## תיאור המערכת

מערכת ניהול משק בית מאפשרת למשתמש לנהל את ההכנסות וההוצאות בצורה חכמה ומסודרת. היא נותנת מענה לאתגר המרכזי של שמירה על איזון כלכלי וניהול תקציב יעיל, במיוחד עבור משפחות ויחידים המעוניינים לעקוב אחר הוצאותיהם ולתכנן את כספם בצורה טובה יותר.

המערכת תומכת בהעלאת קבלות וחשבוניות ושמירתן, בנוסף להכנסת הוצאות באופן ידני, כולל סכום, נושא ההוצאה ופירוט. המשתמש יוכל להגדיר הוצאות קבועות שיתעדכנו אוטומטית בתאריכים מסוימים בכל חודש. גם ההכנסות תחולקנה להכנסות קבועות וחד-פעמיות.

בנוסף, המערכת מציגה גרפים ודו"חות ויזואליים שממחישים את סכום ההכנסות וההוצאות החודשיות, לצד השוואות בין חודשים קודמים. כך, המשתמש מקבל תובנות כלכליות ומסוגל לקבל החלטות פיננסיות מושכלות יותר.

**יתרונות עיקריים:**

* ניהול תקציב חכם עם מעקב בזמן אמת.
* העלאת ושמירת קבלות וחשבוניות להיסטוריית תשלומים מסודרת.
* הגדרת הוצאות והכנסות קבועות עם עדכון אוטומטי. -פיתוח בהמשך
* הצגת גרפים ודוחות ויזואליים להשוואת תקופות והפקת תובנות פיננסיות.

**קהל היעד**

* משפחות המעוניינות לעקוב אחר ההוצאות וההכנסות ולשפר את ניהול התקציב.
* יחידים שרוצים לנהל טוב יותר את הוצאותיהם האישיות.

## פונקציונליות של המערכת

1. **הזנת הכנסות והוצאות** - אפשרות להזין הכנסות והוצאות ידנית כולל פירוט מלא של העסקה. וסיווג לפי קטגוריה.
2. **העלאת מסמכים -** אפשרות לצרף קבלות וחשבוניות לכל הוצאה או הכנסה, עם אפשרות חיפוש ושליפה לפי קטגוריה או עסקה ספציפית
3. **(הוצאות והכנסות קבועות** - תזמון ותיעוד אוטומטי של הכנסות והוצאות מחזוריות (שכר, שכירות, מנויים וכו').**)-פיתוח המשך**
4. **דו"חות וגרפים** - הצגת דוחות גרפיים דינמיים לצורך ניתוח ההתנהלות הכלכלית, כולל השוואת תקופות קודמות לפי תאריכים וקטגוריות.

## בעיות שהמערכת פותרת

* **חוסר שליטה בהוצאות וחוסר שקיפות בתזרים המזומנים** – המערכת מאפשרת מעקב קל ומיידי אחרי כל הוצאה והכנסה, ובכך מספקת תמונה ברורה של מצב הכספים בזמן אמת.
* **קושי בארגון ושמירת קבלות וחשבוניות בצורה מסודרת** – המערכת מאפשרת העלאה ושמירה דיגיטלית של כל הקבלות והחשבוניות, עם אפשרות לחפש ולהציג אותן לפי קטגוריות או עסקה ספציפית.
* **חוסר מעקב אחר חיובים והוצאות קבועות** – המערכת תומכת בהגדרה ותזמון אוטומטי של הוצאות והכנסות קבועות, ומציגה גם אותם.
* **צורך בניתוח נתונים על ההוצאות והכנסות לטובת קבלת החלטות כלכליות חכמות יותר** – המערכת מציעה דו"חות וגרפים דינמיים, המאפשרים ניתוח ברור של נתוני ההוצאות וההכנסות, ומסייעת למשתמשים לקבל החלטות כלכליות מבוססות נתונים.

## טכנולוגיות בשימוש

1. **צד שרת (Backend):**NET 9 לבניית API גמיש ומאובטח.
2. **צד לקוח (Frontend)**: React ו-Angular לבניית ממשק משתמשי קצה.
3. **ניהול מערכת** -אנגולר19 לבניית ממשק לניהול המערכת.
4. **מסד נתונים**: MySQL לניהול ואחסון נתוני הכנסות, הוצאות, ומשתמשים.
5. **אחסון קבצים**: שימוש בשירותי אחסון בענן לשמירת קבלות וחשבוניות.

## מתודולוגיית פיתוח (Agile)

הפרויקט ינוהל בשיטת Agile תוך חלוקה לספרינטים:

### ספרינט 1 - הקמת תשתיות

* יצירת מסד הנתונים עם טבלאות משתמשים, הכנסות, הוצאות ומסמכים.
* פיתוח API ראשוני ב .NET לניהול נתונים בסיסי.
* הוספת אפשרות להעלאת קבלות וחשבוניות.
* הגדרת מנגנון אימות משתמשים עם JWT.
* פריסת המערכת לשרתים בענן.

### ספרינט 2 - בניית ממשק משתמש

* פיתוח דפי כניסה ורישום.
* עיצוב מסכים להזנת הכנסות והוצאות.
* חיבור API לממשק המשתמש ב-React

### (ספרינט 3 - פונקציונליות מתקדמת) פיתוח המשך ב"ה

* הגדרת הוצאות והכנסות קבועות עם חישוב אוטומטי.
* (יצירת התראות ותזכורות להוצאות חריגות).

### ספרינט 4 - דו"חות וגרפים

* שילוב ספריות להצגת גרפים ודוחות ויזואליים.
* הצגת סיכום חודשי והשוואת חודשים קודמים.

### מטרות הספרינטים:

* ספרינט 1 מתמקד בהקמה של תשתיות בסיסיות, פריסת המערכת והבטחת אבטחה.
* ספרינט 2 מתמקד בממשק משתמש נגיש וידידותי למשתמשים.
* ספרינט 4 משלים את הפיתוח עם כלים לניתוח נתונים וביצועים.

## עיצוב ממשק משתמש ויזואלי

#### מסכים עיקריים**:** (מצורף ויזואלי)

1. **מסך כניסה/רישום** - אימות משתמשים.
2. **מסך הוצאות** - אפשרות להזנה, עריכה וסיווג לפי קטגוריות.
3. **מסך הכנסות** - אפשרות להזנה, עריכה וסיווג לפי קטגוריות.
4. **מסך וגרפים** - הצגת גרף נתונים.
5. **מסך דוחות**-דוח שנתי לפי קטגוריות ודוח לפי תארכים נבחרים

## פונקציות מערכת

#### **- AuthController** ניהול אימות

* **login**
* **Route:**
* POST /api/Auth/login
* **פרמטרים:**
  + **Body:** { "username": "string", "password": "string" }
* **פלט:**
  + **200:** { "token": "string", "expiresIn": "number" }
  + **401:** { "error": "Invalid credentials" }
* **לוגיקה עסקית:** מבצע אימות משתמש, יוצר טוקן JWT ומחזירו למשתמש.
* **Register**:
* **Route:**
  + OST /api/Auth/register
* **פרמטרים:**
  + **Body:** { "username": "string", "email": "string", "password": "string" }
* **פלט:**
  + **201:** { "message": "User created successfully" }
  + **400:** { "error": "User already exists" }
* **לוגיקה עסקית:** יוצר משתמש חדש ושומרו במסד הנתונים.

#### **ExpensesController** ניהול הוצאות

* **GetExpenses**
* **Route:**
  + GET /api/Expenses
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
* **פלט:**
  + **200:** [{ "id": "number", "amount": "number", "category": "string" }]
  + **403:** { "error": "Unauthorized" }
* **לוגיקה עסקית:** מחזיר רשימת הוצאות למשתמש המחובר.
* **AddExpense**
* **Route:**
  + POST /api/Expenses
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
  + **Body:** { "amount": "number", "category": "string" }
* **פלט:**
  + **201:** { "message": "Expense added successfully" }
  + **400:** { "error": "Invalid data" }
* **לוגיקה עסקית:** מוסיף הוצאה חדשה למערכת.

#### **FileController** ניהול קבצים

* **UploadFile**
* **Route:**
  + POST /api/File/upload
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
  + **Body:** multipart/form-data { file: binary }
* **פלט:**
  + **200:** { "message": "File uploaded successfully", "fileId": "string" }
  + **400:** { "error": "Invalid file format" }
* **לוגיקה עסקית:** שומר קובץ במסד הנתונים או בענן.
* **DownloadFile**
* **Route:** GET /api/File/download/{fileId}
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
* **פלט:**
  + **200:** file/binary
  + **404:** { "error": "File not found" }
* **לוגיקה עסקית:** מחזיר קובץ קיים להורדה.

#### **S3Controller** אינטגרציה עם S3

* **GenerateUploadUrl**
* **Route:** GET /api/S3/generate-upload-url/{fileName}
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
* **פלט:**
  + **200:** { "uploadUrl": "string" }
  + **500:** { "error": "Failed to generate URL" }
* **לוגיקה עסקית:** יוצר URL זמני להעלאת קובץ ל-S3.

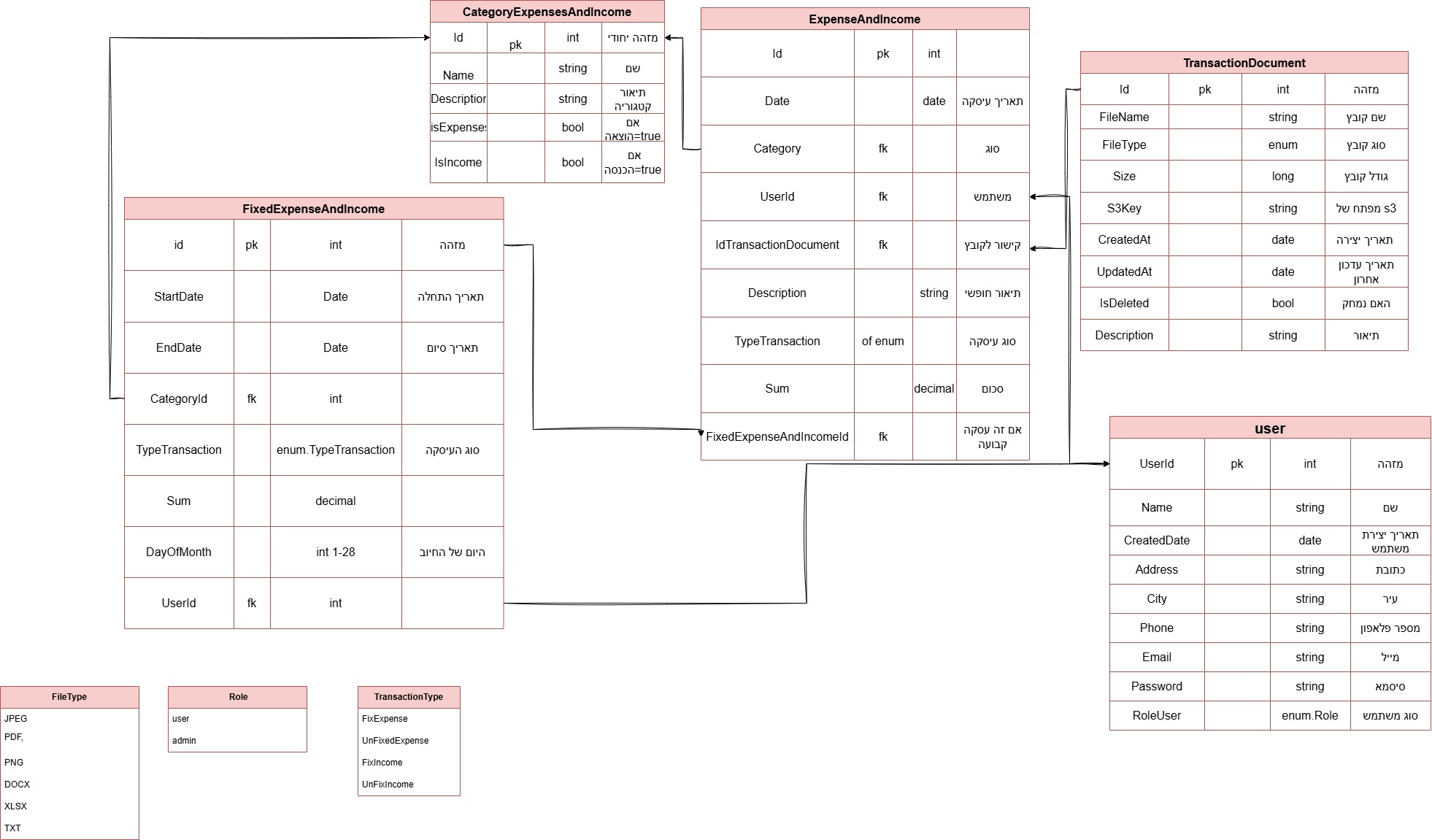
#### **UserController**- ניהול משתמשים

* **GetUserProfile**
* **Route:** GET /api/User/profile
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
* **פלט:**
  + **200:** { "id": "string", "username": "string", "email": "string" }
  + **403:** { "error": "Unauthorized" }
* **לוגיקה עסקית:** מחזיר את פרטי המשתמש המחובר.
* **UpdateProfile**
* **Route:** PUT /api/User/profile
* **פרמטרים:**
  + **Headers:** Authorization: Bearer <token>
  + **Body:** { "username": "string", "email": "string" }
* **פלט:**
  + **200:** { "message": "Profile updated successfully" }
  + **400:** { "error": "Invalid data" }
* **לוגיקה עסקית:** מעדכן את פרטי המשתמש במערכת.

### אימות והרשאות

* אימות מבוסס JWT.
* תפקידי משתמשים:
  + **משתמש קצה** - הוספת וניהול נתונים אישיים בלבד.
  + **מנהל מערכת** - גישה לכל הנתונים, יצירת משתמשים וניהול הרשאות.

## תרשים מבנה הטבלאות

(מצורף גם במסכים ביותר גדול)