returns it.

/// </summary>

/// <param name="userId"></param>

/// <param name="title"></param>

/// <param name="description"></param>

/// <param name="color"></param>

/// <param name="format"></param>

/// <param name="path"></param>

/// <param name="accessibility"></param>

/// <param name="schoolId"></param>

/// <param name="subject"></param>

/// <returns></returns>

public static NotebookTbl CreateNewNotebook(int userId, string title, string description, string color,

string format, string path, string accessibility, int schoolId, string subject)

{

NoteShareContext db = new NoteShareContext();

NotebookTbl notebook = new NotebookTbl();

{

notebook.UserId = userId;

notebook.Title = title;

notebook.Description = description;

notebook.Color = color;

notebook.Format = format;

notebook.Path = path;

notebook.UpdateDate = DateTime.Now;

notebook.CreatedDate = DateTime.Now;

notebook.Accessibility = accessibility;

notebook.SchoolId = schoolId;

notebook.SubjectId = SubjectTbl.GetSubjectByName(subject).SubjectId;

}

db.Entry(notebook).State = Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

return notebook;

}

/// <summary>

/// Updates Notebook.

/// </summary>

/// <param name="notebookId"></param>

/// <param name="userId"></param>

/// <param name="title"></param>

/// <param name="description"></param>

/// <param name="color"></param>

/// <param name="format"></param>

/// <param name="path"></param>

/// <param name="accessibility"></param>

/// <param name="schoolId"></param>

/// <param name="subject"></param>

/// <returns></returns>

public static NotebookTbl UpdateNotebook(int notebookId, int userId, string title, string description, string color,

string format, string path, string accessibility, int schoolId, string subject)

{

NotebookTbl notebook = GetNotebookByNotebookID(notebookId);

notebook.UserId = userId;

notebook.Title = title;

notebook.Description = description;

notebook.Color = color;

notebook.Format = format;

notebook.Path = path;

notebook.UpdateDate = DateTime.Now;

notebook.Accessibility = accessibility;

notebook.SchoolId = schoolId;

notebook.SubjectId = SubjectTbl.GetSubjectByName(subject).SubjectId;

db.Update(notebook).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

return notebook;

}

/// <summary>

/// Returns a list of notebooks by Description.

/// </summary>

/// <param name="text"></param>

/// <returns></returns>

public static List<NotebookTbl> getNotebooksByDescription(string text)

{

return db.NotebookTbl.ToList().FindAll(x => x.Description.Contains(text));

}

/// <summary>

/// Returns a list of notebooks by a certain school.

/// </summary>

/// <param name="SchoolId"></param>

/// <returns></returns>

public static List<NotebookTbl> getNotebooksFromSchoolId(int SchoolId)

{

NoteShareContext db = new NoteShareContext();

return db.NotebookTbl.ToList().FindAll(x => x.SchoolId == SchoolId);

}

/// <summary>

/// Returns a list of all the notebooks.

/// </summary>

/// <returns></returns>

public static List<NotebookTbl> getAllNotebooks()

{

NoteShareContext db = new NoteShareContext();

return db.NotebookTbl.ToList();

}

/// <summary>

/// Returns a notebook selected from an existing list and variable.

/// </summary>

/// <param name="list"></param>

/// <param name="index"></param>

/// <param name="var"></param>

/// <returns></returns>

public static List<NotebookTbl> getNotebooksFromListAndVar(List<NotebookTbl> list, int index, string var)

{

List<NotebookTbl> notebooks = new List<NotebookTbl>();

if (index == 0)//אם מחפשים לפי שם

{

foreach (NotebookTbl notebook in list)

{

if (UserTbl.GetUserByUserId(notebook.UserId).Username == var)

notebooks.Add(notebook);

}

}

if (index == 1)//אם מחפשים לפי מקצוע

{

foreach (NotebookTbl notebook in list)

{

if (SubjectTbl.GetSubjectByID(notebook.SubjectId).Name == var)

notebooks.Add(notebook);

}

}

if (index == 2)//אם מחפשים לפי בית ספר

{

foreach (NotebookTbl notebook in list)

{

if (SchoolTbl.GetSchoolByID((int)notebook.SchoolId) != null &&

SchoolTbl.GetSchoolByID((int)notebook.SchoolId).Name == var)

notebooks.Add(notebook);

}

}

if (index == 3)//אם מחפשים לפי תיאור

{

foreach (NotebookTbl notebook in list)

{

if (notebook.Description.Contains(var))

notebooks.Add(notebook);

}

}

return notebooks;

}

/// <summary>

/// Updates the path property of a notebook.

/// </summary>

/// <param name="notebook"></param>

/// <param name="path"></param>

/// <returns></returns>

public static NotebookTbl UpdatePath(NotebookTbl notebook, string path)

{

notebook.Path = path;

db.Update(notebook).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

return notebook;

}

/// <summary>

/// Returns a notebook list of a certain user.

/// </summary>

/// <param name="userId"></param>

/// <returns></returns>

public static List<NotebookTbl> GetNotebooksByUserID(int userId)

{

return db.NotebookTbl.ToList().FindAll(x => x.UserId == userId);

}

/// <summary>

/// Returns a notebook list from a certain school.

/// </summary>

/// <param name="schoolName"></param>

/// <returns></returns>

public static List<NotebookTbl> GetNotebooksBySchool(string schoolName)

{

SchoolTbl school = SchoolTbl.GetSchoolByName(schoolName);

return db.NotebookTbl.ToList().FindAll(x => x.SchoolId == school.SchoolId);

}

/// <summary>

/// Returns a list of notebooks by a certain subject.

/// </summary>

/// <param name="subject"></param>

/// <returns></returns>

public static List<NotebookTbl> GetNotebooksBySubject(string subject)

{

return db.NotebookTbl.ToList().FindAll(x => SubjectTbl.GetSubjectByID(x.SubjectId).Name == subject);

}

/// <summary>

/// Returns a list of notebooks by username.

/// </summary>

/// <param name="username"></param>

/// <returns></returns>

public static List<NotebookTbl> GetNotebooksByUsername(string username)

{

if (UserTbl.GetUserByUsername(username) == null)

{

return null;

}

return db.NotebookTbl.ToList().FindAll(x => x.UserId == UserTbl.GetUserByUsername(username).UserId);

}

/// <summary>

/// Returns all the public notebooks by username.

/// </summary>

/// <param name="username"></param>

/// <returns></returns>

public static List<NotebookTbl> GetPublicNotebooksByUsername(string username)

{

return GetNotebooksByUsername(username).FindAll(x => x.Accessibility == "public");

}

/// <summary>

/// Returns a list of notebooks by a certain subject and username.

/// </summary>

/// <param name="notebookID"></param>

/// <returns></returns>

public static NotebookTbl GetNotebookByNotebookID(int notebookID)

{

return db.NotebookTbl.ToList().Find(x => x.NotebookId == notebookID);

}

/// <summary>

/// Returns a list of notebooks by a certain subject and username.

/// </summary>

public void DeleteNotebook()

{

DeleteAllLikes();

NoteShareContext db = new NoteShareContext();

db.Remove(this).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Update this noteook's html.

/// </summary>

/// <param name="notebookHtmlText"></param>

/// <param name="notebook"></param>

/// <returns></returns>

public static NotebookTbl UpdateNotebookHtml(string notebookHtmlText, NotebookTbl notebook)

{

notebook.OnlineNotebookFormat = notebookHtmlText;

db.Update(notebook).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

return notebook;

}

/// <summary>

/// Returns the number of notebooks of a certain user.

/// </summary>

/// <param name="userId"></param>

/// <returns></returns>

public static int GetNumberOfNotebooksByUserId(int userId)

{

NoteShareContext db = new NoteShareContext();

return db.NotebookTbl.ToList().FindAll(x => x.UserId == userId).Count;

}

/// <summary>

/// Checks if the notebook is owend by a user.

/// </summary>

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

/// <param name="notebook"></param>

/// <returns></returns>

public static bool IsHisNotebook(UserTbl user, NotebookTbl notebook)

{

if (NotebookTbl.GetNotebooksByUserID(user.UserId).Contains(notebook))

{

return true;

}

return false;

}

/// <summary>

/// adds a comment to a notebook.

/// </summary>

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

/// <param name="comment"></param>

public void AddComment(UserTbl user, string comment)

{

NoteShareContext db = new NoteShareContext();

CommentTbl commentTbl = new CommentTbl();

{

commentTbl.NotebookId = this.NotebookId;

commentTbl.UserId = user.UserId;

commentTbl.Comment = comment;

commentTbl.CreatedDate = DateTime.Now;

}

db.Entry(commentTbl).State = Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

}

/// <summary>

/// likes a notebook, adds a like to the database.

/// </summary>

/// <param name="user">this is the givven user</param>

public void Like(UserTbl user)

{

NoteShareContext db = new NoteShareContext();

LikeTbl like = new LikeTbl();

{

like.UserId = user.UserId;

like.NotebookId = this.NotebookId;

like.CreatedDate = DateTime.Now;

}

db.Entry(like).State = Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

}

/// <summary>

/// Unlike a notebook, removes a like from the database.

/// </summary>

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

public void UnLike(UserTbl user)

{

NoteShareContext db = new NoteShareContext();

LikeTbl like = LikeTbl.GetAllLikes().FindAll(x => x.UserId == user.UserId && x.NotebookId ==

this.NotebookId).First();

db.Remove(like).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Returns the total number of likes of a notebook.

/// </summary>

/// <returns></returns>

public int GetLikeCount()

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList().FindAll(x => x.NotebookId == this.NotebookId).Count;

}

/// <summary>

/// Returns a list of all the likes of a notebook.

/// </summary>

/// <returns></returns>

public List<LikeTbl> GetAllLikes()

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList().FindAll(x => x.NotebookId == this.NotebookId);

}

/// <summary>

/// Deletes all likes of a notebook.

/// </summary>

public void DeleteAllLikes()

{

NoteShareContext db = new NoteShareContext();

foreach (LikeTbl like in db.LikeTbl.ToList())

{

if (like.NotebookId == this.NotebookId)

{

db.Remove(like).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

}

}

/// <summary>

/// Returns a list of all the likes of a notebook from a user.

/// </summary>

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

/// <returns></returns>

public List<LikeTbl> GetLikesFromUser(UserTbl user)

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList().FindAll(x => x.NotebookId == this.NotebookId && x.UserId == user.UserId);

}

/// <summary>

/// checks if a user has liked a notebook.

/// </summary>

/// <param name="userId"></param>

/// <returns></returns>

public bool IsLiked(int userId)

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList().Find(x => x.UserId == userId &&

x.NotebookId == this.NotebookId ) != null ? true :

false;

}

}

#### SubjectTbl – מחלקה שאראית על הנושאים הנלמדים שקיימים במערכת

/// <summary>

/// A class that represents a subject in the database.

/// </summary>

public partial class SubjectTbl

{

/// <summary>

/// using when no a notebook has no subject.

/// </summary>

public static SubjectTbl NONE\_SUBJECT = GetSubjectByName("None");

/// <summary>

/// Get subject by name.

/// </summary>

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

/// <returns></returns>

public static SubjectTbl GetSubjectByName(string name)

{

NoteShareContext context = new NoteShareContext();

return context.SubjectTbl.ToList().Find(x => x.Name == name);

}

/// <summary>

/// Get subject by id.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static SubjectTbl GetSubjectByID(int id)

{

NoteShareContext context = new NoteShareContext();

return context.SubjectTbl.ToList().Find(x => x.SubjectId ==

id);

}

/// <summary>

/// Delete subject.

/// </summary>

/// <param name="subjectId"></param>

public static void DeleteSubject(int subjectId)

{

NoteShareContext db = new NoteShareContext();

SubjectTbl subject = new SubjectTbl(subjectId);

NoteShare.Models.NotebookTbl.GetNotebooksBySubject(subject.Name).ForEach((NotebookTbl notebook) => {

notebook.Subject = NONE\_SUBJECT;

notebook.SubjectId= NONE\_SUBJECT.SubjectId;

db.Update(notebook).State =

Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

});

db.Remove(subject).State = Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Creating a subject, and adding it to the database.

/// </summary>

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

public static void CreateSubject(string name)

{

NoteShareContext db = new NoteShareContext();

SubjectTbl subject = new SubjectTbl();

{

subject.Name = name;

}

db.Entry(subject).State =

Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

}

/// <summary>

/// Get the number of subjects in the database.

/// </summary>

/// <returns></returns>

public static int GetNumberOfSubjects()

{

NoteShareContext noteShareContext = new NoteShareContext();

return noteShareContext.SubjectTbl.ToList().Count();

}

/// <summary>

/// Get all subjects in the database.

/// </summary>

/// <returns></returns>

public static List<SubjectTbl> GetSubjects()

{

NoteShareContext noteShareContext = new NoteShareContext();

return noteShareContext.SubjectTbl.ToList();

}

}

#### CommentTbl – מחלקה האחראית על התגובות למחברות שיש במערכת

/// <summary>

/// this class is an extention of the CommentTbl class created by the

///scaffold command from

/// the database. it has functions that interacts with the Comment

///table.

/// </summary>

public partial class CommentTbl

{

/// <summary>

/// Returns a list of comments by notebookID.

/// </summary>

/// <param name="notebookId"></param>

/// <returns></returns>

public static List<CommentTbl> GetCommentsByNotebookId(int

notebookId)

{

NoteShareContext db = new NoteShareContext();

return db.CommentTbl.ToList().FindAll(x => x.NotebookId ==

notebookId);

}

/// <summary>

/// A class function for deleting this comment.

/// </summary>

public void Delete()

{

DeleteComment(this.CommentId);

}

/// <summary>

/// Deletes comment by a commentId.

/// </summary>

/// <param name="CommentId"></param>

public static void DeleteComment(int CommentId)

{

NoteShareContext db = new NoteShareContext();

CommentTbl comment = new CommentTbl();

{

comment.CommentId = CommentId;

}

db.Remove(comment).State =

Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Returns a UserTbl object of the user that wrote a comment.

/// </summary>

/// <param name="commentID"></param>

/// <returns></returns>

public static UserTbl GetUserThatCommented(int commentID)

{

NoteShareContext db = new NoteShareContext();

return UserTbl.GetUserByUserId(db.CommentTbl.ToList().Find(x

=> x.CommentId == commentID).UserId);

}

}

#### LikeTbl – מחלקה האחראית על הלייקים של מחברות במערכת

/// <summary>

/// A class that represents the LikeTbl table in the database.

/// </summary>

public partial class LikeTbl

{

/// <summary>

/// Checks of a notebook is liked by userId and notebookId.

/// </summary>

/// <param name="userId"></param>

/// <param name="notebookId"></param>

/// <returns></returns>

public static bool IsLiked(int userId, int notebookId)

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList().Find(x => x.UserId == userId &&

x.NotebookId == notebookId) != null ? true : false;

}

/// <summary>

/// A class function for unliking this notebook.

/// </summary>

public void UnLike()

{

UnLikeByLikeId(this.LikeId);

}

/// <summary>

/// Removes a like from notebook by likeId.

/// </summary>

/// <param name="likeId"></param>

public static void UnLikeByLikeId(int likeId)

{

NoteShareContext db = new NoteShareContext();

LikeTbl like = new LikeTbl();

{

like.LikeId = likeId;

}

db.Remove(like).State =

Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

/// <summary>

/// Reutrns a list of all the likes in the LikeTbl.

/// </summary>

/// <returns></returns>

public static List<LikeTbl> GetAllLikes()

{

NoteShareContext db = new NoteShareContext();

return db.LikeTbl.ToList();

}

}

#### SchoolTbl – מחלקה האחראית על ניהול בתי הספר במערכת

/// <summary>

/// A class that represents the SchoolTbl table in the database.

/// </summary>

public partial class SchoolTbl

{

/// <summary>

/// Constructor for SchoolTbl.

/// </summary>

/// <param name="id"></param>

public SchoolTbl(int id)

{

UserInSchoolTbl = new HashSet<UserInSchoolTbl>();

}

/// <summary>

/// returns a list of all schools.

/// </summary>

/// <returns></returns>

public static List<SchoolTbl> GetSchools()

{

NoteShareContext db = new NoteShareContext();

return db.SchoolTbl.ToList();

}

/// <summary>

/// Get school by name.

/// </summary>

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

/// <returns></returns>

public static SchoolTbl GetSchoolByName(string name)

{

NoteShareContext db = new NoteShareContext();

return db.SchoolTbl.ToList().Find(x => x.Name == name);

}

/// <summary>

/// Returns a SchoolTbl object by id.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static SchoolTbl GetSchoolByID(int id)

{

NoteShareContext db = new NoteShareContext();

return db.SchoolTbl.ToList().Find(x => x.SchoolId == id);

}

/// <summary>

/// create a school in the database.

/// </summary>

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

/// <param name="country"></param>

/// <param name="address"></param>

/// <returns></returns>

public static SchoolTbl CreateSchool(string name, string country,

string address)

{

NoteShareContext db = new NoteShareContext();

SchoolTbl school = new SchoolTbl();

{

school.Name = name;

school.Address = address;

school.Country = country;

}

db.Entry(school).State =

Microsoft.EntityFrameworkCore.EntityState.Added;

db.SaveChanges();

return school;

}

/// <summary>

/// Delete school from database.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static SchoolTbl DeleteSchool(int id)

{

NoteShare.Models.UserInSchoolTbl.

DeleteUserInSchoolBySchool(id);

NoteShareContext db = new NoteShareContext();

SchoolTbl school = new SchoolTbl();

{

school.SchoolId = id;

}

db.Remove(school).State =

Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

return school;

}

/// <summary>

/// Edit school.

/// </summary>

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

/// <param name="country"></param>

/// <param name="address"></param>

/// <param name="schoolId"></param>

/// <returns></returns>

public static SchoolTbl EditSchool(string name, string country,

string address, int schoolId)

{

NoteShareContext db = new NoteShareContext();

SchoolTbl school = new SchoolTbl();

{

school.SchoolId = schoolId;

school.Name = name;

school.Address = address;

school.Country = country;

}

db.Update(school).State =

Microsoft.EntityFrameworkCore.EntityState.Modified;

db.SaveChanges();

return school;

}

/// <summary>

/// A function that deletes this school.

/// </summary>

public void Delete()

{

DeleteSchool(this.SchoolId);

}

/// <summary>

/// A function that edits this school.

/// </summary>

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

/// <param name="country"></param>

/// <param name="address"></param>

public void Edit(string name, string country, string address)

{

EditSchool(name, country, address, this.SchoolId);

}

}

#### UserInSchoolTbl – מחלקה האחראית על משתמש בבית ספר

/// <summary>

/// A class that represents the UserInSchoolTbl table in the database

/// </summary>

public partial class UserInSchoolTbl

{

// A static instance of the database context

public static NoteShareContext db = new NoteShareContext();

/// <summary>

/// Get school by user id.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static List<SchoolTbl> GetSchoolsByUserId(int id)

{

List<UserInSchoolTbl> usersInSchool =

db.UserInSchoolTbl.ToList().FindAll(x => x.UserId == id);

List<SchoolTbl> schoolTbls = new List<SchoolTbl>();

foreach (UserInSchoolTbl userInSchool in usersInSchool)

{

schoolTbls.Add(SchoolTbl.GetSchoolByID(userInSchool.SchoolId));

}

return schoolTbls;

}

/// <summary>

/// Get all users in school by user id.

/// </summary>

/// <param name="userId"></param>

/// <returns></returns>

public static List<UserInSchoolTbl> getAllUserInSchoolByUserId(int

userId)

{

return db.UserInSchoolTbl.ToList().FindAll(x => x.UserId ==

userId);

}

/// <summary>

/// Get all users in school by school id.

/// </summary>

/// <param name="schoolId"></param>

/// <returns></returns>

public static List<UserInSchoolTbl>

getAllUserInSchoolBySchoolId(int schoolId)

{

return db.UserInSchoolTbl.ToList().FindAll(x => x.SchoolId ==

schoolId);

}

/// <summary>

/// get all users in school by school id.

/// </summary>

/// <param name="schoolId"></param>

/// <returns></returns>

public static List<UserInSchoolTbl>

getAllUsersInSchoolBySchoolId(int schoolId)

{

return db.UserInSchoolTbl.ToList().FindAll(x => x.SchoolId ==

schoolId);

}

/// <summary>

/// Deletes user in school by user id.

/// </summary>

/// <param name="userId"></param>

public static void DeleteUserInSchool(int userId)

{

foreach (UserInSchoolTbl userInSchool in

getAllUserInSchoolByUserId(userId))

{

db.Remove(userInSchool).State =

Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

}

/// <summary>

/// Delete user in school by school id.

/// </summary>

/// <param name="schoolId"></param>

public static void DeleteUserInSchoolBySchool(int schoolId)

{

foreach (UserInSchoolTbl userInSchool in

getAllUserInSchoolBySchoolId(schoolId))

{

db.Remove(userInSchool).State =

Microsoft.EntityFrameworkCore.EntityState.Deleted;

db.SaveChanges();

}

}

}

#### CommentWS – מחלקה מחקה של מחלקת CommentTbl

מחלקה זו עוברת בשירות הרשת.

//This class is a copy of the CommentTbl class that can be

transfered through a web service

public class CommentWS

{

/// <summary>

/// Constructor that gets all the props and initializes them.

/// </summary>

/// <param name="userId"></param>

/// <param name="notebookId"></param>

/// <param name="comment"></param>

/// <param name="createdDate"></param>

/// <param name="commentId"></param>

public CommentWS(int userId, int notebookId, string comment,

DateTime createdDate, int commentId)

{

UserId = userId;

NotebookId = notebookId;

Comment = comment;

CreatedDate = createdDate;

CommentId = commentId;

}

/// <summary>

/// Copy constructor that copys a CommentTbl object to CommentWS.

/// </summary>

/// <param name="comment"></param>

public CommentWS(CommentTbl comment)

{

UserId = comment.CommentId;

NotebookId = comment.NotebookId;

Comment = comment.Comment;

CreatedDate = comment.CreatedDate;

CommentId = comment.CommentId;

Username = UserTbl.GetUserByUserId(comment.UserId).Username;

}

public CommentWS()

{

}

public int UserId { get; set; }

public int NotebookId { get; set; }

public string Comment { get; set; }

public string Username { get; set; }

public DateTime CreatedDate { get; set; }

public int CommentId { get; set; }

}

#### NotebookWS – מחלקה זו מייצגת מחברת בשירות הרשת

/// <summary>

/// A class that represents a notebook in the web service.

/// </summary>

public class NotebookWS

{

public NotebookWS(int notebookId, int userId, string title, string

description, string color, string format, string path, DateTime

createdDate, DateTime updateDate, string accessibility, int? schoolId, int subjectId, string onlineNotebookFormat)

{

NotebookId = notebookId;

UserId = userId;

Title = title;

Description = description;

Color = color;

Format = format;

Path = path;

CreatedDate = createdDate;

UpdateDate = updateDate;

Accessibility = accessibility;

SchoolId = schoolId;

SubjectId = subjectId;

OnlineNotebookFormat = onlineNotebookFormat;

}

public NotebookWS()

{

}

public NotebookWS(NotebookTbl notebook)

{

NotebookId = notebook.NotebookId;

UserId = notebook.UserId;

Title = notebook.Title;

Description = notebook.Description;

Color = notebook.Color;

Format = notebook.Format;

Path = notebook.Path;

CreatedDate = notebook.CreatedDate;

UpdateDate = notebook.UpdateDate;

Accessibility = notebook.Accessibility;

SchoolId = notebook.SchoolId;

SubjectId = notebook.SubjectId;

OnlineNotebookFormat = notebook.OnlineNotebookFormat;

}

public int NotebookId { get; set; }

public int UserId { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string Color { get; set; }

public string Format { get; set; }

public string Path { get; set; }

public DateTime CreatedDate { get; set; }

public DateTime UpdateDate { get; set; }

public string Accessibility { get; set; }

public int? SchoolId { get; set; }

public int SubjectId { get; set; }

public string OnlineNotebookFormat { get; set; }

}

# צד לקוח

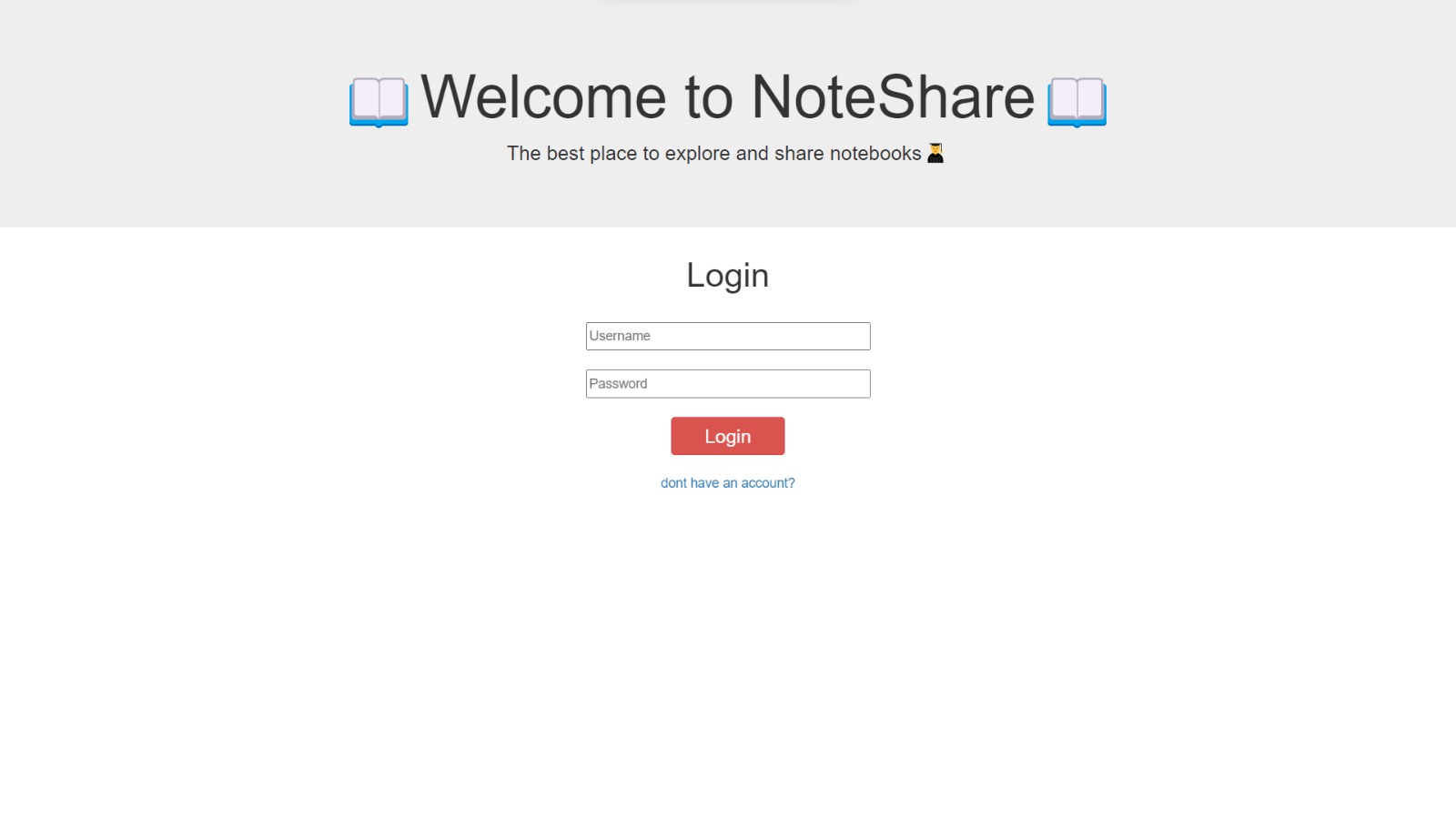
## ממשקי משתמש/ת

### מפת אתר

מדריך למשתמש/ת

## מדריך למשתמש רגיל

1. מרגע פתיחת האתר
   1. אם קיים משתמש:
      1. להכניס את שם המשתמש והסיסמה
      2. ללחוץ על כפתור ההתחברות
   2. אם לא קיים משתמש:
      1. לחיצה על "don’t have an account?"
      2. מילוי הפרטים הנחוצים
      3. Graphical user interface, application

         Description automatically generatedלחיצה על "Register"

1.2

1.1

1. מעבר אל דף הבית. כאן יוכל המשתמש:
   1. ליצור מחברת חדשה **(ראה סעיף 3)**
   2. להיכנס לדף המחברות האהובות שלו **(ראה סעיף 4)**
   3. להיכנס לדף המחברות שלו **(ראה סעיף 5)**
   4. להיכנס לדף המידע האישי שלו **(ראה סעיף 6)**
   5. לחפש מחברות ששמורות במערכת **(ראה סעיף 7)**
   6. בכל שלב מתאפשר למשתמש:
      1. לחיצה על כפתור back בפינה השמאלית העליונה של המסך תחזיר את המשתמש אל דף הבית
      2. Graphical user interface, website

         Description automatically generatedבכל שלב יוכל המשתמש להתנתק מן המערכת על ידי לחיצה על "Log out"

2.4

2.3

2.5

2.1

2.2

1. מעבר אל דף יצירת מחברת
   1. מילוי הפרטים שבטופס
   2. בחירת סוג מחברת:
      1. בחירה בonline editor תאפשר יצירת מחברת וכתיבה באתר
      2. בחירה בdocument תאפשר להעלות קבצים אל המחברת
   3. לחיצה על "Create!" ליצירת המחברת

Graphical user interface, application

Description automatically generated

3.3

3.2

1. מעבר אל דף המחברות האהובות
   1. לחיצה על מחברת תעביר את המשתמש אל דף המחברת **(ראה סעיף 8)**

Graphical user interface, application

Description automatically generated

1. מעבר אל דף המחברות של המשתמש

Graphical user interface, application

Description automatically generatedלחיצה על מחברת תעביר את המשתמש אך דף המחברת **(ראה סעיף 8)**

1. **מעבר אל דף המידע האישי של המשתמש**
   1. **לחיצה על "Edit Info" תפתח פאנל עריכת פרטי משתמש**
   2. **לחיצה על "Submit" תעדכן את הפרטים האישים של המשתמש במערכת**
   3. **לחיצה על "Cancel" תבטל את הפעולה**

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

6.3

6.2

6.1

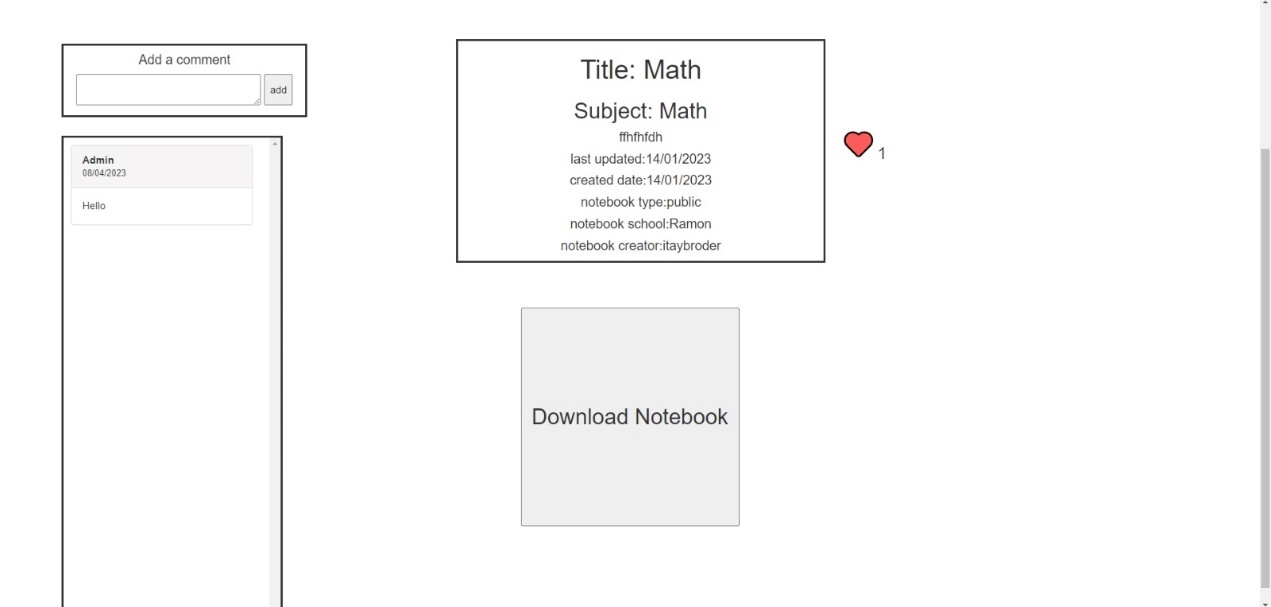
1. **מעבר אל דף חיפוש מחברות**
   1. **ניתן לחפש לפי:**
      1. **שם התלמיד (משתמש)**
      2. **מקצוע נלמד**
      3. **בית הספר**
      4. **תיאור המחברת**
   2. **לחיצה על מחברת תעביר את המשתמש אל דף המחברת (ראה סעיף 8)**

* **ניתן לחפש לפי מספר פרמטרים**
* **מוצגות המחברות האהובות ביותר על משתמשים**
* **מוצגות המחברות האחרונות שנוספו**
* **לא מוצגות מחברות מסוג private**

Graphical user interface, application, Teams

Description automatically generated

1. מעבר אל דף המחברת:
   1. במידה והמשתמש הצופה בדף הוא מחבר המחברת/אדמין:
      1. ניתן למחוק את המחברת באמצעות לחיצה על "Delete notebook" בפינה השמאלית התחתונה של המסך
      2. ניתן למחוק תגובות שנרשמו באמצעות לחיצה על כפתור המחיקה בסמוך לתגובה
      3. ניתן לערוך את תוכן-העל של המחברת באמצעות לחיצה על "Update Notebook"
   2. אם המחברת מסוג Document:
      1. במידה והמשתמש הצופה בדף הוא מחבר המחברת/אדמין:
         1. ניתן להעלות קובץ חדש
      2. ניתן להוריד את תוכן המחברת על ידי לחיצה על "Download Notebook"
   3. אם המחברת מסוג Online:
      1. במידה והמשתמש הצופה בדף הוא מחבר המחברת/אדמין:
         1. ניתן לערוך את תוכן המחברת באמצעות לחיצה על "Switch To Edit View" בפינה הימנית העליונה של המסך
            1. בכדי לשמור את השינויים יש ללחוץ על כפתור ה"Save"
   4. בשני המחברות ניתן לכתוב תגובה למחבר
      1. כתיבת תגובה בשדה הטקסט המתאים
      2. לחיצה על כפתור "add"
   5. Diagram

      Description automatically generated with low confidenceלשני המחברות ניתן לעשות לייק באמצעות לחיצה על כפתור ה"❤️"

8.4.2

8.1.2

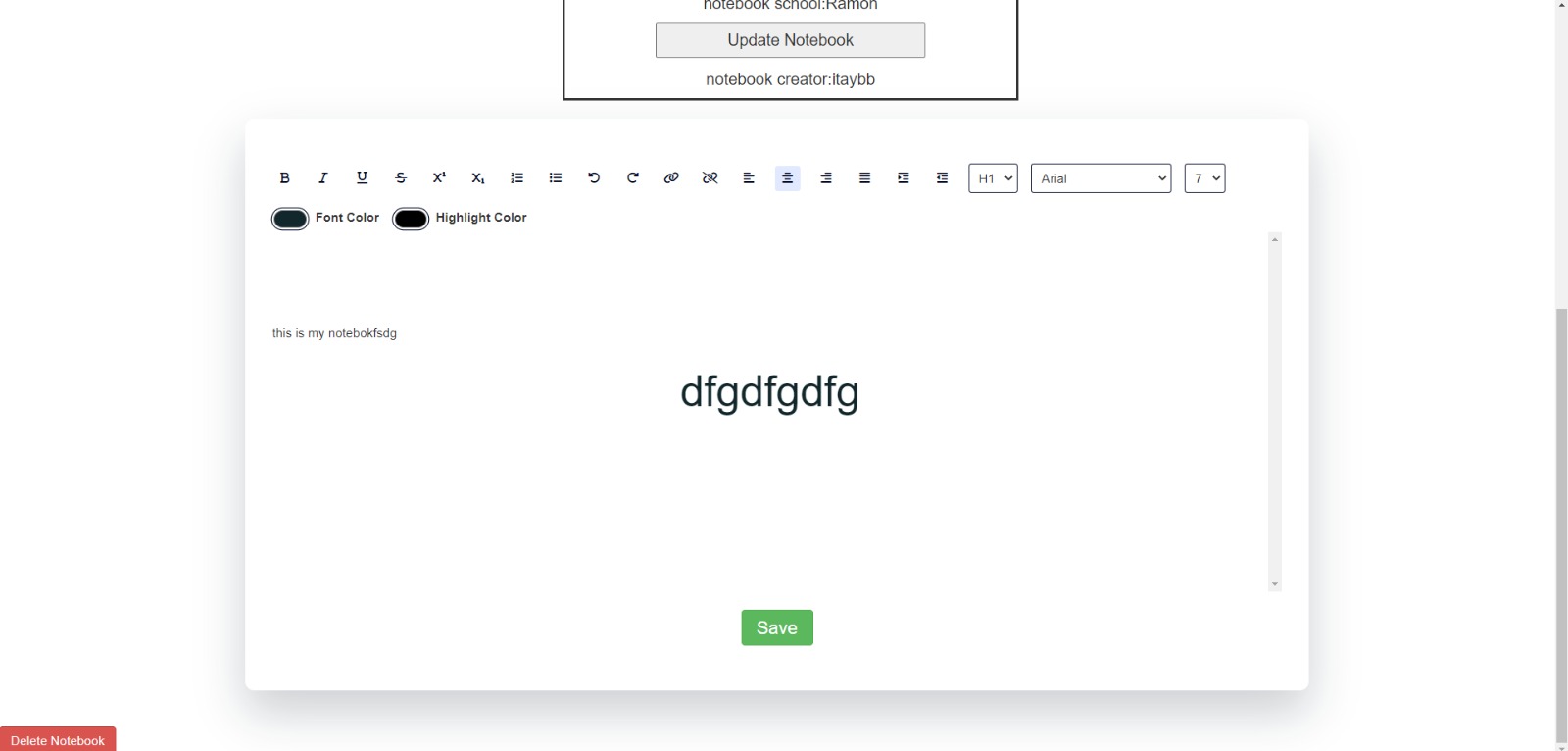
8.1.3

8.3.1.1

8.2.2

8.5

8.4.1



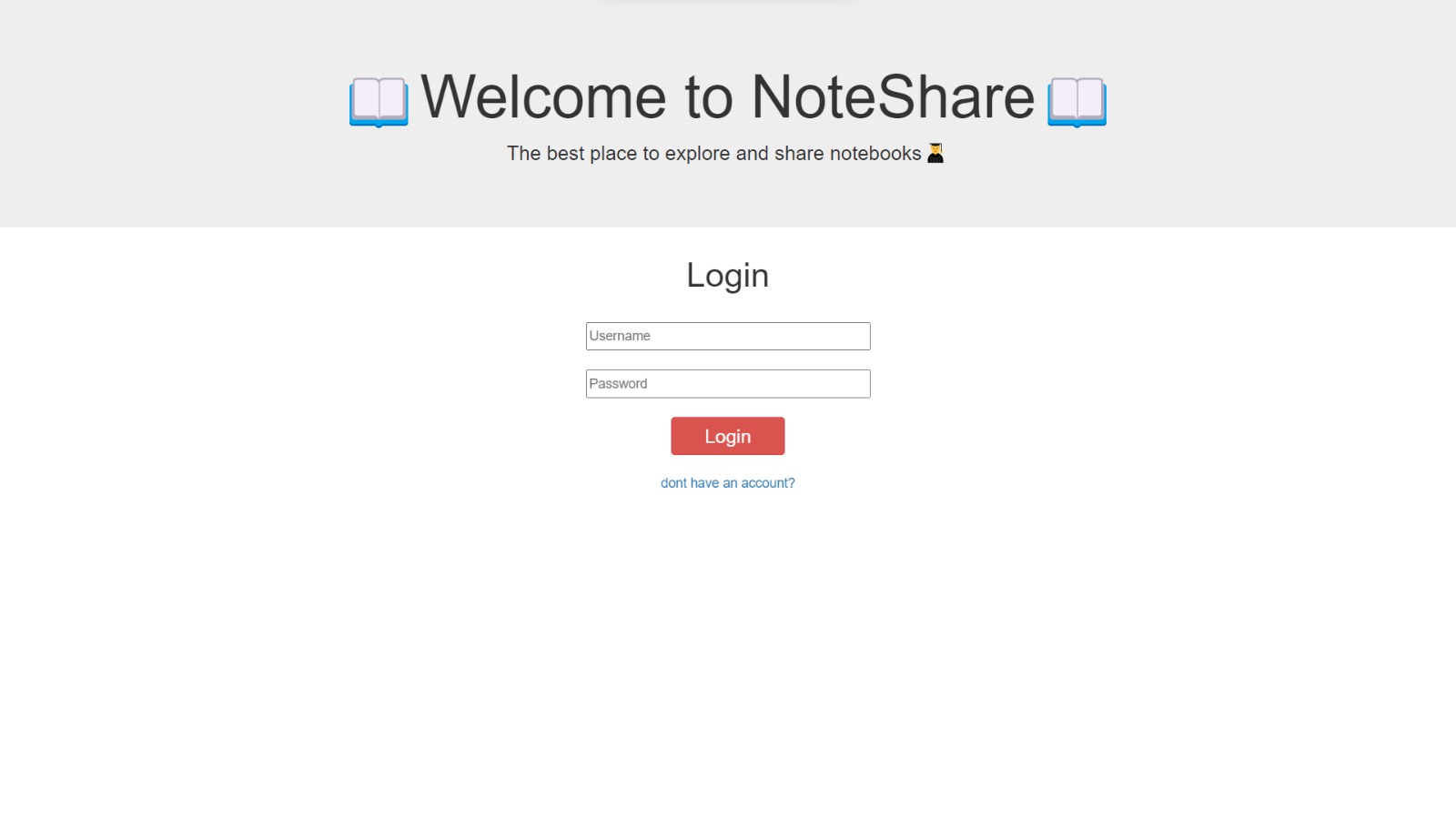
8.1.1

8.3.1.1.1

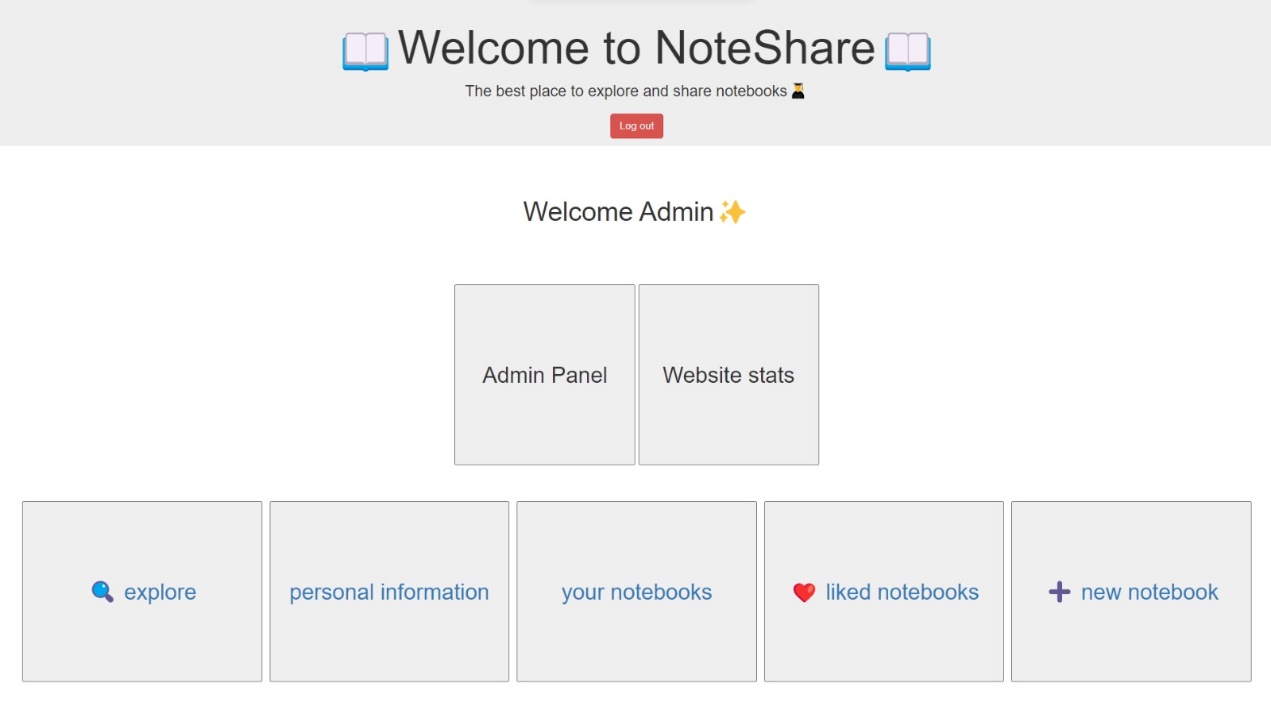
## **מדריך למשתמש אדמין**

האדמין הוא למעשה מנהל המערכת. יש לו גישה אל כולה ואל כל המידע שהיא מכילה.  
יחד עם זאת, האדמין הוא כמו משתמש רגיל במערכת – הוא יכול ליצור מחברות חדשות, לאהוב מחברות קיימות, לצפות במחברות שלו, לעדכן את פרטיו במערכת ולחפש מחברות שמצויות במערכת.

1. מרגע פתיחת האתר:
   1. להכניס את שם המשתמש והסיסמה
   2. ללחוץ על כפתור ההתחברות



1. מעבר אל דף הבית. כאן יוכל האדמין:
   1. ליצור מחברת חדשה **(ראה סעיף 3 במדריך למשתמש רגיל)**
   2. להיכנס לדף המחברות האהובות שלו **(ראה סעיף 4 במדריך למשתמש רגיל)**
   3. להיכנס לדף המחברות שלו **(ראה סעיף 5 במדריך למשתמש רגיל)**
   4. להיכנס לדף המידע האישי שלו **(ראה סעיף 6 במדריך למשתמש רגיל)**
   5. לחפש מחברות ששמורות במערכת **(ראה סעיף 7 במדריך למשתמש רגיל)**
   6. בכל שלב מתאפשר לאדמין:
      1. לחיצה על כפתור back בפינה השמאלית העליונה של המסך תחזיר את המשתמש אל דף הבית
      2. בכל שלב יוכל המשתמש להתנתק מן המערכת על ידי לחיצה על "Log out"
   7. בניגוד למשתמש רגיל, לאדמין יש גישה:
      1. לדף הסטטיסטיקה של האתר **(ראה סעיף 3)**
      2. אל פאנל הניהול היעודי שלו **(ראה סעיף 4)**



2.7.2

2.7.1

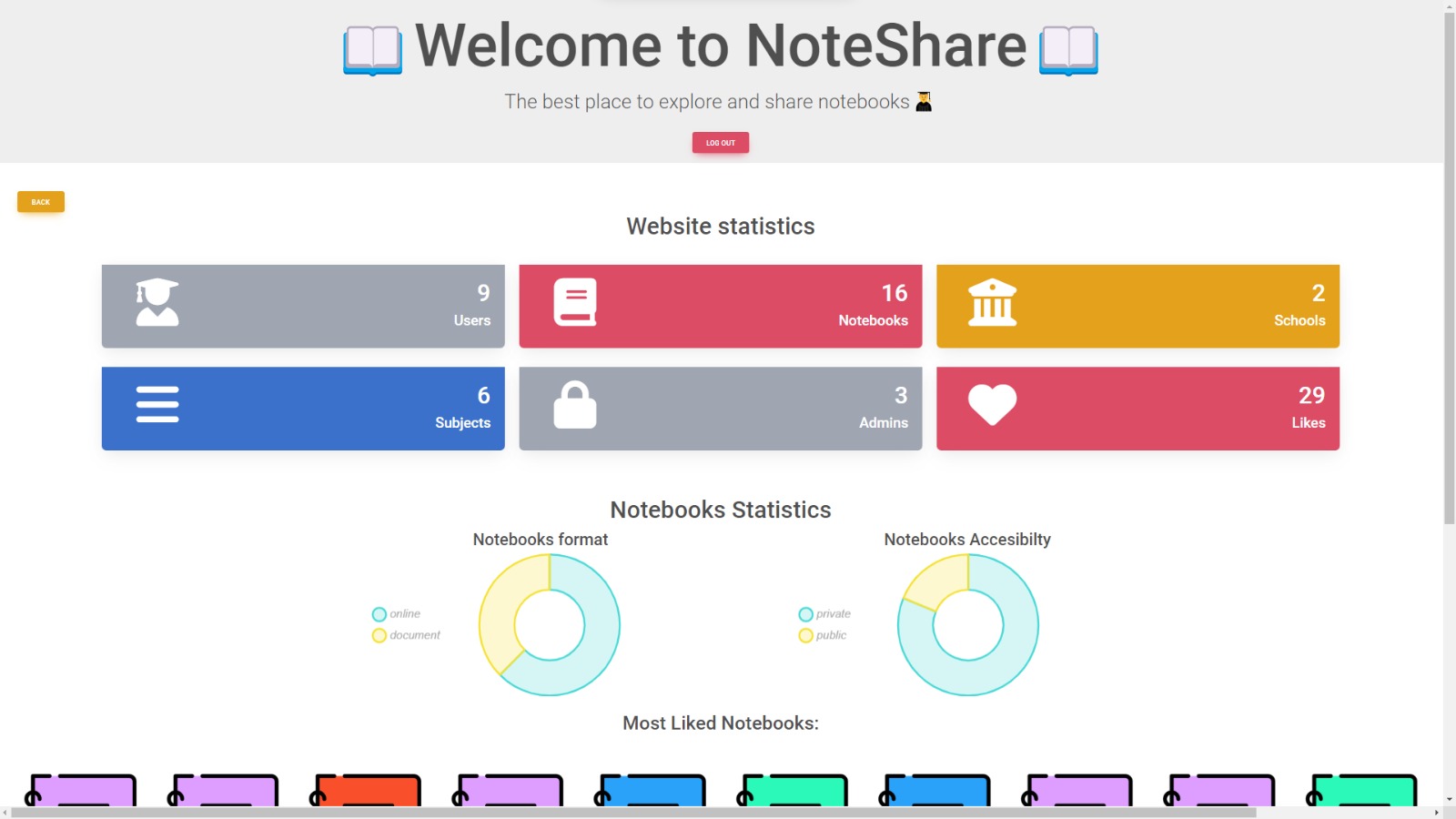
2.5

2.4

2.1

2.2

2.3

1. מעבר אל דף הסטטיסטיקה של האתר
   1. בחלקו העליון של הדף – מוצגים הנתונים של האתר:
      1. מספר בתי הספר במערכת
      2. מספר המחברות במערכת
      3. מספר המשתמשים במערכת
      4. מספר הלייקים הכולל במערכת
      5. מספר האדמינים במערכת
      6. מספר הנושאים במערכת
   2. בחלקו האמצעי של הדף – מוצגים הנתונים של המחברות:
      1. מוצגים חתכים של:
         1. סוג המחברת
         2. סוג גישת המחברת
      2. מוצגות במחברות האהובות ביותר. לחיצה על מחברת תעביר את האדמין אל דף המחברת **(ראה סעיף 8 במדריך למשתמש רגיל)**
   3. בחלקו התחתון של בדף – מוצגים המשתמשים הפעילים של המערכת
      1. לחיצה על "Delete" תסיר את המשתמש מהמערכת
      2. Graphical user interface, application

         Description automatically generatedלחיצה על "Profile" תציג את דף המשתמש **(ראה סעיף 5)**

3.3.2

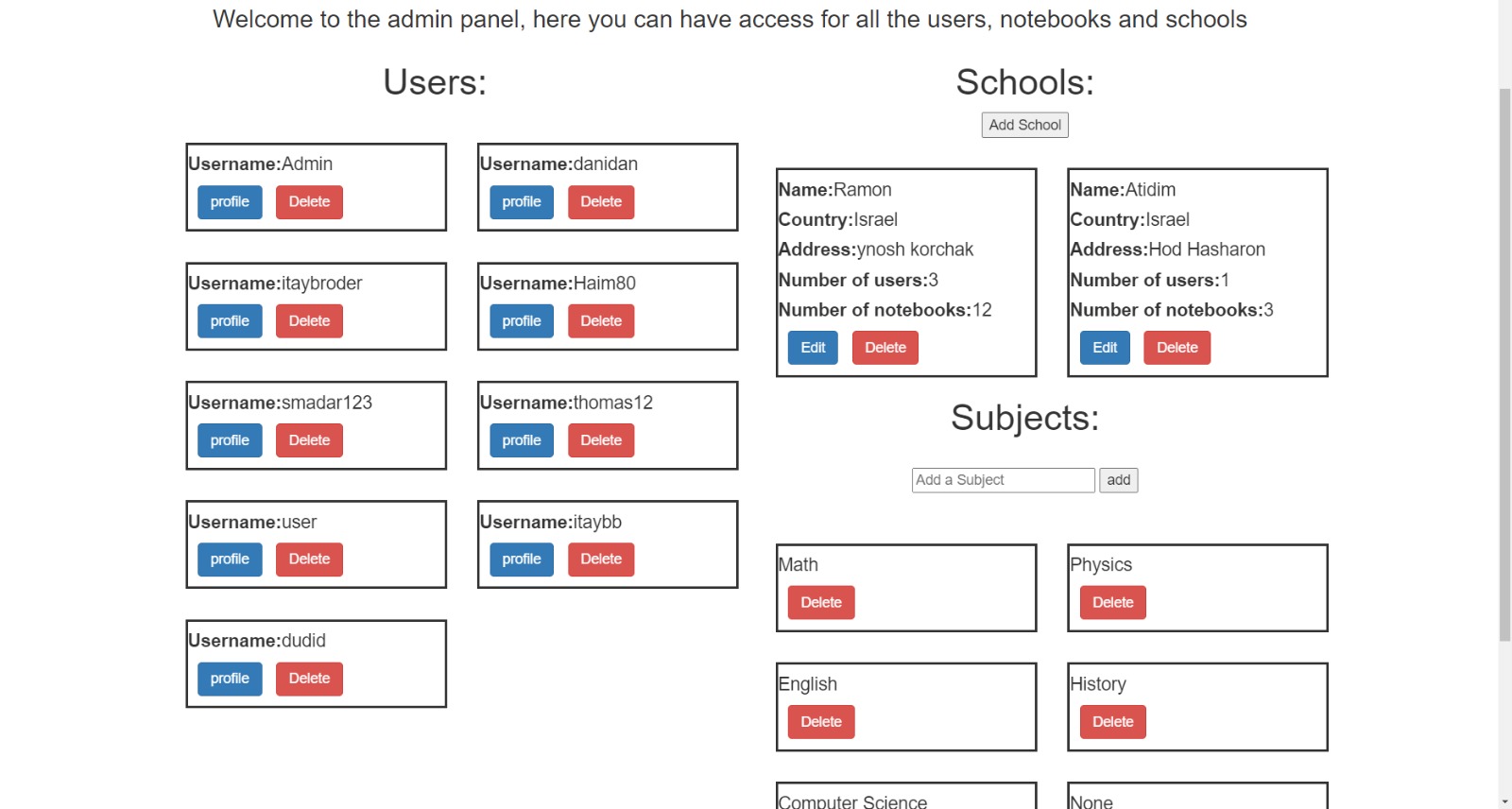
3.3.1

3.2.2

3.2.1

3.1

1. מעבר אל פאנל הניהול הייעודי של האדמין
   1. ניתן לראות את המשתמשים הקיימים במערכת
      1. לחיצה על "Delete" תמחק את המשתמש מן המערכת
      2. לחיצה על "Profile" תעביר את האדמין אל דף המשתמש הנבחר **(ראה סעיף 5)**
      3. בכדי לחפש משתמש, לכתוב את שם המשתמש, או חלק ממנו בשדה הטקסט וללחוץ אנטר.
   2. ניתן לראות את בתי הספר הקיימים במערכת
      1. לחיצה על "Add School" תפתח פאנל הוספת בית ספר
         1. יש למלא את הפרטים במקומות המתאימים
         2. לחיצה על "cancel" תבטל את הפעולה
         3. לחיצה על "add" תוסיף בית ספר חדש למערכת
      2. לחיצה על "Delete" תמחק את בית הספר מן המערכת
      3. לחיצה על "Edit" תפתח פאנל שבו ניתן לערוך את פרטי בית הספר
   3. ניתן לראות את מקצועות המחברות הקיימים במערכת
      1. בכדי להוסיף מקצוע חדש:
         1. הזן את שם המקצוע בשדה הטקסט
         2. לחץ על "add"
      2. לחיצה על "Delete" תמחק את המקצוע מן המערכת



4.1.3

4.1.1

4.3.2

4.3.1.2

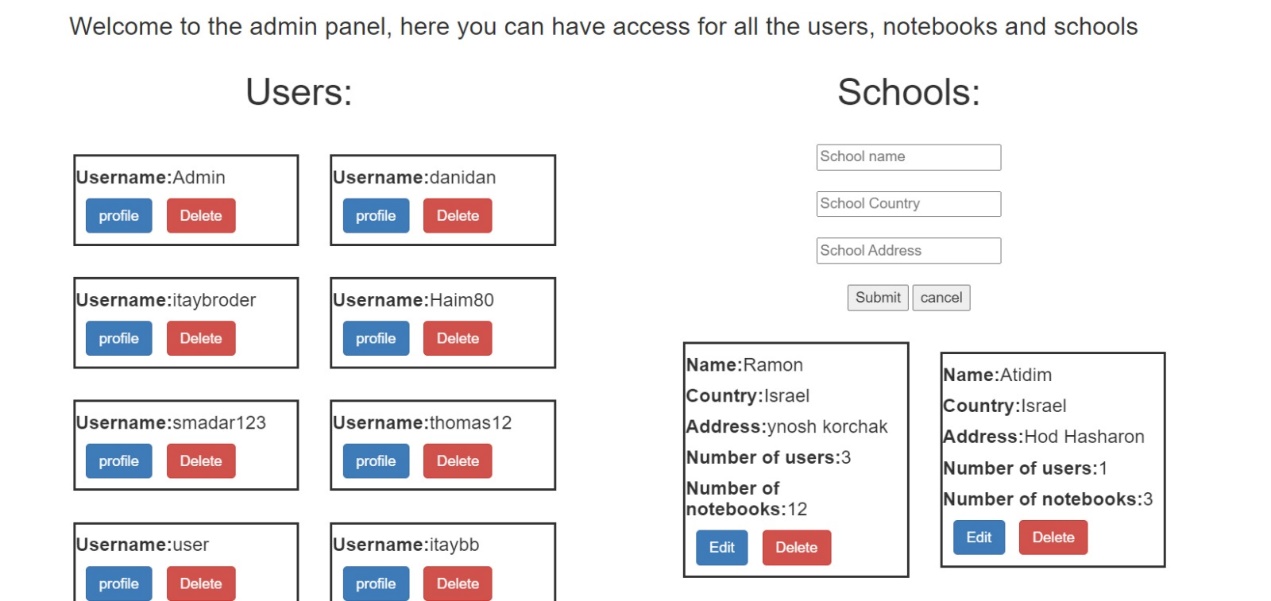
4.3.1.1

4.2.3

4.2.2

4.2.1

4.1.2



4.2.1.3

4.2.1.1

4.2.1.2

1. מעבר אל דף משתמש כאדמין
   1. כעת האדמין מחובר בתור המשתמש הנבחר
   2. לחיצה על "Log Out" בפינה השמאלית העליונה של המסך תנתק את האדמין מן המשתמש הנבחר
   3. האדמין יכול:
      1. לנווט באתר כאילו היה המשתמש המחובר **(ראה סעיף 2 במדריך למשתמש רגיל)**
      2. לגשת אל אזורי הניהול שלו:
         1. אל דף הסטטיסטיקה של האתר **(ראה סעיף 3)**
         2. Graphical user interface, website

            Description automatically generatedאל פאנל הניהול היעודי שלו **(ראה סעיף 4)**

5.2

סיכום אישי / רפלקציה

נהנתי מאוד לעבוד על הפרוייקט, מדובר בפרוייקט שקרוב ללבי מכיוון שאני סטודנט ואני אוהב פיתוח WEB אני מאמין שהצלחתי להגיע לפרוייקט מוצלח שעובד היטב. האתגרים והקשיים היו בעיקר צורך בשעות רבות של עבודה על הפרוייקט, שהצלחתי לפתור בעזרת ניהול זמן נכון ובנוסף היו חלקים בקוד שלא ידעתי איך לממש, דרכי הפתרון שלי היו בייעוץ ועזרה שך המורות המלוות, סמדר ורחלי וחיפוש באינטרנט בפורומים שונים. הלמידה הייתה רבה, למדתי הרבה על טכנולוגיות WEB על פיתוח של backend וfrontend עבודה עם מסדי נתונים וSQL בעצמי למדתי גאווה סקריפט וbootstrap על מנת לעצב את הUI. הכלים שנלקחים להמשך הם הכלים לפיתוח בסדר גודל גדול, עם פיצול לשכבות וניהול זמן מיטבי. בעזרת העזרה שקיבלתי ממורותי, למדתי כיצד לעבוד בשכבות וכיצד לממש חלקים שונים בפרוייקט. בנוסף לכך, התייעצתי עם חבריי לכיתה בנושאים בהם היה צורך ושיתפתי אותם גם בידע שצברתי לבעיות שעלו להם וכך כולנו השתפרנו יחד. בראייה לאחור, לא הייתי עורך שינויים רבים בפרוייקט אם כי במחשבה לאחור יתכן שהייתי משנה את הטיפול בספריית עיצוב שהשתמשתי בה bootstrap לספרייה יותר מתקדמת על מנת ליצור עיצוב יותר יפה. אם היו ברשותי משאבים נוספים, הייתי מוסיף מערכת מתקדמת יותר לשמירת המחברות, במקום לשמור אותן כפורמט של html במסד הנתונים, אלא בונה מערכת מורכבת יותר.

# ביבליוגרפיה

* Microsoft. (n.d.). C# documentation. Retrieved from <https://docs.microsoft.com/en-us/dotnet/csharp/>
* Stack Overflow. (n.d.). Retrieved from <https://stackoverflow.com/>
* W3Schools. (n.d.). Web development tutorials. Retrieved from <https://www.w3schools.com/>
* Draw.io. (n.d.). Online diagram software & visual solution. Retrieved from <https://app.diagrams.net/>
* Flaticon. (n.d.). The largest database of free icons. Retrieved from <https://www.flaticon.com/free-icons/>
* Logo.com. (n.d.). Logo creation made simple. Retrieved from <https://logo.com/>

# נספחים

## AdminPanel.aspx

<style>

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<center>

<h3>Welcome to the admin panel, here you can have access for all the users, notebooks and schools</h3>

<div class="container animate\_\_animated animate\_\_bounceInLeft">

<div class="row">

<div class="col-lg-6">

<h1>Users:</h1>

<asp:TextBox runat="server" placeholder="search" OnTextChanged="Search\_TextChanged" ID="SearchText" AutoPostBack = True/>

<asp:DataList ID="UsersDataList" runat="server" Height="117px" OnItemCommand="Users\_ItemCommand" Width="268px" RepeatColumns="2">

<ItemTemplate>

<div class="school-block">

<h4><b>Username:</b><asp:Label Text='<%#Eval("Username") %>' runat="server" /></h4>

<asp:Button Text="profile" runat="server" CssClass="btn btn-primary" Style="margin-left: 10px; margin-bottom: 10px;" CommandName="UserClick" CommandArgument='<%# Eval("UserId") %>' />

<asp:Button Text="Delete" runat="server" Style="margin-left: 10px; margin-bottom: 10px;" class="btn btn-danger" CommandName="DeleteUser" CommandArgument='<%#Eval("UserId") %>' />

</div>

</ItemTemplate>

</asp:DataList>

</div>

<div class="col-lg-6">

<style>

.school-block {

border-style: solid;

margin-top: 30px;

margin-left: 30px;

width: 30vh;

}

</style>

<h1>Schools:</h1>

<asp:Button Text="Add School" runat="server" ID="AddSchoolButton" OnClick="AddSchoolButton\_Click" />

<asp:Panel runat="server" ID="createSchool" Visible="false">

<br />

<asp:TextBox runat="server" ID="CreateSchoolName" placeholder="School name" /><br />

<br />

<asp:TextBox runat="server" ID="CreateSchoolCountry" placeholder="School Country" /><br />

<br />

<asp:TextBox runat="server" ID="CreateSchoolAddress" placeholder="School Address" />

<br />

<br />

<asp:Button Text="Submit" runat="server" ID="submitSchoolButton" OnClick="submitSchoolButton\_Click" />

<asp:Button Text="cancel" runat="server" ID="CancelButton" OnClick="CancelButton\_Click" />

</asp:Panel>

<asp:Panel runat="server" ID="EditSchool" Visible="false">

<br />

<asp:TextBox runat="server" ID="EditSchoolName" placeholder="School name" /><br />

<br />

<asp:TextBox runat="server" ID="EditSchoolCountry" placeholder="School Country" /><br />

<br />

<asp:TextBox runat="server" ID="EditSchoolAddress" placeholder="School Address" />

<br />

<asp:Label Text="" runat="server" Visible="false" ID="schoolIdLabel" />

<br />

<asp:Button Text="Submit" runat="server" ID="EditSchoolSummit" OnClick="EditSchoolSummit\_Click" />

<asp:Button Text="cancel" runat="server" ID="Button2" OnClick="CancelButton\_Click" />

</asp:Panel>

<asp:DataList ID="SchoolsDataList" runat="server" Height="117px" OnItemCommand="Schools\_ItemCommand" Width="268px" RepeatColumns="2">

<ItemTemplate>

<div class="school-block">

<h4><b>Name:</b><asp:Label Text='<%#Eval("Name") %>' runat="server" /></h4>

<h4><b>Country:</b><asp:Label Text='<%#Eval("Country") %>' runat="server" /></h4>

<h4><b>Address:</b><asp:Label Text='<%#Eval("Address") %>' runat="server" /></h4>

<h4><b>Number of users:</b><asp:Label Text='<%# NoteShare.Models.UserInSchoolTbl.getAllUsersInSchoolBySchoolId((int)Eval("SchoolId")).Count %>' runat="server" /></h4>

<h4><b>Number of notebooks:</b><asp:Label Text='<%# NoteShare.Models.NotebookTbl.getNotebooksFromSchoolId((int)Eval("SchoolId")).Count %>' runat="server" /></h4>

<asp:Button Text="Edit" runat="server" Style="margin-left: 10px; margin-bottom: 10px;" CssClass="btn btn-primary" CommandName="EditSchool" CommandArgument='<%#Eval("SchoolId") %>' />

<asp:Button Text="Delete" runat="server" Style="margin-left: 10px; margin-bottom: 10px;" class="btn btn-danger" CommandName="DeleteSchool" CommandArgument='<%#Eval("SchoolId") %>' />

</div>

</ItemTemplate>

</asp:DataList>

</div>

<div class="col-lg-6">

<style>

.school-block {

border-style: solid;

margin-top: 30px;

margin-left: 30px;

width: 30vh;

}

</style>

<h1>Subjects:</h1>

<asp:Panel runat="server" ID="Panel1">

<br />

<asp:TextBox runat="server" ID="SubjectTextBox" placeholder="Add a Subject" />

<asp:Button Text="add" runat="server" ID="AddSubjectButton" OnClick="AddSubjectButton\_Click" /><br />

<br />

</asp:Panel>

<asp:DataList ID="SubjectsDataList" runat="server" Height="117px" Width="268px" OnItemCommand="SubjectsDataList\_ItemCommand" RepeatColumns="2">

<ItemTemplate>

<div class="school-block">

<h4>

<asp:Label Text='<%#Eval("Name") %>' runat="server" /></h4>

<asp:Button Text="Delete" runat="server" Style="margin-left: 10px; margin-bottom: 10px;" class="btn btn-danger" CommandName="DeleteSubject" CommandArgument='<%#Eval("SubjectId") %>' />

</div>

</ItemTemplate>

</asp:DataList>

</div>

</div>

</div>

</center>

## AdminPanel.aspx.cs

public partial class AdminPanel : System.Web.UI.Page

{

// The user connected.

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

if (user.Permission != "admin")

{

Response.Redirect("login.aspx");

return;

}

if (!Page.IsPostBack)

{

RefreshLists();

}

}

/// <summary>

/// The function refreshes all the lists in the admin panel.

/// </summary>

public void RefreshLists()

{

List<UserTbl> users = UserTbl.GetAllUsers();

UsersDataList.DataSource = users;

UsersDataList.DataBind();

List<SchoolTbl> schools = SchoolTbl.GetSchools();

SchoolsDataList.DataSource = schools;

SchoolsDataList.DataBind();

List<SubjectTbl> subjects = SubjectTbl.GetSubjects();

SubjectsDataList.DataSource = subjects;

SubjectsDataList.DataBind();

}

/// <summary>

/// This function is called when an item command occures on the users DataList.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Users\_ItemCommand(object source, DataListCommandEventArgs e)

{

//Delete user item command.

if (e.CommandName == "DeleteUser")

{

UserTbl userToDelete = new UserTbl(int.Parse((String)e.CommandArgument));

userToDelete.Delete();

RefreshLists();

}

//user click item command. logs you in as a user.

if (e.CommandName == "UserClick")

{

UserTbl user = UserTbl.GetUserByUserId(int.Parse((String)e.CommandArgument));

Session["UserFromAdmin"] = user;

Response.Redirect("Home.aspx");

}

}

/// <summary>

/// This function is called when the search text changes. and it filters the users list.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void Search\_TextChanged(object sender, EventArgs e)

{

List<UserTbl> users;

if (SearchText.Text == "")

{

users = UserTbl.GetAllUsers();

}

else

{

users = UserTbl.GetAllUsers().FindAll(t => t.Username.StartsWith(SearchText.Text));

}

UsersDataList.DataSource = users;

UsersDataList.DataBind();

}

/// <summary>

/// This function is called when an item command occures on the schools DataList

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Schools\_ItemCommand(object source, DataListCommandEventArgs e)

{

//Delete school item command.

if (e.CommandName == "DeleteSchool")

{

SchoolTbl.DeleteSchool(int.Parse((String)e.CommandArgument));

RefreshLists();

}

//Edit school item command.

else if (e.CommandName == "EditSchool")

{

SchoolTbl school = SchoolTbl.GetSchoolByID(int.Parse((String)e.CommandArgument));

EditSchool.Visible = true;

EditSchoolName.Text = school.Name;

EditSchoolCountry.Text = school.Country;

EditSchoolAddress.Text = school.Address;

schoolIdLabel.Text = (String)e.CommandArgument;

}

}

/// <summary>

/// Adds school.

/// makes the addschool button visible

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void AddSchoolButton\_Click(object sender, EventArgs e)

{

//makes the addschool button visible

createSchool.Visible = true;

AddSchoolButton.Visible = false;

}

/// <summary>

/// Cancels the create school state.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void CancelButton\_Click(object sender, EventArgs e)

{

createSchool.Visible = false;

AddSchoolButton.Visible = true;

}

/// <summary>

/// Submits the create school form.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void submitSchoolButton\_Click(object sender, EventArgs e)

{

string name = CreateSchoolName.Text;

string country = CreateSchoolCountry.Text;

string address = CreateSchoolAddress.Text;

//updates the database

SchoolTbl school = SchoolTbl.CreateSchool(name, country, address);

List<SchoolTbl> schools = SchoolTbl.GetSchools();

RefreshLists();

CancelButton\_Click(null, null);

}

/// <summary>

/// Returns to the home page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

/// <summary>

/// Submits the edit school form.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void EditSchoolSummit\_Click(object sender, EventArgs e)

{

string name = EditSchoolName.Text;

string country = EditSchoolCountry.Text;

string address = EditSchoolAddress.Text;

SchoolTbl school = SchoolTbl.EditSchool(name, country, address, int.Parse(schoolIdLabel.Text));

RefreshLists();

EditSchool.Visible = false;

}

/// <summary>

/// Creating a subject.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void AddSubjectButton\_Click(object sender, EventArgs e)

{

//Subject name.

string newSubjectText = SubjectTextBox.Text;

SubjectTbl.CreateSubject(newSubjectText);

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

/// <summary>

/// Subjects list item commands.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void SubjectsDataList\_ItemCommand(object source, DataListCommandEventArgs e)

{

//Delets a subject on click.

if (e.CommandName == "DeleteSubject")

{

SubjectTbl.DeleteSubject(int.Parse((String)e.CommandArgument));

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

}

}

## CreateNotebook.aspx

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" Style="margin-left: 20px;" CssClass="btn btn-warning" />

<center>

<style>

img {

width: 140px;

height: 140px;

}

</style>

<h1>

<asp:Label Text="Create your own notebook!" runat="server" ID="ModeLabel" /></h1>

<br />

<asp:TextBox runat="server" ID="TitleLBL" placeholder="Notebook Title" /><br />

<br />

<h2>Pick a subject:</h2>

<asp:DropDownList runat="server" ID="SubjectsDropdownlist" DataTextField="Name" DataValueField="Name" SelectMethod="GetSubjects">

</asp:DropDownList>

<br />

<br />

<asp:TextBox runat="server" Height="127px" TextMode="MultiLine" Width="361px" ID="Description" placeholder="notebook description..." /><br />

<br />

<h4>choose your notebook accessibility</h4>

<asp:RadioButtonList ID="accessibilityList" runat="server" RepeatDirection="Horizontal">

<asp:ListItem Text='public' Value="public" style="margin-right: 25px; font-size: 20px;" />

<asp:ListItem Text='private' Value="private" Selected="True" style="font-size: 20px;" />

</asp:RadioButtonList>

<h4>choose a color</h4>

<asp:RadioButtonList ID="ColorsList" runat="server" RepeatDirection="Horizontal" >

<asp:ListItem Text='<img src="colors/red.png" alt="img1"/>' Value="red.png" Selected="True" />

<asp:ListItem Text='<img src="colors/blue.png" alt="img2" />' Value="blue.png" />

<asp:ListItem Text='<img src="colors/purple.png" alt="img2" />' Value="purple.png" />

<asp:ListItem Text='<img src="colors/green.png" alt="img2" />' Value="green.png" />

</asp:RadioButtonList>

<br />

<h4>choose the notebook school</h4>

<asp:DropDownList runat="server" ID="SchoolsDropDownList" DataTextField="Name" DataValueField="SchoolId" SelectMethod="GetUserSchools">

</asp:DropDownList>

<h4>choose your notebook format</h4>

<asp:RadioButtonList ID="FormatList" runat="server" RepeatDirection="Horizontal" OnSelectedIndexChanged="FormatList\_SelectedIndexChanged" AutoPostBack="true">

<asp:ListItem Text='document' Value="document" style="margin-right: 25px; font-size: 20px;" />

<asp:ListItem Text='online editor' Value="online" Selected="True" style="font-size: 20px;" />

</asp:RadioButtonList>

<asp:FileUpload ID="DocumentUpload" runat="server" /><br />

<asp:Button Text="Create!" runat="server" OnClick="SubmitData" ID="SubmitButton" CssClass="btn btn-primary" /><br />

<br />

</center>

## CreateNotebook.aspx.cs

public partial class WebForm3 : System.Web.UI.Page

{

//The current User.

public static UserTbl user;

/// <summary>

/// Returns the Id of the notebook from the url.

/// </summary>

/// <returns></returns>

public static string GetIDFromUrl()

{

string desiredValue = "";

foreach (string item in HttpContext.Current.Request.Url.Query.Split('&'))

{

string[] parts = item.Replace("?", "").Split('=');

if (parts[0] == "id")

{

desiredValue = parts[1];

break;

}

}

return desiredValue;

}

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["User"] == null)

{

Response.Redirect("Home.aspx");

}

user = ((UserTbl)Session["User"]);

if (!IsPostBack)

{

string desiredValue = GetIDFromUrl();

if (desiredValue != "")

{

Session["CurrentNotebook"] = NotebookTbl.GetNotebookByNotebookID(Int32.Parse(desiredValue));

NotebookTbl notebook = (NotebookTbl)Session["CurrentNotebook"];

TitleLBL.Text = notebook.Title;

Description.Text = notebook.Description;

ColorsList.SelectedValue = notebook.Color;

FormatList.SelectedValue = notebook.Format;

SubjectsDropdownlist.SelectedValue = SubjectTbl.GetSubjectByID(notebook.SubjectId).Name;

ModeLabel.Text = "Update notebook";

accessibilityList.SelectedValue = notebook.Accessibility;

SubmitButton.Text = "Update notebook";

}

}

if (user.Permission == "admin")

{

if (Session["UserFromAdmin"] != null)

{

user = ((UserTbl)Session["UserFromAdmin"]);

}

}

FormatList\_SelectedIndexChanged(null, null);

ValidationSettings.UnobtrusiveValidationMode = UnobtrusiveValidationMode.None;

}

/// <summary>

/// Submits all the notebook data to the DB.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void SubmitData(object sender, EventArgs e)

{

NotebookTbl notebook = (NotebookTbl)Session["CurrentNotebook"];

int userID = user.UserId;

string title = TitleLBL.Text;

string description = Description.Text;

string color = ColorsList.SelectedValue;

string format = FormatList.SelectedValue;

string accessibility = accessibilityList.SelectedValue;

string subject = SubjectsDropdownlist.SelectedValue;

int schoolId = 0;

if (SchoolsDropDownList.SelectedValue != null && SchoolsDropDownList.SelectedValue != "")

{

schoolId = int.Parse(SchoolsDropDownList.SelectedValue);

}

if (notebook == null)

{

notebook = NotebookTbl.CreateNewNotebook(userID, title, description, color, format, "", accessibility, schoolId, subject);

}

else

{

notebook = NotebookTbl.UpdateNotebook(notebook.NotebookId, userID, title, description, color, format, notebook.Path, accessibility, schoolId, subject);

}

if (format == "document")

{

if (DocumentUpload.HasFile)

{

string[] tmp = DocumentUpload.FileName.Split('.');

string ext = tmp[tmp.Length - 1];

string path = $"notebookfiles/notebook-{notebook.NotebookId}.{ext}";

DocumentUpload.SaveAs(Server.MapPath(path));

NotebookTbl.UpdatePath(notebook, path);

}

}

Response.Redirect("Home.aspx");

}

/// <summary>

/// returns to home page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

/// <summary>

/// Toggels the document upload, if the user chose the document notebook format.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void FormatList\_SelectedIndexChanged(object sender, EventArgs e)

{

if (FormatList.SelectedValue == "document")

{

DocumentUpload.Visible = true;

}

else

{

DocumentUpload.Visible = false;

}

}

/// <summary>

/// Returns all the subjects, used for the dropdown list.

/// </summary>

/// <returns></returns>

public static List<SubjectTbl> GetSubjects()

{

return SubjectTbl.GetSubjects();

}

/// <summary>

/// Returns all the schools, used for the dropdown list.

/// </summary>

/// <returns></returns>

public static List<SchoolTbl> GetUserSchools()

{

return UserInSchoolTbl.GetSchoolsByUserId(user.UserId);

}

}

## Explore.aspx

<style>

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<style>

.data-list-item {

width: 150px;

height: 150px;

font-size: 10px;

transition: 0.4s;

}

.data-list-item:hover {

width: 180px;

height: 180px;

font-size: 12px;

}

</style>

<center>

<h1>Explore Notebooks</h1>

<div>

<asp:TextBox runat="server" type="text" ID="UserTB" placeholder="User" />

<asp:DropDownList runat="server" ID="SubjectsDropdownlist" AppendDataBoundItems=true DataTextField="Name" DataValueField="Name" SelectMethod="GetSubjects">

</asp:DropDownList>

<asp:DropDownList runat="server" ID="SchoolsDropDownList" DataTextField="Name" DataValueField="Name" SelectMethod="GetUserSchools">

</asp:DropDownList>

<asp:TextBox runat="server" type="text" ID="DescripionTB" placeholder="Descripion" />

</div>

<br />

<asp:Button Text="Search" runat="server" ID="SearchButton" CssClass="btn btn-primary" OnClick="SearchButton\_Click" /><br />

<br />

<asp:Label Text="Notebooks in this context doesn't exist" runat="server" Visible="false" ID="NotFoundLabel" />

<asp:DataList ID="Notebooks" runat="server" Height="117px" OnItemCommand="Notebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

<br />

<br />

<h1>Explore the most liked notebooks❤️</h1>

<br />

<br />

<asp:Label Text="Notebooks in this context doesn't exist" runat="server" Visible="false" ID="Label1" />

<asp:DataList ID="MostLikedDataList" runat="server" Height="117px" OnItemCommand="MostLikeNotebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

<br />

<br />

<h1>Reacently added notebooks</h1>

<br />

<br />

<asp:Label Text="Notebooks in this context doesn't exist" runat="server" Visible="false" ID="Label2" />

<asp:DataList ID="RecentlyAddedNotebooks" runat="server" Height="117px" OnItemCommand="Notebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

<h4 style="text-align: center;">

<asp:Label ID="CreateDateLBL" Text='<%# ((DateTime)Eval("CreatedDate")).ToString("dd/MM/yyyy") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

</center>

## Explore.aspx.cs

public partial class Explore : System.Web.UI.Page

{

// The current user.

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

if (!IsPostBack)

{

if (user.Permission == "admin")

{

MostLikedDataList.DataSource = MostLikeNotebookList(10);

RecentlyAddedNotebooks.DataSource = RecentlyAddedNotebookList(10);

}

else

{

//show only public notebooks for regular users.

MostLikedDataList.DataSource = MostLikeNotebookList(10).FindAll(x => x.Accessibility == "public" || x.UserId == user.UserId);

RecentlyAddedNotebooks.DataSource = RecentlyAddedNotebookList(10).FindAll(x => x.Accessibility == "public" || x.UserId == user.UserId);

}

MostLikedDataList.DataBind();

RecentlyAddedNotebooks.DataBind();

}

}

/// <summary>

/// Returns all the subjects, used for the dropdown list.

/// </summary>

/// <returns></returns>

public static List<SubjectTbl> GetSubjects()

{

SubjectTbl emp = new SubjectTbl();

emp.Name = "-- Select Subject --";

List<SubjectTbl > subjects = SubjectTbl.GetSubjects();

subjects.Insert(0, emp);

return subjects;

}

/// <summary>

/// Returns all the schools, used for the dropdown list.

/// </summary>

/// <returns></returns>

public static List<SchoolTbl> GetUserSchools()

{

SchoolTbl empty = new SchoolTbl();

empty.Name = "-- Select School --";

List<SchoolTbl> schools = SchoolTbl.GetSchools();

schools.Insert(0, empty);

return schools;

}

/// <summary>

/// Returns the most liked notebooks.

/// </summary>

/// <param name="number">How many notebooks to return</param>

/// <returns></returns>

public static List<NotebookTbl> MostLikeNotebookList(int number)

{

List<LikeTbl> likedNotebooks = LikeTbl.GetAllLikes();

Dictionary<int, int> mostLikedNotebooksIds = new Dictionary<int, int>();

foreach (NotebookTbl notebook in NotebookTbl.getAllNotebooks())

{

//notebookd ID, number of likes.

mostLikedNotebooksIds.Add(notebook.NotebookId, likedNotebooks.FindAll(x => x.NotebookId == notebook.NotebookId).Count());

}

var sortedDict = from entry in mostLikedNotebooksIds orderby entry.Value descending select entry;

List<NotebookTbl> mostLikedNotebooks = new List<NotebookTbl>();

for (int i = 0; i < number; i++)

{

if (i >= sortedDict.Count())

{

break;

}

mostLikedNotebooks.Add(NotebookTbl.GetNotebookByNotebookID(sortedDict.ElementAt(i).Key));

}

return mostLikedNotebooks;

}

/// <summary>

/// Returns the recently added notebooks.

/// </summary>

/// <param name="number">the number of notebooks to return</param>

/// <returns></returns>

public static List<NotebookTbl> RecentlyAddedNotebookList(int number)

{

List<NotebookTbl> recentlyAddedNotebooks = NotebookTbl.getAllNotebooks();

recentlyAddedNotebooks.Sort((x, y) => DateTime.Compare(y.CreatedDate, x.CreatedDate));

return (List<NotebookTbl>)recentlyAddedNotebooks.GetRange(0, number);

}

/// <summary>

/// The notebooks data list item command

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Notebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

// Triggered when clicking on a notebook.

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

/// <summary>

/// The most liked notebook section item command.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void MostLikeNotebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

/// <summary>

/// Returns the user to the home page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

/// <summary>

/// Searching through the different fillters in the Explore page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void SearchButton\_Click(object sender, EventArgs e)

{

List<NotebookTbl> searchedNotebooks = NotebookTbl.getAllNotebooks();

if (UserTB.Text != "")

{

searchedNotebooks = NotebookTbl.getNotebooksFromListAndVar(searchedNotebooks, 0, UserTB.Text);

}

if (SubjectsDropdownlist.SelectedValue != "-- Select Subject --")

{

searchedNotebooks = NotebookTbl.getNotebooksFromListAndVar(searchedNotebooks, 1, SubjectsDropdownlist.SelectedValue);

}

if (SchoolsDropDownList.SelectedValue != "-- Select School --")

{

searchedNotebooks = NotebookTbl.getNotebooksFromListAndVar(searchedNotebooks, 2, SchoolsDropDownList.SelectedValue);

}

if (DescripionTB.Text != "")

{

searchedNotebooks = NotebookTbl.getNotebooksFromListAndVar(searchedNotebooks, 3, DescripionTB.Text);

}

if (searchedNotebooks != null)

{

if (searchedNotebooks.Count == 0)

{

Notebooks.Visible = false;

NotFoundLabel.Visible = true;

}

else

{

if (user.Permission == "admin")

{

Notebooks.DataSource = searchedNotebooks;

}

else

{

Notebooks.DataSource = searchedNotebooks.FindAll(x => x.Accessibility == "public" || x.UserId == user.UserId);

}

Notebooks.DataBind();

NotFoundLabel.Visible = false;

Notebooks.Visible = true;

}

}

else

{

Notebooks.Visible = false;

NotFoundLabel.Visible = true;

}

}

}

## Home.aspx

<style>

.home-btn {

height: 25vh;

width: 25vh;

font-size: 40px;

margin-top: 5vh;

transition: 0.5s;

}

.home-btn:hover {

height: 27vh;

width: 27vh;

}

</style>

<asp:Panel runat="server" ID="loggedInAsPanel" Visible="false">

<h4>you are logged in as:<asp:Label Text="" ID="LoggedAsUserLabel" runat="server" /></h4>

<asp:Button Text="Log out" CssClass="btn btn-danger" runat="server" ID="LogOutUseruButton" OnClick="LogOutUseruButton\_Click" />

</asp:Panel>

<center>

<br />

<h1>

<asp:Label ID="WelcomeLBL" runat="server" Text="Welcome"></asp:Label></h1>

<br />

<asp:Button class="home-btn" runat="server" ID="AdminPanelButton" Text="Admin Panel" Visible="false" OnClick="AdminPanelButton\_Click" Style="font-size: 30px;"></asp:Button>

<asp:Button class="home-btn" runat="server" ID="WebsiteStats" Text="Website stats" Visible="false" OnClick="WebsiteStats\_Click" Style="font-size: 30px;"></asp:Button>

<style>

.buttons-container {

display: flex;

justify-content: center;

align-items: center;

gap: 10px;

padding: 0px 50px;

}

.home-btn {

flex-basis: 20%; /\* set the initial size of the buttons \*/

flex-shrink: 0; /\* prevent the buttons from shrinking \*/

}

</style>

<div class="buttons-container">

<button class="home-btn" style="font-size: 30px;">🔍 &#13;&#10;<a href="Explore.aspx">explore</a></button>

<button class="home-btn" style="font-size: 30px;"><a href="UserInfo.aspx">personal information</a></button>

<button class="home-btn" style="font-size: 30px;"><a href="UserNotebooks.aspx" style="font-size: 30px;">your notebooks</a></button>

<button class="home-btn" style="font-size: 30px;">❤️ &#13;&#10;<a href="LikedNotebooks.aspx">liked notebooks</a></button>

<button class="home-btn" style="font-size: 30px;">➕&#13;&#10;<a href="CreateNotebook.aspx">new notebook</a></button>

</div>

<br />

<br />

<br />

## [Home.aspx.cs](http://Home.aspx.cs)

public partial class WebForm1 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

loggedInAsPanel.Visible = false;

UserTbl user = (UserTbl)Session["User"];

if (!IsPostBack)

{

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

WelcomeLBL.Text = "Welcome " + user.FirstName + "✨";

}

if (user.Permission == "admin")

{

AdminPanelButton.Visible = true;

WebsiteStats.Visible = true;

if (Session["UserFromAdmin"] != null)

{

UserTbl loggedInUser = (UserTbl)Session["UserFromAdmin"];

loggedInAsPanel.Visible = true;

LoggedAsUserLabel.Text = loggedInUser.Username;

}

}

}

/// <summary>

/// Redirecting to the admin panel.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void AdminPanelButton\_Click(object sender, EventArgs e)

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect("AdminPanel.aspx");

}

/// <summary>

/// Logout user button click.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void LogOutUseruButton\_Click(object sender, EventArgs e)

{

loggedInAsPanel.Visible = false;

Session["UserFromAdmin"] = null;

}

/// <summary>

/// Redirecting to the website stats page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void WebsiteStats\_Click(object sender, EventArgs e)

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect("WebsiteStats.aspx");

}

}

## LikedNotebooks.aspx

<style>

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<center>

<div class="animate\_\_animated animate\_\_rubberBand">

<h3>Your Liked Notebooks</h3>

<br />

<br />

<style>

.data-list-item {

width: 150px;

height: 150px;

font-size: 10px;

transition: 0.4s;

}

.data-list-item:hover {

width: 180px;

height: 180px;

font-size: 12px;

}

</style>

<asp:DataList ID="Notebooks" runat="server" Height="117px" OnItemCommand="Notebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

</div>

</center>

## LikedNotebooks.aspx.cs

public partial class LikedNotebooks : System.Web.UI.Page

{

//the current user.

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

List<NotebookTbl> likedNotebooks = user.GetLikedNotebooks();

Notebooks.DataSource = likedNotebooks;

Notebooks.DataBind();

}

if (user.Permission == "admin")

{

if (Session["UserFromAdmin"] != null)

{

user = ((UserTbl)Session["UserFromAdmin"]);

List<NotebookTbl> likedNotebooks = user.GetLikedNotebooks();

Notebooks.DataSource = likedNotebooks;

Notebooks.DataBind();

}

}

}

/// <summary>

/// Notebooks data list item command.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Notebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

/// <summary>

/// Returns to the home screen.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

}

## Login.aspx

<center>

<div class="animate\_\_animated animate\_\_rubberBand">

<h1>Login</h1>

<br />

<asp:TextBox runat="server" ID="UsernameTB" CssClass="login-textbox" placeholder="Username" /><br />

<br />

<asp:TextBox runat="server" ID="PasswordTB" CssClass="login-textbox" placeholder="Password" type="password" /><br />

<br />

<asp:Button ID="LoginBTN" runat="server" Text="Login" CssClass="btn btn-danger login-btn" OnClick="LoginBTN\_Click" /><br />

<asp:Label ID="lblMessage" runat="server" Text="" CssClass="error"></asp:Label><br />

<a href="Register.aspx">dont have an account?</a>

</div>

</center>

## Login.aspx.cs

public partial class Login : System.Web.UI.Page

{

/// <summary>

/// Logs in the user.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void LoginBTN\_Click(object sender, EventArgs e)

{

string username = UsernameTB.Text;

string password = PasswordTB.Text;

UserTbl user = UserTbl.UserLogin(username, password);

if (user == null)

{

lblMessage.Text = "login unsuccessfull";

}

else

{

Session["User"] = user;

Response.Redirect("home.aspx");

}

}

}

## Register.aspx

<center>

<div <%--class="animate\_\_animated animate\_\_rubberBand"--%>>

<h1>Register</h1>

<br />

<asp:TextBox runat="server" ID="FirstNameTB" CssClass="login-textbox" placeholder="First name" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorFN" runat="server" ErrorMessage="Required Field!" ControlToValidate="FirstNameTB"></asp:RequiredFieldValidator>

<br />

<asp:TextBox runat="server" ID="LastNameTB" CssClass="login-textbox" placeholder="Last name" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorLN" runat="server" ErrorMessage="Required Field!" ControlToValidate="LastNameTB"></asp:RequiredFieldValidator>

<br />

<asp:TextBox runat="server" ID="UsernameTB" CssClass="login-textbox" placeholder="Username" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorU" runat="server" ErrorMessage="Required Field!" ControlToValidate="UsernameTB"></asp:RequiredFieldValidator>

<br />

<br />

<asp:TextBox runat="server" ID="AdressTB" CssClass="login-textbox" placeholder="Address" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorAdd" runat="server" ErrorMessage="Required Field!" ControlToValidate="AdressTB"></asp:RequiredFieldValidator>

<br />

<p>birthday:</p>

<asp:TextBox ID="from\_date" runat="server" placeholder="From" type="date" style="font-size:20px;" ></asp:TextBox>

<script>

// Get the birthday input element

const birthdayInput = document.getElementById("ContentPlaceHolder1\_from\_date");

// Calculate the minimum and maximum allowed dates

const maxDate = new Date().toISOString().split("T")[0];

const minDate = new Date();

minDate.setFullYear(minDate.getFullYear() - 90);

const minDateISO = minDate.toISOString().split("T")[0];

// Set the minimum and maximum attributes on the input element

birthdayInput.setAttribute("min", minDateISO);

birthdayInput.setAttribute("max", maxDate);

</script>

<asp:RequiredFieldValidator ID="RequiredFieldValidatorBD" runat="server" ErrorMessage="Required Field!" ControlToValidate="from\_date"></asp:RequiredFieldValidator>

<br />

<br />

<p>Your password should contain minimum eight characters, at least one letter and one number</p>

<asp:TextBox runat="server" ID="PasswordTB" CssClass="login-textbox" placeholder="Password" type="password" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorPass" runat="server" ErrorMessage="Required Field!" ControlToValidate="PasswordTB"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="RegularExpressionValidatorPasswordRegex" runat="server" ErrorMessage="Password not in format" ControlToValidate="PasswordTB" ValidationExpression="^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$"></asp:RegularExpressionValidator>

<br />

<asp:TextBox runat="server" ID="PasswordTB2" CssClass="login-textbox" placeholder="re-enter password" type="password" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorPass2" runat="server" ErrorMessage="Required Field!" ControlToValidate="PasswordTB2"></asp:RequiredFieldValidator>

<br />

<h2>Choose which schools you are in:</h2>

<asp:DataList ID="SchoolsDataList" runat="server" Height="117px" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<asp:RadioButton Text='<%# Eval("Name") %>' runat="server" ID="CheckBox" AutoPostBack="true" />

</ItemTemplate>

</asp:DataList>

<asp:Button ID="RegisterBTN" runat="server" Text="Register" CssClass="btn btn-danger login-btn" OnClick="RegisterBTN\_Click" /><br />

<asp:Label ID="lblMessage" runat="server" Text="" CssClass="error"></asp:Label>

<a href="Login.aspx">already have an account?</a>

</div>

</center>

## Register.aspx.cs

public partial class WebForm2 : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

List<SchoolTbl> schoolsList = SchoolTbl.GetSchools();

SchoolsDataList.DataSource = schoolsList;

SchoolsDataList.DataBind();

}

ValidationSettings.UnobtrusiveValidationMode = UnobtrusiveValidationMode.None;

}

/// <summary>

/// Registers the user.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void RegisterBTN\_Click(object sender, EventArgs e)

{

string firstName = FirstNameTB.Text;

string lastName = LastNameTB.Text;

string password = PasswordTB.Text;

DateTime birthday = Convert.ToDateTime(from\_date.Text);

List<SchoolTbl> schools = new List<SchoolTbl>();

foreach (DataListItem item in SchoolsDataList.Items)

{

RadioButton chkSmall = (RadioButton)item.FindControl("CheckBox");

if (chkSmall.Checked)

{

schools.Add(SchoolTbl.GetSchoolByName(chkSmall.Text));

}

}

string address = AdressTB.Text;

string username = UsernameTB.Text;

if (PasswordTB.Text != PasswordTB2.Text)

{

lblMessage.Text = "Password doesn't match";

return;

}

UserTbl user = UserTbl.Register(username, password, firstName, lastName, birthday, schools, address);

if (user == null) lblMessage.Text = "User already exist";

else

{

Session["user"] = user;

Response.Redirect("home.aspx");

}

}

}

## UserInfo.aspx

<style>

.backbtn {

margin-left: 20px;

}

.personal-info {

border-style: solid;

margin-top: 30px;

width: 50vh;

}

.personal-info-header {

text-align: center;

}

</style>

<style>

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" CausesValidation="false" /><br />

<center>

<asp:Panel runat="server" ID="UserInfoPanel">

<div id="personal-info" class="personal-info">

<h3 class="personal-info-header">Personal information</h3>

<div style="font-size: larger">

<p>

name:

<asp:Label ID="NameLBL" runat="server" Text=""></asp:Label>

</p>

<p>

password:

<asp:Label ID="PassLBL" runat="server" Text=""></asp:Label>

</p>

<p>

birthday:

<asp:Label ID="BDLBL" runat="server" Text=""></asp:Label>

</p>

<p>

schools:

<asp:Label ID="SchoolLBL" runat="server" Text=""></asp:Label>

</p>

<p>

address:

<asp:Label ID="AdressLBL" runat="server" Text=""></asp:Label>

</p>

<p>

username:

<asp:Label ID="UsernameLBL" runat="server" Text=""></asp:Label>

</p>

</div>

</div>

</asp:Panel>

<br />

<asp:Button Text="Edit Info" runat="server" ID="EditButton" OnClick="EditButton\_Click" class="btn btn-primary" /><br />

<br />

<asp:Panel runat="server" ID="EditUserPanel" Visible="false">

<asp:TextBox runat="server" ID="FirstNameTB" CssClass="login-textbox" placeholder="First name" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorFN" runat="server" ErrorMessage="Required Field!" ControlToValidate="FirstNameTB"></asp:RequiredFieldValidator>

<br />

<asp:TextBox runat="server" ID="LastNameTB" CssClass="login-textbox" placeholder="Last name" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorLN" runat="server" ErrorMessage="Required Field!" ControlToValidate="LastNameTB"></asp:RequiredFieldValidator>

<br />

<asp:TextBox runat="server" ID="UsernameTB" CssClass="login-textbox" placeholder="Username" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorU" runat="server" ErrorMessage="Required Field!" ControlToValidate="UsernameTB"></asp:RequiredFieldValidator>

<br />

<br />

<asp:TextBox runat="server" ID="AdressTB" CssClass="login-textbox" placeholder="Address" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorAdd" runat="server" ErrorMessage="Required Field!" ControlToValidate="AdressTB"></asp:RequiredFieldValidator>

<br />

<p>birthday:</p>

<asp:TextBox ID="from\_date" runat="server" placeholder="From" type="date"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidatorBD" runat="server" ErrorMessage="Required Field!" ControlToValidate="from\_date"></asp:RequiredFieldValidator>

<br />

<br />

<p>Your password should contain minimum eight characters, at least one letter and one number</p>

<asp:TextBox runat="server" ID="PasswordTB" CssClass="login-textbox" placeholder="Password" type="password" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorPass" runat="server" ErrorMessage="Required Field!" ControlToValidate="PasswordTB"></asp:RequiredFieldValidator>

<asp:RegularExpressionValidator ID="RegularExpressionValidatorPasswordRegex" runat="server" ErrorMessage="Password not in format" ControlToValidate="PasswordTB" ValidationExpression="^(?=.\*[A-Za-z])(?=.\*\d)[A-Za-z\d]{8,}$"></asp:RegularExpressionValidator>

<br />

<asp:TextBox runat="server" ID="PasswordTB2" CssClass="login-textbox" placeholder="re-enter password" type="password" /><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidatorPass2" runat="server" ErrorMessage="Required Field!" ControlToValidate="PasswordTB2"></asp:RequiredFieldValidator>

<br />

<asp:DataList ID="SchoolsDataList" runat="server" Height="40px" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<asp:RadioButton Text='<%# Eval("Name") %>' runat="server" ID="CheckBox" AutoPostBack="true" />

</ItemTemplate>

</asp:DataList>

<asp:Button ID="RegisterBTN" runat="server" Text="submit" CssClass="btn btn-danger login-btn" OnClick="RegisterBTN\_Click" />

<asp:Button ID="CancelButton" runat="server" Text="cancel" CausesValidation="false" CssClass="btn btn-warning login-btn" OnClick="CancelButton\_Click" /><br />

<asp:Label ID="lblMessage" runat="server" Text="" CssClass="error"></asp:Label>

<br />

</asp:Panel>

</center>

## UserInfo.aspx.cs

public partial class profile : System.Web.UI.Page

{

// The current user.

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

if (user.Permission == "admin")

{

string desiredValue = "";

foreach (string item in HttpContext.Current.Request.Url.Query.Split('&'))

{

string[] parts = item.Replace("?", "").Split('=');

if (parts[0] == "id")

{

desiredValue = parts[1];

break;

}

}

if (Session["UserFromAdmin"] != null)

{

desiredValue = ((UserTbl)Session["UserFromAdmin"]).UserId.ToString();

}

if (desiredValue != "")

{

if (UserTbl.GetUserByUserId(int.Parse(desiredValue)) != null)

{

user = UserTbl.GetUserByUserId(int.Parse(desiredValue));

}

}

}

NameLBL.Text = user.FirstName + " " + user.LastName;

PassLBL.Text = user.Password;

BDLBL.Text = user.Birthday.ToString("dd/MM/yyyy");

SchoolLBL.Text = "";

foreach (SchoolTbl school in UserInSchoolTbl.GetSchoolsByUserId(user.UserId))

{

SchoolLBL.Text += school.Name + ", ";

}

if (SchoolLBL.Text.Length >= 2)

{

SchoolLBL.Text = SchoolLBL.Text.Substring(0, SchoolLBL.Text.Length - 2);

}

AdressLBL.Text = user.Address;

UsernameLBL.Text = user.Username;

if (!IsPostBack)

{

List<SchoolTbl> schoolsList = SchoolTbl.GetSchools();

SchoolsDataList.DataSource = schoolsList;

SchoolsDataList.DataBind();

InitUpdatePanel();

}

ValidationSettings.UnobtrusiveValidationMode = UnobtrusiveValidationMode.None;

}

## UserNotebooks.aspx

<style>

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<center>

<div class="animate\_\_animated animate\_\_rubberBand">

<h3>

<asp:Label Text="" runat="server" ID="title" /></h3>

<br />

<br />

<style>

.data-list-item {

width: 150px;

height: 150px;

font-size: 10px;

transition: 0.4s;

}

.data-list-item:hover {

width: 180px;

height: 180px;

font-size: 12px;

}

</style>

<asp:DataList ID="Notebooks" runat="server" Height="117px" OnItemCommand="Notebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

</div>

</center>

## UserNotebooks.aspx.cs

public partial class user\_notebooks : System.Web.UI.Page

{

// The current user.

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

List<NotebookTbl> userNotebooks = NotebookTbl.GetNotebooksByUserID(user.UserId);

Notebooks.DataSource = userNotebooks;

Notebooks.DataBind();

}

title.Text = $"{user.FirstName}'s notebooks";

if (user.Permission == "admin")

{

if (Session["UserFromAdmin"] != null)

{

user = ((UserTbl)Session["UserFromAdmin"]);

title.Text = $"{user.FirstName}'s notebooks";

List<NotebookTbl> userNotebooks = NotebookTbl.GetNotebooksByUserID(user.UserId);

Notebooks.DataSource = userNotebooks;

Notebooks.DataBind();

}

}

}

/// <summary>

/// Notebooks data list item command.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Notebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

}

## ViewNotebook.aspx

<!-- FontAwesome Icons -->

<link

rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css" />

<style>

.backbtn {

margin-left: 20px;

}

</style>

<style>

.home-btn {

height: 300px;

width: 300px;

font-size: 30px;

margin-top: 5vh;

transition: 0.5s;

}

.home-btn:hover {

height: 320px;

width: 320px;

}

.backbtn {

margin-left: 20px;

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<asp:Button ID="SwitchView" runat="server" class="btn btn-info" Text="Switch View" OnClick="SwitchView\_Click" Style="position: absolute; top: 29%; left: 90%;" />

<br />

<div style="position: absolute; top: 54%; left: 67%;">

<style>

.like-btn {

width: 40px;

height: 40px;

transition: 0.1s;

}

.like-btn:hover {

width: 42px;

height: 42px;

}

</style>

<center>

<asp:ImageButton ID="LikeButton" runat="server" ImageUrl="~/UI/Assets/LikeBefore.png" OnClick="LikeButton\_Click" class="like-btn" />

<asp:Label Text="0" runat="server" Style="font-size: 23px; margin-left: 2px;" ID="LikeCountLabel" />

</center>

</div>

<center>

<br />

<style>

.notebookInfo-block {

border-style: solid;

margin-top: 30px;

margin-left: 30px;

width: 60vh;

}

.notebookContent-block {

border-style: solid;

width: 90vh;

}

</style>

<div class="notebookInfo-block">

<h1>Title:

<asp:Label ID="NoteTitle" runat="server" Text=""></asp:Label></h1>

<h2>Subject:

<asp:Label ID="NotebookSubject" runat="server" Text=""></asp:Label></h2>

<h4>

<asp:Label ID="DescriptionLabel" runat="server" Text=""></asp:Label></h4>

<h4>last updated:<asp:Label ID="DateLBL" runat="server" Text=""></asp:Label></h4>

<h4>created date:<asp:Label ID="CreateDate" runat="server" Text=""></asp:Label></h4>

<h4>notebook type:<asp:Label ID="NotebookTypeLabel" runat="server" Text=""></asp:Label></h4>

<h4>notebook school:<asp:Label ID="NotebookSchoolLabel" runat="server" Text=""></asp:Label></h4>

<asp:Button ID="UpdateButton" runat="server" Text="Update Notebook" class="home-btn" OnClick="UpdateButton\_Click" Style="margin-top: 0px; margin-bottom: 5px; width: 300px; height: 40px; font-size: 18px;" />

<h4>

<h4>notebook creator:<asp:Label ID="UserLinkl" runat="server" Text=""></asp:Label></h4>

</div>

<br />

</center>

<asp:Panel ID="OnlinePanel" runat="server" Visible="false">

<asp:Panel runat="server" ID="EditView" Visible="false">

<center>

<div class="containerr">

<div class="options">

<button id="bold" type="button" class="option-button format">

<i class="fa-solid fa-bold"></i>

</button>

<button id="italic" type="button" class="option-button format">

<i class="fa-solid fa-italic"></i>

</button>

<button id="underline" type="button" class="option-button format">

<i class="fa-solid fa-underline"></i>

</button>

<button id="strikethrough" type="button" class="option-button format">

<i class="fa-solid fa-strikethrough"></i>

</button>

<button id="superscript" type="button" class="option-button script">

<i class="fa-solid fa-superscript"></i>

</button>

<button id="subscript" type="button" class="option-button script">

<i class="fa-solid fa-subscript"></i>

</button>

<!-- List -->

<button id="insertOrderedList" type="button" class="option-button">

<div class="fa-solid fa-list-ol"></div>

</button>

<button id="insertUnorderedList" type="button" class="option-button">

<i class="fa-solid fa-list"></i>

</button>

<!-- Undo/Redo -->

<button id="undo" type="button" class="option-button">

<i class="fa-solid fa-rotate-left"></i>

</button>

<button id="redo" type="button" class="option-button">

<i class="fa-solid fa-rotate-right"></i>

</button>

<!-- Link -->

<button id="createLink" type="button" class="adv-option-button">

<i class="fa fa-link"></i>

</button>

<button id="unlink" type="button" class="option-button">

<i class="fa fa-unlink"></i>

</button>

<!-- Alignment -->

<button id="justifyLeft" type="button" class="option-button align">

<i class="fa-solid fa-align-left"></i>

</button>

<button id="justifyCenter" type="button" class="option-button align">

<i class="fa-solid fa-align-center"></i>

</button>

<button id="justifyRight" type="button" class="option-button align">

<i class="fa-solid fa-align-right"></i>

</button>

<button id="justifyFull" type="button" class="option-button align">

<i class="fa-solid fa-align-justify"></i>

</button>

<button id="indent" type="button" class="option-button spacing">

<i class="fa-solid fa-indent"></i>

</button>

<button id="outdent" type="button" class="option-button spacing">

<i class="fa-solid fa-outdent"></i>

</button>

<!-- Headings -->

<select id="formatBlock" class="adv-option-button">

<option value="H1">H1</option>

<option value="H2">H2</option>

<option value="H3">H3</option>

<option value="H4">H4</option>

<option value="H5">H5</option>

<option value="H6">H6</option>

</select>

<!-- Font -->

<select id="fontName" type="button" class="adv-option-button"></select>

<select id="fontSize" class="adv-option-button"></select>

<!-- Color -->

<div class="input-wrapper">

<input type="color" id="foreColor" class="adv-option-button" />

<label for="foreColor">Font Color</label>

</div>

<div class="input-wrapper">

<input type="color" id="backColor" class="adv-option-button" />

<label for="backColor">Highlight Color</label>

</div>

</div>

<div runat="server" id="textinput" contenteditable="true" onchange="HtmlChange()" style="overflow-y: scroll; height: 400px;">

</div>

<script src="editor.js"></script>

<!-- Stylesheet -->

<link rel="stylesheet" href="editor.css" />

<br />

<button type="button" onclick="sendData()" class="btn btn-success" style="font-size: 20px; width: 80px; height: 40px;">Save</button>

<script>

function get(name) {

if (name = (new RegExp('[?&]' + encodeURIComponent(name) + '=([^&]\*)')).exec(location.search))

return decodeURIComponent(name[1]);

}

function sendData() {

var xhr = new XMLHttpRequest();

xhr.open("POST", "https://localhost:44326/UI/ViewNotebook.aspx", true);

xhr.setRequestHeader('Content-Type', 'application/json');

xhr.setRequestHeader('mytext', document.getElementById("ContentPlaceHolder2\_textinput").innerHTML);

xhr.setRequestHeader('notebookId', get("id"));

xhr.send(JSON.stringify({

"mytext": document.getElementById("ContentPlaceHolder2\_textinput").innerHTML,

}));

}

</script>

</div>

</center>

</asp:Panel>

<asp:Panel runat="server" ID="CantEditPanel" Visible="true" DefaultButton="">

<center>

<div runat="server" id="CantEditDiv" class="notebookContent-block" style="overflow-y: scroll; height: 60vmin; width: 100vmin;">

</div>

</center>

<br />

<br />

</asp:Panel>

<!--Script-->

</asp:Panel>

<asp:Panel runat="server" ID="DocumentPanel" Visible="false">

<center>

<div style="display: inline-flex; gap: 100px">

<asp:Button ID="DownloadButton" runat="server" Text="Download Notebook" class="home-btn" OnClick="DownloadButton\_Click" />

</div>

</center>

<h3>

<asp:Label Text="" runat="server" ID="ErrorLabel" /></h3>

</asp:Panel>

<style>

.comments-section {

position: absolute;

top: 55vmin;

left: 10vmin;

width: 36vmin;

height: 80vmin;

border-style: solid;

overflow-y: scroll;

}

.add-comment-block {

position: absolute;

top: 40vmin;

left: 10vmin;

width: 40vmin;

height: 12vmin;

border-style: solid;

}

</style>

<asp:Panel runat="server" ID="CommentPanel" DefaultButton="AddCommentBTN">

<div class="comments-section" style="padding: 10px;">

<asp:DataList runat="server" ID="CommentsDataList" OnItemCommand="Comments\_ItemCommand" OnItemDataBound="CommentsDataList\_ItemDataBound">

<ItemTemplate>

<div class="panel panel-default" style="width: 250px">

<div class="panel-heading" style="display: flex; justify-content: space-between">

<div>

<b><%# NoteShare.Models.UserTbl.GetUserByUserId((int)Eval("UserId")).Username %> </b>

<p style="padding: 0; margin: 0; font-size: 12px"><%# ((DateTime)Eval("CreatedDate")).ToString("dd/MM/yyyy") %></p>

</div>

<div>

<asp:LinkButton runat="server" Visible="false" ID="DeleteCommentButton" class="btn btn-danger px-3" CommandName="DeleteUser" CommandArgument='<%#Eval("CommentId") %>'>

<i class="fa fa-trash" aria-hidden="true"></i>

</asp:LinkButton>

</div>

</div>

<div class="panel-body"><%# Eval("Comment") %></div>

</div>

</ItemTemplate>

</asp:DataList>

</div>

<div class="add-comment-block">

<center>

<h4>Add a comment</h4>

<div style="display: inline-flex; gap: 5px;">

<asp:TextBox runat="server" Style="height: 5vmin; width: 30vmin; font-size: 20px; overflow: auto;" ID="CommentTextBox" TextMode="multiline" />

<asp:Button Text="add" runat="server" ID="AddCommentBTN" OnClick="AddCommentBTN\_Click" />

</div>

</center>

</div>

</asp:Panel>

<asp:Button Text="Delete Notebook" CssClass="btn btn-danger" ID="DeleteNotebookButton" OnClick="DeleteNotebookButton\_Click" runat="server" />

## ViewNotebook.aspx.cs

public partial class ViewNotebook : System.Web.UI.Page

{

public static UserTbl user;

public static NotebookTbl notebook;

protected void Page\_Load(object sender, EventArgs e)

{

//handling online notebook update

//this if is called when the Save button is clicked.

if (Request.HttpMethod == "POST")

{

var notebookHtmlText = Request.Headers.Get("mytext");

var notebookId = Request.Headers.Get("notebookId");

if (notebookId != null && notebookHtmlText != null && notebookId != "" && notebookHtmlText != "")

{

notebook = NotebookTbl.GetNotebookByNotebookID(int.Parse(notebookId));

NotebookTbl.UpdateNotebookHtml(notebookHtmlText, notebook);

CantEditDiv.InnerHtml = notebook.OnlineNotebookFormat;

}

}

if (!IsPostBack)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

string desiredValue = "";

foreach (string item in HttpContext.Current.Request.Url.Query.Split('&'))

{

string[] parts = item.Replace("?", "").Split('=');

if (parts[0] == "id")

{

desiredValue = parts[1];

break;

}

}

//update mode

if (desiredValue != "")

{

notebook = NotebookTbl.GetNotebookByNotebookID(Int32.Parse(desiredValue));

NoteTitle.Text = notebook.Title;

DescriptionLabel.Text = notebook.Description;

DateLBL.Text = notebook.UpdateDate.ToString("dd/MM/yyyy");

CreateDate.Text = notebook.CreatedDate.ToString("dd/MM/yyyy");

NotebookTypeLabel.Text = notebook.Accessibility;

NotebookSubject.Text = SubjectTbl.GetSubjectByID(notebook.SubjectId).Name;

if(SchoolTbl.GetSchoolByID((int)notebook.SchoolId) != null)

{

NotebookSchoolLabel.Text = SchoolTbl.GetSchoolByID((int)notebook.SchoolId).Name;

}

UserLinkl.Text = UserTbl.GetUserByUserId(notebook.UserId).Username;

if (notebook.Format == "document")

{

DocumentPanel.Visible = true;

}

if (notebook.Format == "online")

{

OnlinePanel.Visible = true;

textinput.InnerHtml = notebook.OnlineNotebookFormat;

CantEditDiv.InnerHtml = notebook.OnlineNotebookFormat;

}

}

if (user.UserId != notebook.UserId)

{

UpdateButton.Visible = false;

SwitchView.Visible = false;

DeleteNotebookButton.Visible = false;

}

else

{

UpdateButton.Visible = true;

SwitchView.Visible = true;

DeleteNotebookButton.Visible = true;

}

if (user.Permission == "admin")

{

UpdateButton.Visible = true;

SwitchView.Visible = true;

DeleteNotebookButton.Visible = true;

}

if (notebook.Format != "online")

{

OnlinePanel.Visible = false;

}

if (notebook.Format == "document")

{

SwitchView.Visible = false;

}

}

if (LikeTbl.IsLiked(user.UserId, notebook.NotebookId))

{

LikeButton.ImageUrl = "~/UI/Assets/LikeAfter.png";

}

LikeCountLabel.Text = "" + notebook.GetLikeCount();

SwitchView.Text = "Switch To Edit View";

List<CommentTbl> list = new List<CommentTbl>(CommentTbl.GetCommentsByNotebookId(notebook.NotebookId));

list.Reverse();

CommentsDataList.DataSource = list;

CommentsDataList.DataBind();

}

protected void BackBTN\_Click(object sender, EventArgs e)

{

if (Session["url"].ToString() != null)

{

Response.Redirect(Session["url"].ToString());

}

else

{

Response.Redirect("Home.aspx");

}

}

/// <summary>

/// Downloads the notebook file.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void DownloadButton\_Click(object sender, EventArgs e)

{

string filePath = notebook.Path;

if (filePath == null || filePath == "")

{

ErrorLabel.Text = "no file detected, try uploading a new one by clicking update notebook";

return;

}

FileInfo file = new FileInfo(Server.MapPath(filePath));

if (file.Exists)

{

// Clear Rsponse reference

Response.Clear();

// Add header by specifying file name

Response.AddHeader("Content-Disposition", "attachment; filename=" + file.Name);

// Add header for content length

Response.AddHeader("Content-Length", file.Length.ToString());

// Specify content type

Response.ContentType = "text/plain";

// Clearing flush

Response.Flush();

// Transimiting file

Response.TransmitFile(file.FullName);

Response.End();

}

else

{

ErrorLabel.Text = "Requested file is not available to download(Error in server)";

}

}

/// <summary>

/// Redirecting to the update notebook page.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void UpdateButton\_Click(object sender, EventArgs e)

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"CreateNotebook.aspx?id={notebook.NotebookId}");

}

/// <summary>

/// Switches views between the edit view, which the user can edit the notebook, and the normal view.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void SwitchView\_Click(object sender, EventArgs e)

{

if (EditView.Visible == false)

{

EditView.Visible = true;

CantEditDiv.Visible = false;

SwitchView.Text = "Switch To Normal View";

CommentPanel.Visible = false;

}

else

{

EditView.Visible = false;

CantEditDiv.Visible = true;

SwitchView.Text = "Switch To Edit View";

Response.Redirect(Request.RawUrl);

CommentPanel.Visible = true;

}

}

/// <summary>

/// Likes the notebook.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void LikeButton\_Click(object sender, ImageClickEventArgs e)

{

if (notebook.IsLiked(user.UserId))

{

//unlike

notebook.UnLike(user);

LikeButton.ImageUrl = "~/UI/Assets/LikeBefore.png";

LikeCountLabel.Text = "" + notebook.GetLikeCount();

}

else

{

//like

notebook.Like(user);

LikeButton.ImageUrl = "~/UI/Assets/LikeAfter.png";

LikeCountLabel.Text = "" + notebook.GetLikeCount();

}

}

/// <summary>

/// Deletes the notebook.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void DeleteNotebookButton\_Click(object sender, EventArgs e)

{

notebook.DeleteNotebook();

Response.Redirect("Home.aspx");

}

/// <summary>

/// Adds a comment to the notebook.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void AddCommentBTN\_Click(object sender, EventArgs e)

{

string comment = CommentTextBox.Text;

notebook.AddComment(user, comment);

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

/// <summary>

/// Comments item command, from here you can delete a comment.

/// </summary>

/// <param name="source"></param>

/// <param name="e"></param>

protected void Comments\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "DeleteUser")

{

CommentTbl.DeleteComment(int.Parse((String)e.CommandArgument));

Page.Response.Redirect(Page.Request.Url.ToString(), true);

}

}

/// <summary>

/// Comments data bound.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

protected void CommentsDataList\_ItemDataBound(object sender, DataListItemEventArgs e)

{

if (e.Item.ItemType == ListItemType.Item ||

e.Item.ItemType == ListItemType.AlternatingItem)

{

System.Web.UI.WebControls.LinkButton BT = e.Item.FindControl("DeleteCommentButton") as System.Web.UI.WebControls.LinkButton;

int commentID = int.Parse(BT.CommandArgument);

if ((user.Permission == "admin"))

{

BT.Visible = true;

}

if (user.UserId == notebook.UserId)

{

BT.Visible = true;

}

if (user.UserId == CommentTbl.GetUserThatCommented(commentID).UserId)

{

BT.Visible = true;

}

}

}

}

## WebsiteStats.aspx

<style>

.backbtn {

margin-left: 20px;

}

</style>

<!-- Font Awesome -->

<link

href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"

rel="stylesheet" />

<!-- Google Fonts -->

<link

href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"

rel="stylesheet" />

<!-- MDB -->

<link

href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/6.1.0/mdb.min.css"

rel="stylesheet" />

<!-- MDB -->

<script

type="text/javascript"

src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/6.1.0/mdb.min.js"></script>

<style>

.jumbotron {

padding-top: 30px;

padding-bottom: 30px;

margin-bottom: 30px;

color: inherit;

background-color: #eee

}

.jumbotron .h1, .jumbotron h1 {

color: inherit

}

.jumbotron p {

margin-bottom: 15px;

font-size: 21px;

font-weight: 200

}

.jumbotron > hr {

border-top-color: #d5d5d5

}

.container .jumbotron, .container-fluid .jumbotron {

padding-right: 15px;

padding-left: 15px;

border-radius: 6px

}

.jumbotron .container {

max-width: 100%

}

@media screen and (min-width:768px) {

.jumbotron {

padding-top: 48px;

padding-bottom: 48px

}

.container .jumbotron, .container-fluid .jumbotron {

padding-right: 60px;

padding-left: 60px

}

.jumbotron .h1, .jumbotron h1 {

font-size: 63px

}

}

</style>

<asp:Button ID="BackBTN" runat="server" Text="Back" OnClick="BackBTN\_Click" class="backbtn btn btn-warning" /><br />

<center>

<h1>Website statistics</h1>

<div class="container">

<div class="row">

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-secondary">

<div class="card-body bg-secondary text-white">

<div class="row">

<div class="col-3">

<i class="fa fa-user-graduate fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="text" runat="server" ID="NumberOfUsersLabel" /></h1>

<h4>Users</h4>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-danger">

<div class="card-body bg-danger text-white">

<div class="row">

<div class="col-3">

<i class="fa fa-book fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="text" runat="server" ID="NumberOfNotebooksLabel" /></h1>

<h4>Notebooks</h4>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-warning">

<div class="card-body bg-warning text-white">

<div class="row">

<div class="col-3">

<i class="fa fa-university fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="text" runat="server" ID="NumberOfSchoolsLabel" /></h1>

<h4>Schools</h4>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-primary">

<div class="card-body bg-primary text-white">

<div class="row">

<div class="col-3">

<i class="fa-solid fa-bars fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="text" runat="server" ID="NumberOfSubjectsLabel" /></h1>

<h4>Subjects</h4>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-secondary">

<div class="card-body bg-secondary text-white">

<div class="row">

<div class="col-3">

<i class="fa-solid fa-lock fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="1" runat="server" ID="NumberOfAdminsLabel" /></h1>

<h4>Admins</h4>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-4 col-md-6" style="margin-top: 20px">

<div class="card border-danger">

<div class="card-body bg-danger text-white">

<div class="row">

<div class="col-3">

<i class="fa-solid fa-heart fa-5x"></i>

</div>

<div class="col-9 text-right">

<h1>

<asp:Label Text="1" runat="server" ID="NumberOfLikesLabel" /></h1>

<h4>Likes</h4>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<br />

<br />

<br />

<div style="display: inline-flex; gap: 50px;justify-content:center">

<div>

<h1>Notebooks Statistics</h1>

<div style="display: flex; gap: 150px">

<div>

<h3 style="margin-left:70px;">Notebooks format</h3>

<canvas id="pieChart5"></canvas>

</div>

<div>

<h3 style="margin-left:70px;">Notebooks Accesibilty</h3>

<canvas id="pieChart6"></canvas>

</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.min.js"></script>

<link href="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.4/Chart.css" rel="stylesheet" />

<script>

// MAKE ROUNDED EDGES FOR PIE CHARTS

Chart.defaults.RoundedDoughnut = Chart.helpers.clone(Chart.defaults.doughnut);

Chart.controllers.RoundedDoughnut = Chart.controllers.doughnut.extend({

draw: function (ease) {

var ctx = this.chart.ctx;

var easingDecimal = ease || 1;

var arcs = this.getMeta().data;

Chart.helpers.each(arcs, function (arc, i) {

arc.transition(easingDecimal).draw();

var pArc = arcs[i === 0 ? arcs.length - 1 : i - 1];

var pColor = pArc.\_view.backgroundColor;

var vm = arc.\_view;

var radius = (vm.outerRadius + vm.innerRadius) / 2;

var thickness = (vm.outerRadius - vm.innerRadius) / 2;

var startAngle = Math.PI - vm.startAngle - Math.PI / 2;

var angle = Math.PI - vm.endAngle - Math.PI / 2;

ctx.save();

ctx.translate(vm.x, vm.y);

ctx.fillStyle = i === 0 ? vm.backgroundColor : pColor;

ctx.beginPath();

ctx.arc(radius \* Math.sin(startAngle), radius \* Math.cos(startAngle), thickness, 0, 2 \* Math.PI);

ctx.fill();

ctx.fillStyle = vm.backgroundColor;

ctx.beginPath();

ctx.arc(radius \* Math.sin(angle), radius \* Math.cos(angle), thickness, 0, 2 \* Math.PI);

ctx.fill();

ctx.restore();

});

}

});

// EXAMPLE 5 - notebook format

var ctx5 = document.getElementById("pieChart5");

var pieChart5 = new Chart(ctx5, {

type: 'pie',

options: {

cutoutPercentage: 50,

legend: {

position: 'left',

labels: {

boxWidth: 10,

fontStyle: 'italic',

fontColor: '#aaa',

usePointStyle: true,

}

},

},

data: {

labels: [

"online",

"document",

],

datasets: [

{

data: [<%=NoteShare.Models.NotebookTbl.getAllNotebooks().FindAll(x => x.Format == "online").Count%>, <%= NoteShare.Models.NotebookTbl.getAllNotebooks().FindAll(x => x.Format == "document").Count%>],

borderWidth: 2,

borderColor: [

'#46d8d5',

"#f5e132",

],

backgroundColor: [

'rgba(70, 215, 212, 0.22)',

"rgba(245, 225, 50, 0.23)",

],

hoverBackgroundColor: [

'#46d8d5',

"#f5e132",

]

}]

}

});

// EXAMPLE 5 - notebook format

var ctx6 = document.getElementById("pieChart6");

var pieChart6 = new Chart(ctx6, {

type: 'pie',

options: {

cutoutPercentage: 50,

legend: {

position: 'left',

labels: {

boxWidth: 10,

fontStyle: 'italic',

fontColor: '#aaa',

usePointStyle: true,

}

},

},

data: {

labels: [

"private",

"public",

],

datasets: [

{

data: [<%=NoteShare.Models. NotebookTbl.getAllNotebooks().FindAll(x => x.Accessibility == "public").Count%>, <%= NoteShare.Models.NotebookTbl.getAllNotebooks().FindAll(x => x.Accessibility == "private").Count%>],

borderWidth: 2,

borderColor: [

'#46d8d5',

"#f5e132",

],

backgroundColor: [

'rgba(70, 215, 212, 0.22)',

"rgba(245, 225, 50, 0.23)",

],

hoverBackgroundColor: [

'#46d8d5',

"#f5e132",

]

}]

}

});

</script>

</div>

</div>

</div>

<br />

<br />

<div>

<h2>Most Liked Notebooks:</h2>

<style>

.data-list-item {

width: 150px;

height: 150px;

font-size: 10px;

transition: 0.4s;

}

.data-list-item:hover {

width: 180px;

height: 180px;

font-size: 12px;

}

</style>

<br />

<br />

<asp:Label Text="Notebooks in this context doesn't exist" runat="server" Visible="false" ID="Label1" />

<asp:DataList ID="MostLikedDataList" runat="server" Height="117px" OnItemCommand="MostLikeNotebooks\_ItemCommand" RepeatDirection="Horizontal" Width="268px">

<ItemTemplate>

<div class="data-list-item">

<center>

<asp:ImageButton ID="ColorIMG" class='data-list-item' ImageUrl='<%#"colors/" + Eval("Color") %>' runat="server" style="width: 150px; height: 150px;" CommandName="NotebookClick" CommandArgument='<%# Eval("NotebookId") %>' />

</center>

<h4 style="text-align: center;">

<asp:Label ID="TitleLBL" Text='<%# Eval("title") %>' runat="server" /></h4>

</div>

</ItemTemplate>

</asp:DataList>

</div>

<br />

<br />

<br />

<div>

<h2>Active users</h2>

<br />

<br />

<style>

.school-block {

border-style: solid;

margin-top: 30px;

margin-left: 30px;

width: 30vh;

}

</style>

<asp:DataList ID="ActiveUsersDataList" runat="server" Height="117px" OnItemCommand="Users\_ItemCommand" Width="268px" RepeatColumns="6">

<ItemTemplate>

<div class="school-block">

<h4><b>Username:</b><asp:Label Text='<%#Eval("Username") %>' runat="server" /></h4>

<asp:Button Text="profile" runat="server" CssClass="btn btn-primary" Style="margin-left: 10px; margin-bottom: 10px;" CommandName="UserClick" CommandArgument='<%# Eval("UserId") %>' />

<asp:Button Text="Delete" runat="server" Style="margin-left: 10px; margin-bottom: 10px;" class="btn btn-danger" CommandName="DeleteUser" CommandArgument='<%#Eval("UserId") %>' />

</div>

</ItemTemplate>

</asp:DataList>

<br />

<br />

<br />

</div>

</center>

</asp:Content>

## WebsiteStats.aspx.cs

public partial class WebsiteStats : System.Web.UI.Page

{

public static UserTbl user;

protected void Page\_Load(object sender, EventArgs e)

{

user = (UserTbl)Session["User"];

if (user == null)

{

Response.Redirect("login.aspx");

return;

}

if (user.Permission != "admin")

{

Response.Redirect("login.aspx");

return;

}

NumberOfSchoolsLabel.Text = "" + SchoolTbl.GetSchools().Count;

NumberOfUsersLabel.Text = "" + UserTbl.GetAllUsers().Count;

NumberOfNotebooksLabel.Text = "" + NotebookTbl.getAllNotebooks().Count;

NumberOfSubjectsLabel.Text = "" + SubjectTbl.GetNumberOfSubjects();

NumberOfLikesLabel.Text = "" + LikeTbl.GetAllLikes().Count;

NumberOfAdminsLabel.Text = "" + UserTbl.GetAllUsers().FindAll(x=>x.Permission == "admin").Count;

List<NotebookTbl> allNotebooks = NotebookTbl.getAllNotebooks();

List<string> subjectsList = new List<string>();

foreach (NotebookTbl notebook in allNotebooks)

{

if (notebook.Subject != null)

{

subjectsList.Add(SubjectTbl.GetSubjectByID(notebook.SubjectId).Name);

}

}

if (!IsPostBack)

{

List<LikeTbl> likedNotebooks = LikeTbl.GetAllLikes();

Dictionary<int, int> mostLikedNotebooksIds = new Dictionary<int, int>();

foreach (NotebookTbl notebook in NotebookTbl.getAllNotebooks())

{

//notebookd ID, number of likes.

mostLikedNotebooksIds.Add(notebook.NotebookId, likedNotebooks.FindAll(x => x.NotebookId == notebook.NotebookId).Count());

}

var sortedDict = from entry in mostLikedNotebooksIds orderby entry.Value descending select entry;

List<NotebookTbl> mostLikedNotebooks = new List<NotebookTbl>();

for (int i = 0; i < 10; i++)

{

if (i >= sortedDict.Count())

{

break;

}

mostLikedNotebooks.Add(NotebookTbl.GetNotebookByNotebookID(sortedDict.ElementAt(i).Key));

}

MostLikedDataList.DataSource = mostLikedNotebooks;

MostLikedDataList.DataBind();

List<UserTbl> activeUsers = UserTbl.GetAllUsers();

var sortedUsers = activeUsers.OrderBy(user => NotebookTbl.GetNumberOfNotebooksByUserId(user.UserId)).ToList();

ActiveUsersDataList.DataSource = sortedUsers.Take(10).ToList();

ActiveUsersDataList.DataBind();

}

}

protected void BackBTN\_Click(object sender, EventArgs e)

{

Response.Redirect("Home.aspx");

}

protected void MostLikeNotebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

protected void Notebooks\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "NotebookClick")

{

Session["url"] = Request.UrlReferrer.AbsoluteUri.ToString();

Response.Redirect($"ViewNotebook.aspx?id={e.CommandArgument}");

}

}

protected void Users\_ItemCommand(object source, DataListCommandEventArgs e)

{

if (e.CommandName == "DeleteUser")

{

UserTbl userToDelete = new UserTbl(int.Parse((String)e.CommandArgument));

userToDelete.Delete();

Response.Redirect(Request.RawUrl);

}

if (e.CommandName == "UserClick")

{

UserTbl user = UserTbl.GetUserByUserId(int.Parse((String)e.CommandArgument));

Session["UserFromAdmin"] = user;

Response.Redirect("Home.aspx");

}

}

}