עבודת גמר בתכנון ותכנות מערכות

במסלול שירותי רשת אינטרנט (תשפ"ה)

שם התלמיד: לירון כנר

ת"ז: XXXXXXXXX

בית ספר: כיח אליאנס – חיפה

שם המורה: ולרי קריפוקס

תוכן עניינים

Contents

מבוא

מטרות האתר

1. לספק מערכת ניהול ספרייה דיגיטלית.
2. לאפשר למשתמשים ליצור, להצטרף ולנהל ספריות.
3. לאפשר למשתמשים להוסיף, לשאול ולהחזיר ספרים.
4. לקדם מעורבות קהילתית סביב ספרים וקריאה.

קהל היעד

קהל היעד מורכב מחובבי ספרים וקוראים נלהבים, ספריות ומוסדות חינוך, מועדוני ספרים וקבוצות קריאה ואנשים המעוניינים לארגן ולשתף את אוסף הספרים האישיים שלהם.

תיאור האתר

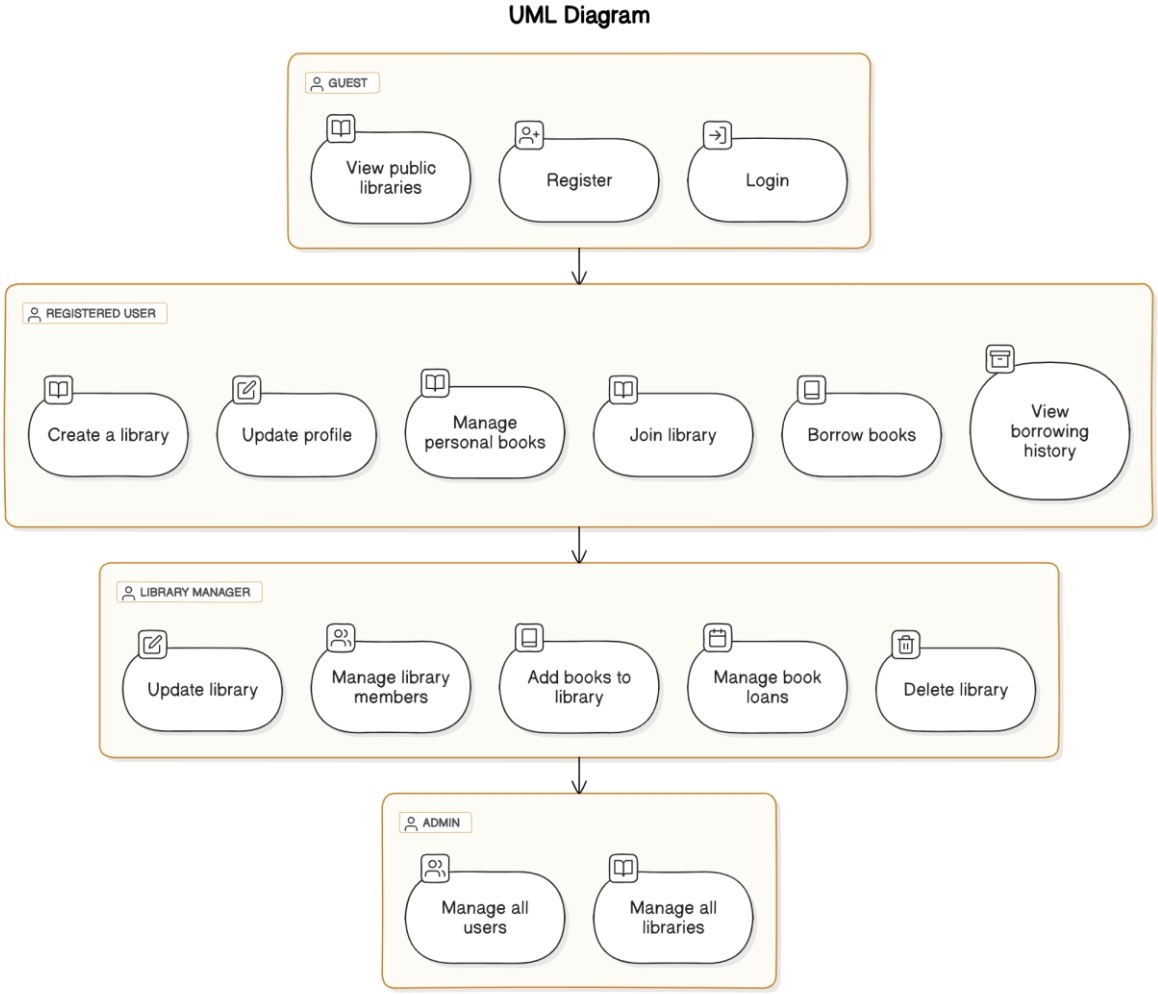
AspHarmony היא פלטפורמה מקיפה לניהול ספרייה דיגיטלית האתר מציע דברים מרובים.

בתור אורח באתר ניתן להתחבר, להרשם ולצפות ברשימת הספרים והספריות שישנן באתר.

בתור משתמש באתר ניתן לנהל את פרטיך, להתנתק, ליצור למחוק לערוך ולצפות בספרים ובנוסף ניתן ליצור, להצטרף, לשאול ספרים ולצאת מן ספריות.

בתור מנהל באתר ניתן לנהל כל ספרייה ובנוסף לנהל את המשתמשים שבאתר.

דיאגרמת UML


UML Diagram
GUEST
View public libraries
Register
Login
REGISTERED USER
Create a library
Update profile
Manage personal books
Join library
Borrow books
View borrowing history
LIBRARY MANAGER
Update library
Manage library members
Add books to library
Manage book loans
Delete library
ADMIN
Manage all users
Manage all libraries

מבנה בסיס הנתונים

בפרויקט השתמשתי בספרייה הנקראת Entity Framework כך הטבלאות במסד הנתונים נוצרות מן מחלקות בקוד.

```
public class ApplicationContext(DbContextOptions<ApplicationContext> options) : IdentityDbContext<User>(options) {
    protected override void OnModelCreating(ModelBuilder modelBuilder) {
    }

    public DbSet<Sender> Senders { get; set; }
    public DbSet<UserSender> UserSenders { get; set; }
    public DbSet<Feedback> Feedbacks { get; set; }
    public DbSet<Book> Books { get; set; }
    public DbSet<Library> Libraries { get; set; }
    public DbSet<LibraryBook> LibraryBooks { get; set; }
    public DbSet<LibraryMembership> LibraryMemberships { get; set; }
    public DbSet<BookLoan> BookLoans { get; set; }
    public DbSet<Notification> Notifications { get; set; }
}
```

המחלקה ApplicationContext יורשת מן המחלקה IdentityDbContext (מחלקה מן ASP.NET Identity) ומציינת את הקשר למבנה הנתונים ובעזרתה ניתן לבצע פעולות על המבנה נתונים.

להלן הסבר על הטבלאות:

Libraries - טבלת הספריות.

Books – טבלת הספרים.

LibraryBooks – טבלת קשר המחברת בין ספר לבין ספרייה.

BooksLoans – טבלה המתארת את ההשאלות הספרים שבספרייה.

LibraryMembership – טבלת קשר בין משתמש לבין ספרייה.

AspNetUsers – טבלת משתמשים סטנדרטים מן ASP.NET Identity.

AspNetRoles – טבלת התפקידים שבאתר.

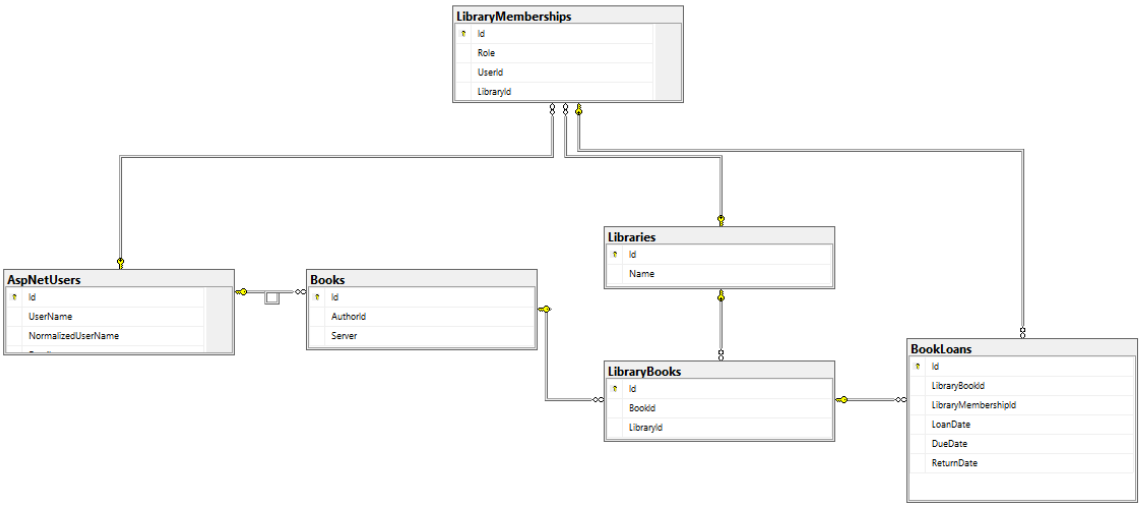
AspNetUserRoles – טבלת קשר בין משתמש לבין תפקיד באתר.

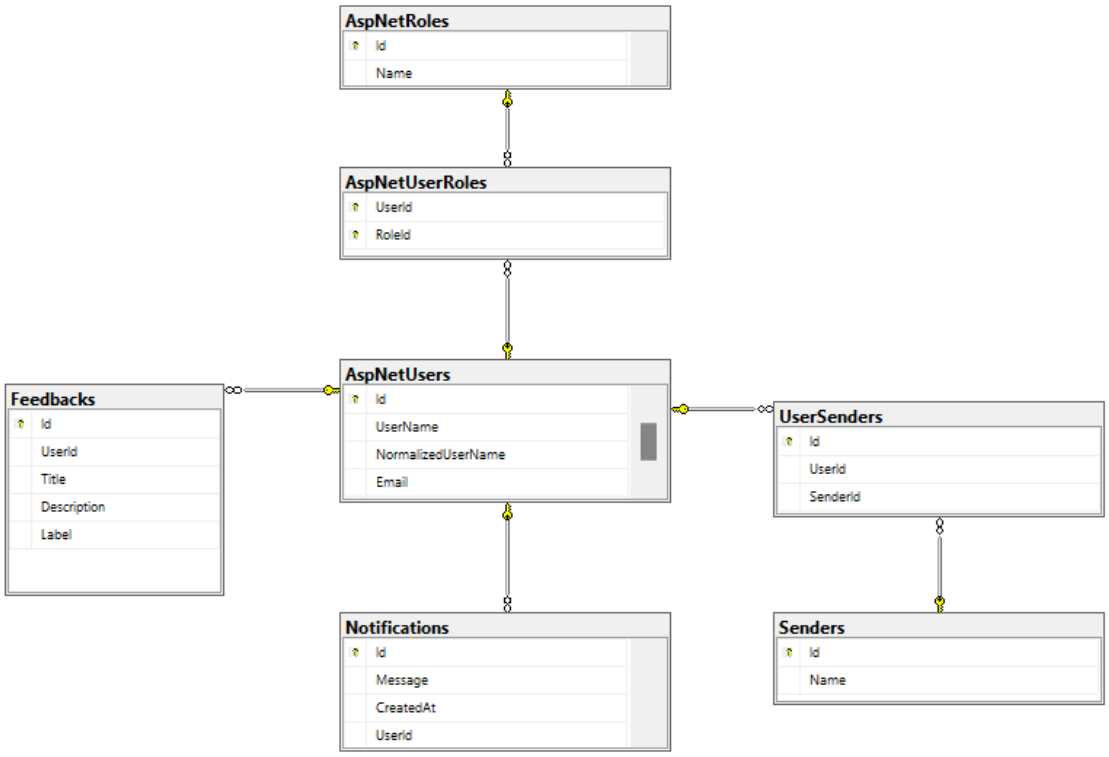
Notifications – טבלת התראות משתמש.

Senders – טבלת מתקשרים.

UserSenders – טבלת קשר בין מתקשר למשתמש.

התרשים הראשון מציג את הקשרים בין הטבלאות העיקריות שבמערכת. בעוד והתרשים השני מציג את מבנה הטבלאות של המשתמשים תפקידים וההתראות.





מבנה הפרויקט

Data Access Layer

שכבה זו אחראית על הגישה למסד הנתונים ומכילה מודלים (Entities) מחלקות המייצגות את טבלאות בסיס הנתונים, מאגרים (Repositories) מחלקות המבצעות פעולות על טבלאות בסיס הנתונים ובנוסף ApplicationContext מחלקה המציינת את הקשר לבסיס הנתונים.

שכבה זו משתמשת ב Entity Framework.

Business Logic Layer

שכבה זו אחראית על הלוגיקה העסקית של האתר. היא מטפלת בעיבוד נתונים, אכיפת כללים עסקיים וקואורדינציה בין שכבות שונות של האתר. השכבה מכילה שירותים (Services) מחלקות הלוגיקה, אירועים (Events) מחלקות המאזינות לאירועים השונים שבאתר, מיזמים (initiators) מחלקות שמטרתן להחל דברים ושרתים.

Presentation Layer

שכבה זו אחראית על תצוגת האתר במבנה MVC.

הקונטרולרים (Controllers) מקבלים בקשות מהמשתמש, מעבדים אותן ומכינים את הנתונים להצגה.

התצוגות (Views) אחראיות על הצגת המידע למשתמש בפורמט HTML ועל קבלת קלט מהמשתמש.

מודלי התצוגה (View Models) משמשים להעברת נתונים בין הקונטרולרים לתצוגות ומגדירים את מבנה הנתונים הספציפי הנדרש לכל תצוגה.

המבנה הזה מאפשר הפרדה ברורה בין הלוגיקה העסקית ,הצגת הנתונים וניהול הבקשות ,מה שמקל על תחזוקה ופיתוח של האפליקציה.

Web Services

פרויקט זה נועד לשירותי רשת ומכיל שירות רשת אחד המספק פונקציונליות הקשורה לניהול ספרים. הוא מאפשר ביצוע פעולות כמו הוספה ,עדכון ,מחיקה וקבלת מידע על ספרים .השימוש ב SOAP- מאפשר תקשורת מבוססת XML עם מערכות אחרות שתומכות בפרוטוקול זה.

Web Api

פרויקט זה מיועד לשמש כממשק API של המערכת. הוא מאפשר גישה לפונקציונליות של המערכת דרך נקודות קצה ,RESTful מה שמאפשר אינטגרציה עם מערכות חיצוניות או פיתוח של אפליקציות לקוח נפרדות .

Utils

פרויקט זה מכיל כלים ופונקציות שימושיות שיכולות לשמש את כל חלקי המערכת. לדוגמה ,שירותי הצפנה, חריגים, ופונקציות עזר אחרות.

Storage

פרויקט זה אחראי על אחסון נתוני הפרויקט ומכיל מספר רב של דפי XML, מבני נתונים ודפים נוספים.

מחלקות

במהלך בניית הפרויקט השתמשתי בgit (מערכת בקרת גרסאות), לכן בחרתי שלא לצרף תמונות עבור מחלקות הפרויקט אלא לספק את קוד הפרויקט.

שרתים

שרת הוא דרך לאחסן את מידע הספר, בחרתי לא לאחסן את המידע במבנה הנתונים הראשי על מנת להדגים גמישות ארכיטקטונית וטיפול ייחודי בנתוני הספרים, בנוסף גישה זו מאפשרת למשתמשים לבחור בצורת האחסון המתאימה לצרכיהם.

Aether

שרת זה מתקשר עם שירות רשת SOAP, שירות הרשת נכתב בNode.js והועלה לרשת בעזרת Railway ולכן ניתן להשתמש בו בתור שירות רשת חיצוני.

שירות הרשת מציג את הספרים בדף XML חיצוני לצפייה: קישור, הקוד.

Atlas

שרת זה מתקשר עם API שנכתב בפרויקט WebApi, הAPI מאחסן את המידע בדף JSON.

Dummy

שרת מדמה שיוצר ומחזיר נתוני ספר דמה. שרת זה אינו מבצע שום פעולות אמיתיות אבל מספק יישום פשוט למטרות בדיקה.

Echo

שרת זה משתמש בDataSet על מנת אחסון In Memory ושומר את המידע בדף XML.

Harmony

שרת זה משתמש בAPI שכתובה בjava ומתקשרת עם plugin של משחק מחשב Minecraft על מנת לאחסן ספר ברכיב במשחק המחשב. הקוד, הסבר.

Nimbus 1.0

שרת זה מתקשר עם מבנה נתונים על מנת לאחסן את מידע הספרים במבנה נתונים טבלאי.

במבנה נתונים זה ישנן שתי טבלאות.

BookChapters – טבלת פרקי הספר.

BooksMetadata – טבלת מידע הספר.

Nimbus 2.0

שרת זה הינו הגרסה השנייה של Nimbus, שרת זה משתמש במסד הנתונים MongoDB ולא SQL Server.

Orion

שרת זה משתמש בשירות רשת הנכתב בפרויקט WebServices שמתקשר עם דף XML.

Solace

שרת זה מאחסן את מידע הספרים באחד מן שלושה קבצים PowerPoint, Excel, Word.

למפתח האפשרות לבחור באיזה קובץ לאחסן את המידע.

Stegan 1.0

שרת זה משתמש בסטגנוגרפיה ומצפין את מידע הספרים לתמונה רנדומלית מAPI חיצונית, לאחר מכן מאחסן את התמונה בתיקייה לוקלית בפרויקט Storage.

*בתמונה זו מוצפן המידע של אחד הספרים שבאתר.

Stegan 2.0

שרת זה הינו הגרסה השנייה של Stegan, במקום לאחסן את תמונות הספרים בתיקייה לוקלית, השרת מעלה את התמונות לאינטרנט בעזרת API שכתבתי בNode.js (הקוד) ומאחסן את הקישור לתמונה במבנה נתונים.

ארכיטקטורה

כל שרת מממש את הממשק הבא:

```
public interface IBookServer {
        Task<Book?> GetBookAsync(int id);
        Task<List<Book>> GetAllBooksAsync();
        Task CreateBookAsync(Book newBook);
        Task UpdateBookAsync(Book updatedBook);
        Task DeleteBookAsync(int id);
  }
```

ממשק זה מאפשר לנו ליצור מחלקת Factory המחזירה שרת לפי שמו.

```
return serverType switch {
    ServerType.Aether => new AetherServer(),
    ServerType.Atlas => new ApiServer("https://localhost:7137"),
    ServerType.Dummy => new DummyServer(),
    ServerType.Echo => new EchoServer(),
    ServerType.Harmony => new ApiServer("https://localhost:8000"),
    ServerType.Nimbus1 => CreateNimbusServer(serviceProvider, ServerType.Nimbus1),
    ServerType.Nimbus2 => CreateNimbusServer(serviceProvider, ServerType.Nimbus2),
    ServerType.Orion => new OrionServer(),
    ServerType.Solace => new SolaceServer(),
    ServerType.Stegan1 => new Stegan1Server(),
    ServerType.Stegan2 => new Stegan2Server(),
    _ => throw new InvalidOperationException($"Invalid server type: {serverType}"),
};
```

ובכך יש לנו Encapsulation מכיוון ושאר מחלקות האתר אינו מודע כיצד מאוחסן ספר אלא רק כי קיים מחלקה המממשת את ממשק השרת, ארכיטקטורה זו מאפשרת לנו ליצור שרת כמו Dummy.

שירותי רשת

האתר מספק שירות SOAP לניהול ספרים. שירות זה מאפשר פעולות בסיסיות על ספרים כמו הוספה, מחיקה, עדכון וקבלת מידע. שירות רשת זה משומש בשרת Orion.

האתר בנוסף לכך משתמש בשני שירותי רשת נוספים אשר נכתבו בNode.js (הקוד), שירות רשת נוסף לניהול ספרים שמשומש בשרת Aether. ושירות רשת בדיחות.

בשירות רשת הבדיחות ישנן שתי פעולות הוספת מספרים ופעולה המחזירה בדיחה רנדומלית מAPI חיצונית. בנוסף העלתי דף XML מרוחק נוסף המחזיר X בדיחות, לצפייה.

API

האתר מספק שני API, אחד לניהול ספרים שמשומש בשרת Atlas וAPI שני המבצע פעולות פשוטות על ספריות הפרויקט, כגון GET, DELETE, PUT, POST.

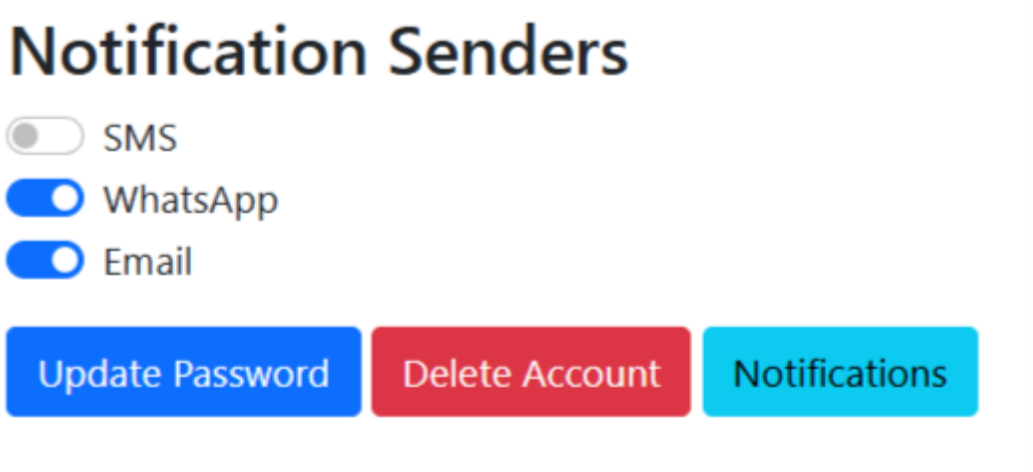
בנוסף על כך הפרויקט משתמש בשני API חיצוני שלא כתבתי, אחד לייצור תמונות רנדומליות (קישור) והשני לבדיחות (קישור).

יתרה על כך הפרויקט משתמש בשני API שכתבתי בNode.js להעלאת קבצים ומעקב אחר אירועים באתר נוסף שלי.

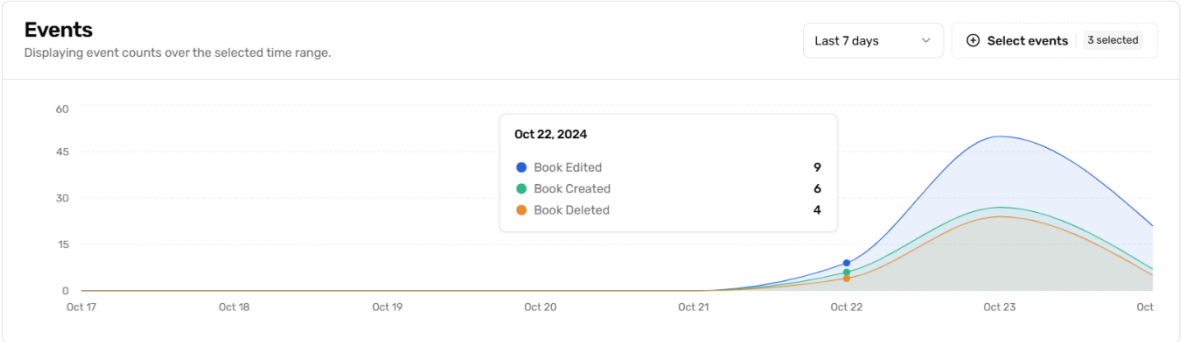
ולבסוף הפרויקט משתמש בAPI שנכתב בJAVA לתקשורת עם plugin במשחק המחשב מיינקראפט.

תכנות מונחה אירועים

האתר שולח הודעה למשתמש כתוצאה מן אירועים מרובים המרחשים באתר, למשתמש האפשרות לבקש מן האתר לשלוח את הודעות אלה בSMS, WhatsApp וEmail.



ישנן מגבלות לשליחת הודעות בSMS וWhatsApp, האתר אינו מסוגל לשלוח הודעה למספר טלפון שלא אישרתי בספק, כלומר על מנת לשלוח הודעה למספר טלפון אצטרך לאשר אותו לפני כן. על מנת להפטר ממגבלה זו אצטרך לשלם מחיר כספי.



האתר משתמש בAPI חיצוני באתר נוסף שכתבתי (קישור) בו ניתן לראות אנליטיקה על אירועי האתר.

נספחים

סרטון הסבר על האתר ללא sound.

סרטון הסבר על האתר עם sound.

API העלאת קבצים.

API מעקב אירועים.

API/Plugin לתקשורת עם משחק המחשב Minecraft.

שירותי רשת בNode.js.

אמצעי כניסה

לצורך כניסה בתור משתמש רגיל, יש למלא את הפרטים הבאים:

אמייל: normal@email.com

סיסמה: Normal123!

לצורך כניסה בתור משתמש מנהל, יש למלא את הפרטים הבאים:

אמייל admin@email.com

סיסמה: Admin123!

ביבליוגרפיה

ASP.NET Core – מסגרת ליצירת אתרים.

Entity Framework – ORM.

Twilio – שליחת הודעות.

MailerSend – שליחת אימיילים.

Syncfusion – ניהול מצגת.