تقرير عن مشروع نظام تتبع النفقات الشخصية (Expense Tracker)

أسماء الفريق:

١- عاصم محمد سعيد علي (القائد)
٢- صقر احمد عبده سعيد
٣- سلطان عبدالله سلطان
٤- عبدالرقيب نعمان احمد
٥- زكريا عبدالسلام

أشراف المهندس عبدالسلام

تقرير عن مشروع نظام تتبع النفقات الشخصية (Expense Tracker)

❖ مقدمة ورسالة المشروع.

يهدف مشروع "نظام تتبع النفقات الشخصية" إلى تمكين المستخدمين من إدارة أموالهم اليومية بكفاءة وشفافية. عبر واجهة رسومية سهلة الاستخدام، يتيح التطبيق تسجيل المصروفات والدخل، تنظيمها حسب فئات مخصصة، وتحديد ميزانيات لكل فئة، مع تقديم تنبيهات وتقارير تحليلية تساعد المستخدم على فهم نمط إنفاقه واتخاذ قرارات مالية أفضل.

❖ وصف المشروع:

نظام تتبع النفقات هو تطبيق مكتبي مكتوب بلغة C# باستخدام بيئة Visual Studio وتقنيةWindows Forms ، مع استخدام قاعدة بيانات .SQLite يهدف إلى مساعدة المستخدمين على إدارة نفقاتهم وميزانياتهم بطريقة مرنة، وذلك من خلال تسجيل المصروفات، وإنشاء فئات خاصة، وتحديد ميزانية لكل فئة، مع توفير تقارير مخصصة لتحليل الإنفاق

❖ وصف المشكلة التي يعالجها المشروع

يعاني العديد من الأفراد من صعوبة في تتبع مصروفاتهم اليومية وتنظيم ميزانياتهم، ما يؤدي إلى نفقات غير منضبطة وتجاوز الحدود المالية المحددة .يحل التطبيق هذه المشكلة بتوفير أداة مرنة لتسجيل ومراقبة النفقات والدخل، وتنبيه المستخدم عند اقتراب أو تجاوز الميزانية.

❖ سبب اختيار قاعدة بيانات SQLite في المشروع

١. **خفة التشغيل وسرعة الأداء**

- SQLiteهو نظام إدارة قواعد بيانات مضمّن (embedded) ، لا يحتاج إلى تثبيت أو تشغيل خادم.(server)
- نظرًا لصغر حجم المشروع ومحدودية حجم البيانات المتوقعة (مصروفات شخصية)، فإن SQLite يوفر وقت استجابة سريعًا وفعالية عالية في التخزين والاستعلام.

٢. **سهولة التوزيع والصيانة**

- قاعدة البيانات مخزّنة في ملف واحد (`ExpenseTracker.db`) ضمن ملفات التطبيق، مما يجعل نسخ التطبيق أو ترقية إصداراته سهلة جدًا دون خطوات معقدة لنقل البيانات.
- لا حاجة لإعداد خادم منفصل أو إدارة اتصالات شبكة، مما يقلل الاعتمادية على بيئة تشغيل مُعقدة.

٣. **اعتمادية واستقرار**

- SQLite مفتوح المصدر ومدعوم على نطاق واسع، مع سجل طويل في الاستقرار.
- يستخدمه آلاف المشاريع الصغيرة والمتوسطة وحتى بعض التطبيقات الكبيرة كـ "cache" أو قاعدة بيانات محلية.

٤. **دعم كامل للخصائص اللازمة**

- يوفّر SQLite دعمًا كاملاً لـ SQL القياسي تقريبًا: الجداول، المفاتيح الأساسية والأجنبية، الفهارس (INDEX) ، الصيغ المحسوبة (VIEW) ، وغيرها.
- يمكنك بسهولة تنفيذ العمليات CRUD (إنشاء، قراءة، تحديث، حذف) والعمليات التجميعية (SUM, COUNT) المطلوبة لتتبع النفقات وإعداد التقارير.

٥. **انعدام التعقيد وبدء سريع**

- لجهة تطوير واجهات WinForms والربط في C# عبر مكتبة System.Data.SQLite، لا يتطلب المشروع إعدادًا أوليًا مطوّلًا أو بنية خادم قواعد بيانات.
- يكفي تضمين ملف الـ DLL وتحديد سلسلة الاتصال إلى ملفSQLite ، فيسرع ذلك من وتيرة التطوير.

❖ مهام كل طالب في المشروع

١. عاصم محمد Admin

- تصميم قاعدة البيانات بالكامل ERD :والجداول والعلاقات.
- تطوير الطبقة الوسيطة (Repositories) لـ CRUD على Expense و.Budget
- تصميم كلاس ثيم لتنسيق الازرار والواجهات
- تصميم واجهة انشاء حساب جديد

٢. عبدالرقيب نعمان

- تطوير شاشة إعداد الميزانية BudgetForm وربط الحقول.
- كتابة دوال Add/Get/Delete للميزانية مع التحقق من صحة التواريخ.
- تصميم واجهات تسجيل الدخول

٣. صقر احمد

- تصميم واجهة إضافة النفقة AddExpenseForm وتعبئة ComboBox و.DataGridView
- إضافة زر حذف النفقة مع رسالة تأكيد.
- تنفيذ تنبيهات تجاوز الميزانية عند.90%

٤. سلطان عبدالله سلطان

- تصميم شاشة التقارير ReportForm و.ReportChartForm
- رسم مخططات أعمدة باستخدام.Charting Library
- إضافة زر تصدير PDF باستخدام iTextSharp
- اختبار حفظ التقارير الكبيرة وضمان استقراريتها.

٥. زكريا عبدالسلام

- ربط الجلسة.Session Management
- التكامل مع AddExpenseForm لإعادة تحميل الميزانيات.
- دمج بيانات النفقة مع الفئة باستخدام.LINQ

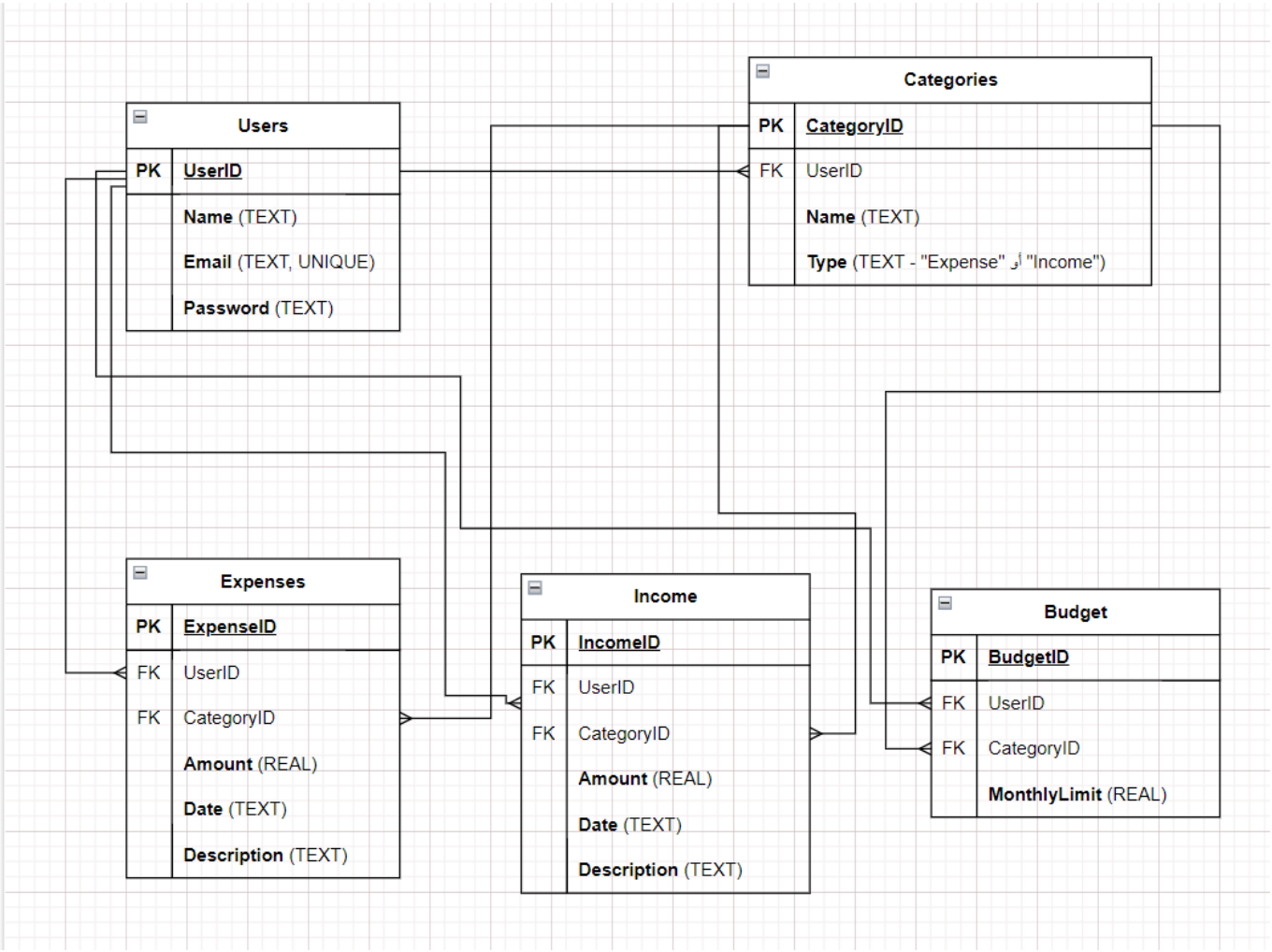
❖ الأهداف الرئيسة

١. **تسجيل وتتبع النفقات :**إضافة، تعديل، وحذف مصروفات المستخدم بسهولة.
٢. **إعداد الميزانيات :**تحديد حد أقصى للإنفاق على كل فئة ومراقبته.
٣. **التصنيف :**تقسيم المصروفات والدخل لفئات (طعام، مواصلات، فواتير، إلخ).
٤. **التنبيهات :**إشعار المستخدم عند اقتراب أو تجاوز حدود الميزانية.
٥. **التقارير والتحليل :**عرض ملخصات ورسم بياني يوضح حجم الإنفاق حسب الفئات والفترات الزمنية.
٦. **تعدد المستخدمين :**كل مستخدم لديه حساب مستقل وبيانات خاصة به.

❖ المتطلبات والوظائف الأساسية

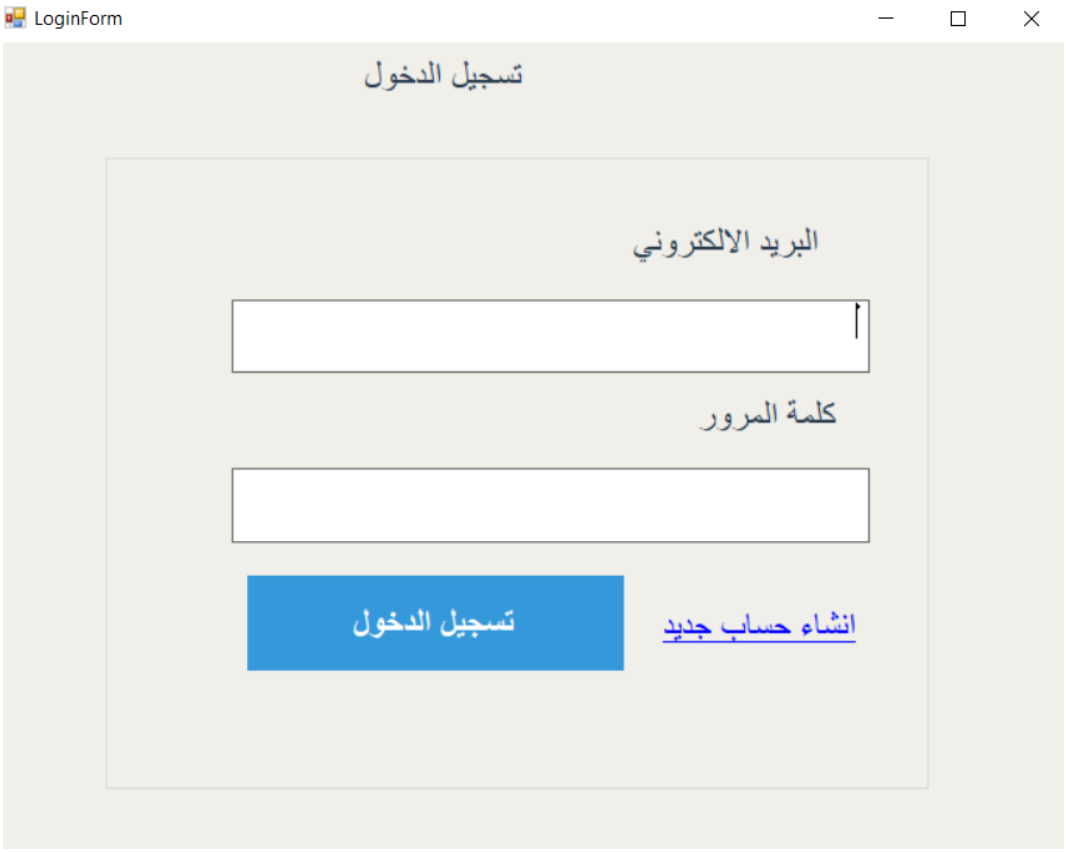
الرقم	الوظيفة	الوصف
1	إدارة الحسابات	**تسجيل مستخدم جديد وتسجيل الدخول.**
2	إدارة الفئات	**إنشاء، وحذف فئات خاصة بكل مستخدم، ودعم فئات عامة.**
3	إضافة النفقات	**إدخال مبلغ، تاريخ، وصف، وفئة.**
4	إعداد الميزانيات	**تحديد ميزانيات للفئات مع تواريخ بداية ونهاية.**
5	تنبيهات الميزانية	**إشعار عند بلوغ ٩٠% أو تجاوز ١٠٠% من الحد.**
6	عرض التقارير	**عرض جدول وتقارير رسومية (أعمدة/دوائر) للنفقات.**
7	إدارة البيانات	**دعم تصدير إلى.PDF**
8	الأمان	**تخزين آمن لكلمات المرور، وإمكانية تأمين التطبيق بكلمة مرور.**

ER Diagram


Users
PK UserID
Name (TEXT)
Email (TEXT, UNIQUE)
Password (TEXT)
Categories
PK CategoryID
FK UserID
Name (TEXT)
Type (TEXT - "Expense" أو "Income")
Expenses
PK ExpenseID
FK UserID
FK CategoryID
Amount (REAL)
Date (TEXT)
Description (TEXT)
Income
PK IncomeID
FK UserID
FK CategoryID
Amount (REAL)
Date (TEXT)
Description (TEXT)
Budget
PK BudgetID
FK UserID
FK CategoryID
MonthlyLimit (REAL)

❖ تصميم الواجهات

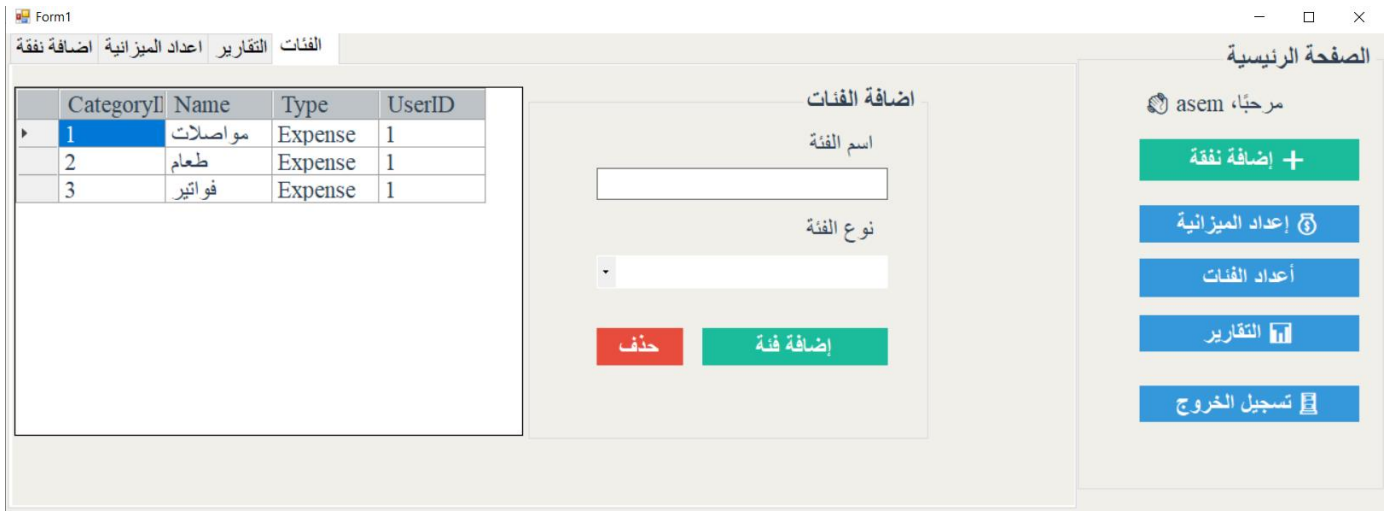
١- واجهة تسجيل الدخول



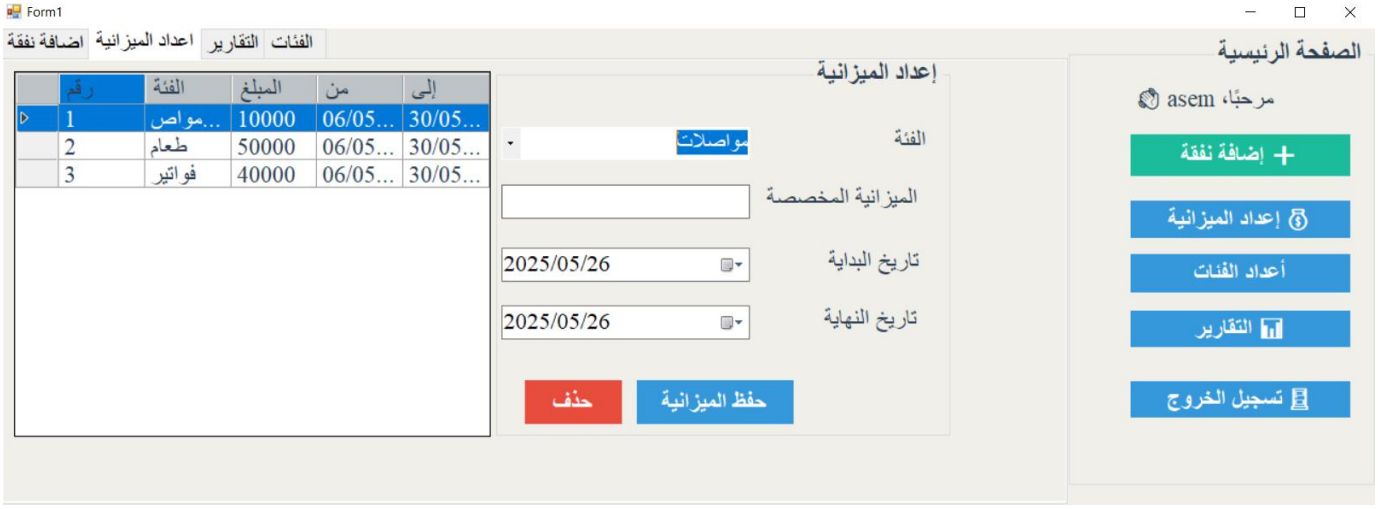
٢- واجهة انشاء حساب جديد

٣- الواجهة الرئيسية مع واجهة اضافة النفقات

٤- واجهة اعداد الفئات

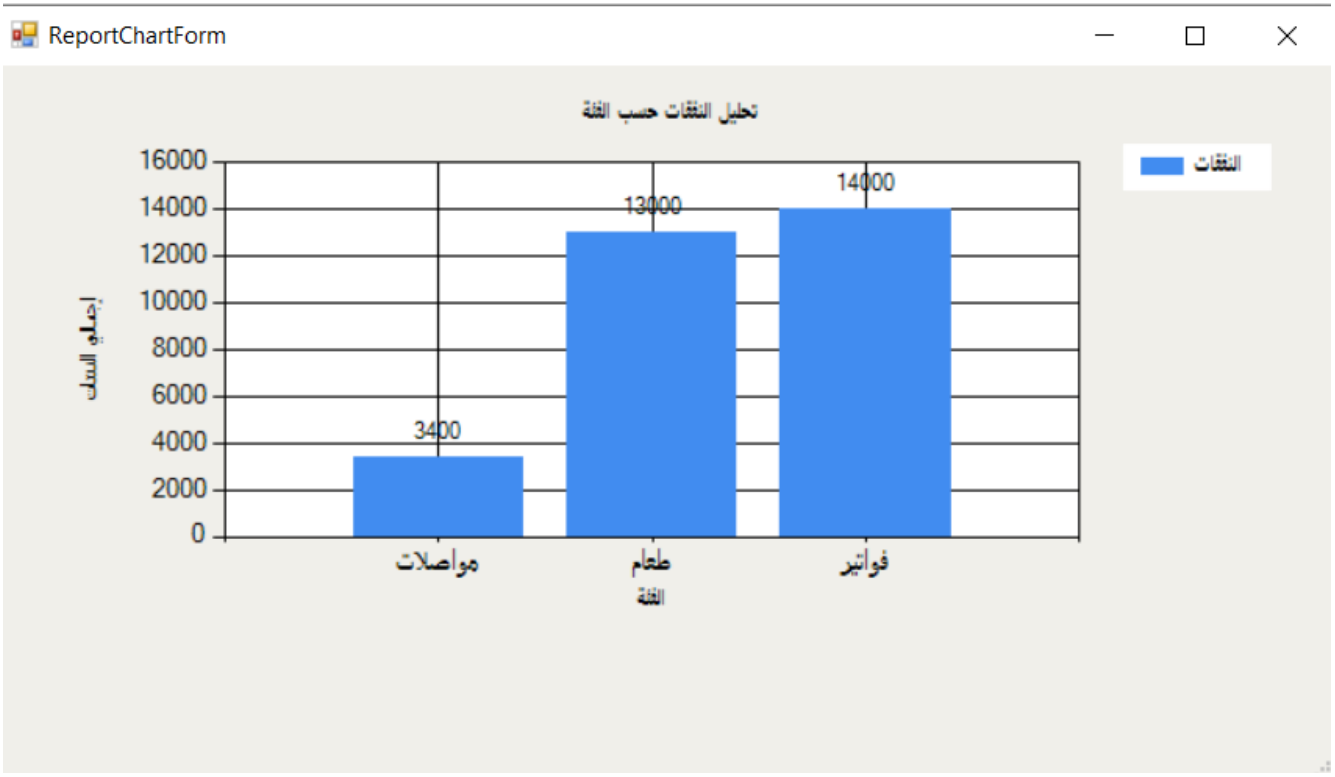


٥- واجهة اعداد الميزانية



٦- واجهة التقارير

٧- واجهة التقرير الرسومي



❖ التصميم المعماري

١. **نموذج MVC (بشكل مبسط):**
   - **Model**:
     يحتوي على الكلاسات `User`, `Category`, `Expense`, `Budget`.
   - **View**:
     Windows Forms (`LoginForm`, `RegisterForm`, `MainForm`, `AddExpenseForm`, `BudgetForm`, `ReportForm`, `ReportChartForm`, `CategoriesForm`).
   - **Controller/Repository**:
     طبقة `Database` المسؤولة عن جميع عمليات CRUD على SQLite من خلال كائنات `UserRepository`, `CategoryRepository`, `ExpenseRepository`, `BudgetRepository`.

٢. **قاعدة البيانات: (SQLite)**
   - **Users (UserID PK, Name, Email, Password)**
   - **Categories (CategoryID PK, Name, Type, UserID FK→Users)**
   - **Expenses (ExpenseID PK, UserID FK→Users, CategoryID FK→Categories, Amount, Date, Description)**
   - **Budgets (BudgetID PK, UserID FK→Users, CategoryID FK→Categories, Amount, StartDate, EndDate)**

٣. **التنقل بين النوافذ:**
   - بعد تسجيل الدخول تظهر `MainForm` التي تحتوي أزرار: إضافة نفقة، عرض التقارير، إعداد ميزانية، إدارة الفئات، تسجيل الخروج.

❖

❖ تفاصيل التنفيذ

5.1. إدارة الحسابات

- **RegisterForm:** التقاط الاسم والإيميل وكلمة المرور، التحقق من تطابق كلمة المرور، ثم `userRepo.AddUser(user)`.
- **LoginForm:** التحقق عبر `userRepo.GetUserByEmailAndPassword(email, pwd)`, تخزين المستخدم في `Session.CurrentUser`.

5.2. إدارة الفئات

- `CategoriesForm` يعرض جدول الفئات (`DataGridView`) ، ويستخدم `GetAllCategoriesByUser(Session.CurrentUser.UserID)` لتصفية الفئات.
- عند إضافة فئة جديدة : `categoryRepo.AddCategory(new Category { Name, Type, UserID = Session.CurrentUser.UserID })`.

5.3. إضافة وتعديل وحذف النفقات

- **AddExpenseForm:**
  ١. تحميل الفئات الخاصة بالمستخدم في ComboBox عبر GetAllCategoriesByUser.
  ٢. زر "إضافة نفقة": يحصل على القيم، يستدعي expenseRepo.AddExpense(userId, categoryId, amount, date, desc).
  ٣. بعد الإضافة يقوم بـ LoadExpenses() لإعادة تحميل الجدول.
  ٤. زر "حذف": يحذف النفقة باستخدام expenseRepo.DeleteExpense(expenseId).

5.4. إعداد الميزانية

- **BudgetForm:**
  ١. تحميل الفئات في ComboBox.
  ٢. إنشاء ميزانية budgetRepo.AddBudget(new Budget { UserID, CategoryID, Amount, StartDate, EndDate }):
  ٣. عرض الميزانيات السابقة في DataGridView باستخدام GetBudgetsByUser(Session.CurrentUser.UserID).
  ٤. زر "حذف الميزانية": budgetRepo.DeleteBudget(budgetId).

5.5. التنبيهات عند قرب تجاوز الميزانية

- بعد إضافة نفقة جديدة في AddExpenseForm, يتم استعلام مجموع النفقات الحالية SUM(Amount) ثم مقارنة مع budget.Amount.
- إذا تجاوزت ٩٠% من الميزانية MessageBox: يظهر تحذيرًا؛ وإذا تجاوزت ١٠٠% يظهر تنبيهًا باللون الخطأ.

5.6. التقارير والتحليلات

- **ReportForm:** تصفية الفئات والتواريخ، عرض جدول DataGridView بالنفقات.
- **ReportChartForm:** رسم بياني عمودي يوضّح مجموع الإنفاق حسب الفئة، عبر System.Windows.Forms.DataVisualization.Charting.

❖ المكتبات والتقنيات

- **C# (.NET Framework 4.x)**
- **Windows Forms (WinForms)**
- **SQLite** (قاعدة بيانات محلية وخفيفة)
- **System.Data.SQLite** الربط بمكتبة SQLite
- **System.Collections.Generic**
- **System.Linq / System.Text / System.Drawing** **لتحسين اداء المعالجة والواجهة**
- **System.Windows.Forms.DataVisualization** (لـ Charts)
- **System.Linq** (للتجميع والفرز)
- **System.IO**
- **iTextSharp.text**
- **iTextSharp.text.pdf**
- **MVC Pattern** (فصل المنطق عن العرض).

❖ اقتراحات للتطوير المستقبلي

١. **التصدير/الاستيراد :** دعم Excel لتحليل خارجي.
٢. **الإخطارات الدورية :** عبر إشعارات سطح المكتب عند مواعيد محددة.
٣. **تطبيق ويب/محمول:** لتوسيع وصول المستخدمين.
٤. **تشفير كلمات المرور :** باستخدام `bcrypt` أو `SHA-256`.
٥. **لوحة تحكم : (Dashboard)** تعرض ملخصات فورية ورسوم بيانية تفاعلية.

❖ الخاتمة:

هذا المشروع يقدم قاعدة قوية لتطبيق متكامل لإدارة النفقات والميزانيات الشخصية، ويعتمد على مكونات بسيطة وفعّالة. مع إضافة التحسينات المقترحة، يمكن أن يتحول إلى أداة مالية متقدمة تدعم المستخدمين في اتخاذ قرارات مالية أفضل.