בס''ד פרקטיקום-חיה טורצין   
  
  
מסמך אפיון עבור מערכת לניהול הוצאות אישיות ומשפחתיות   
  
**מהות הפרויקט:**אפליקציה לניהול הוצאות אישיות ומשפחתיות, שמאפשרת למשתמשים להקליט את ההוצאות וההכנסות שלהם, לנהל תקציבים, לצפות בדוחות אוטומטיים עם גרפים, ולקבל התראות על חריגות בתקציב. המערכת תאפשר גם אינטגרציה עם מערכת ניהול הבנק כדי לבצע מעקב אוטומטי אחר ההוצאות **קהל היעד:**  
בעלי משפחות המעוניינים לנהל את המאזן הכלכלי שלהם.\*

יחידים המעוניינים לנהל את התקציב האישי שלהם בצורה מסודרת.\*

\*בעלי עסקים קטנים המעוניינים לנהל את ההוצאות וההכנסות של העסק לצד הניהול האישי.

### פונקציונליות של המערכת:

**הקלטת הוצאות והכנסות:** \*המשתמשים יכולים להקליט הוצאות והכנסות בצורה נוחה, תוך חיבור לחשבון הבנק לצורך עדכון אוטומטי.

**יצירת קטגוריות והגדרת תקציב:** \*יצירת קטגוריות שונות להוצאות והכנסות (כמו "מזון", "תחבורה", "החזרי הלוואות") והגדרת תקציב לכל קטגוריה.

**הצגת דוחות אוטומטיים:** \*המערכת תייצר דוחות גרפיים ומפורטים המציגים את המצב הכלכלי של המשתמש, עם פילוחים לפי קטגוריות, תקופות זמן וסטטוס חריגות.

**התראות על חריגות:** \*המערכת תשלח התראות למשתמשים כאשר הם חורגים מהתקציב שהוגדר לכל קטגוריה.

**אינטגרציה עם מערכת ניהול הבנק:** \*חיבור ישיר לחשבון הבנק של המשתמש לצורך עדכון אוטומטי של הוצאות והכנסות מהבנק.

### טכנולוגיות בשימוש:

הפרויקט יפותח באמצעות הטכנולוגיות הבאות:

**צד שרת (Backend):** .NET 8 לבניית ה-API.\*

**צד לקוח (Frontend):** React.js לבניית ממשק משתמש אינטראקטיבי.\*

**מסד נתונים:** PostgreSQL (או SQL Server) \*לאחסון נתוני המשתמשים, ההוצאות, ההכנסות, התקציבים והדוחות.

**אינטגרציה עם בנק:** שימוש בשירותי API \*חיצוניים לקבלת נתוני הבנק של המשתמש.

**אחסון קבצים:** Amazon S3 לאחסון קבצים, אם נדרש.\*

### בעיות שהמערכת פותרת

### המערכת פותרת את הבעיה של ניהול לא מסודר של הוצאות והכנסות, ומאפשרת למשתמשים לעקוב אחרי מצבם הכלכלי בצורה ברורה ומאורגנת. היא גם פותרת את בעיית הבעיות בניהול תקציב ביתי או אישי, ומספקת התראות במקרים של חריגות בתקציב. מתודולוגיית פיתוח (Agile):

הפרויקט יפותח במתודולוגיית Agile עם חלוקה ל-5 ספרינטים בן שבוע כל אחד. כל ספרינט יתמקד במשימות ליבה עיקריות.   
להלן פירוט הספרינטים הצפויים בעז''ה:

#### ספרינט 1 - בניית תשתיות וניהול משתמשים:

תכנון מודל מסד הנתונים ב-PostgreSQL (\*טבלאות משתמשים, הוצאות, הכנסות, תקציבים).

דיאגרמת UML להמחשת מבנה המערכת.\*

פיתוח API בסיסי ב-.NET 8 עם endpoints לניהול משתמשים, הוצאות, הכנסות וקטגוריות.\*

הטמעת מערכת אימות משתמשים (Registration/Login) עם JWT.\*

#### ספרינט 2 - ממשק משתמש ראשוני:

בניית רכיבי React.js לעיצוב ממשק מנהל.\*

אינטגרציה בין Frontend ל-API באמצעות Axios.\*

הוספת גישה למשתמשים להקלדת הוצאות והכנסות.\*

#### ספרינט 3 - פיצ'רים ליבה:

יצירת דוחות וגרפים אוטומטיים לפי נתוני המשתמש.\*

התראות על חריגות בתקציב.\*

חיבור לחשבון הבנק של המשתמש לקבלת נתונים אוטומטיים.\*

#### ספרינט 4 - שיפורים ואבטחה:

בדיקות אבטחה ואופטימיזציות ב-API.\*

שיפורים בממשק המשתמש.\*

\*שיפור חוויית המשתמש והתממשקות עם מערכות נוספות (לדוגמה, הוספת פונקציות לתכנון תקציב עתידי).

#### ספרינט 5 - פריסה ופרסום:

פריסת האפליקציה בשרת AWS.\*

הגדרת דומיין אישי.\*

עיצוב דף הבית לשיווק ומשיכת משתמשים פוטנציאליים.\*

### פירוט על הטכנולוגיות:

#### :

#### צד שרת (Backend): NET 8. לבניית הAPI . React.js:

ספריית JavaScript לפיתוח ממשקי משתמש. יתרונות:

* רכיבים ניתנים לשימוש חוזר.
* ביצועים טובים עם Virtual DOM.

#### PostgreSQL:

מערכת ניהול מסדי נתונים רלציונית (RDBMS). יתרונות:

* תמיכה בשפת SQL ובפונקציות מתקדמות.
* גמישות בתמיכה בנתונים לא מסודרים (JSON).

#### אינטגרציה עם בנק:

המערכת תעשה שימוש בשירותי API חיצוניים של בנקים לאסוף אוטומטית נתוני הוצאות והכנסות, ובכך לאפשר למשתמש לעקוב אחרי נתונים בזמן אמת.

### רשימת פונקציות המערכת:

#### פונקציות על נתוני הוצאות והכנסות:

* **העלאת הוצאה/הכנסה** (POST /api/expenses)
  + פרמטרים: {amount: string, category: string, date: string}
  + פלט: {expenseId: string, message: string}
* **יצירת דוח הוצאות** (GET /api/reports)
  + פרמטרים: {startDate: string, endDate: string}
  + פלט: {report: object}

#### פונקציות למשתמשים:

* **רישום משתמש חדש** (POST /api/register)
  + פרמטרים: {username: string, password: string, role: string}
  + פלט: {userId: string, message: string}
* **כניסת משתמש** (POST /api/login)
  + פרמטרים: {username: string, password: string}
  + פלט: {token: string, userId: string}

### תרשים מבנה טבלאות (Database Schema):

#### טבלת Users:

* id (PK): מזהה ייחודי
* username: שם משתמש
* password: סיסמה (מוצפנת)
* role: תפקיד (admin/user)

#### טבלת Expenses:

* id (PK): מזהה ייחודי
* user\_id (FK): מזהה המשתמש
* amount: סכום ההוצאה
* category: קטגוריית ההוצאה
* date: תאריך ההוצאה