Świetny wybór! Aplikacja do zarządzania budżetem osobistym może pomóc użytkownikom śledzić swoje finanse, planować wydatki, oszczędności i osiągać cele finansowe. Poniżej przedstawiam szczegóły dotyczące funkcji i technologii, które mogą być wykorzystane w tym projekcie.

**Funkcje aplikacji do zarządzania budżetem osobistym**

1. **Rejestracja i logowanie użytkownika**
   * Umożliwia tworzenie konta i logowanie do aplikacji, aby użytkownicy mogli przechowywać swoje dane finansowe w chmurze.
   * Funkcje takie jak resetowanie hasła, uwierzytelnianie dwuetapowe mogą zwiększyć bezpieczeństwo.
2. **Śledzenie przychodów i wydatków**
   * Możliwość dodawania różnych przychodów (np. pensja, bonusy) oraz wydatków (np. zakupy, rachunki, subskrypcje).
   * Kategoryzowanie wydatków (np. jedzenie, transport, rozrywka) i przychodów (np. praca, inne źródła).
   * Integracja z bankami lub systemami płatności w celu automatycznego importowania transakcji (np. przez API).
3. **Tworzenie budżetów miesięcznych**
   * Użytkownicy mogą ustawiać limity na poszczególne kategorie wydatków (np. 300 zł na jedzenie, 100 zł na rozrywkę).
   * Aplikacja będzie śledzić postępy i przypominać o przekroczeniu ustalonych limitów.
4. **Analiza wydatków i oszczędności**
   * Wizualizacje: wykresy kołowe, słupkowe, liniowe do analizy przychodów, wydatków i oszczędności w czasie.
   * Zestawienia i raporty dotyczące tego, jak użytkownik wydaje swoje pieniądze, co może pomóc w optymalizacji budżetu.
5. **Cele oszczędnościowe**
   * Użytkownicy mogą ustawiać cele oszczędnościowe (np. „Zaoszczędzić 1000 zł na wakacje”).
   * Aplikacja będzie śledzić postęp w osiąganiu celu i przypominać o oszczędzaniu.
6. **Powiadomienia i przypomnienia**
   * Powiadomienia o nadchodzących rachunkach, przekroczeniu budżetu lub osiągnięciu oszczędności.
   * Codzienne lub cotygodniowe przypomnienia o dodaniu nowych transakcji.
7. **Bezpieczeństwo danych**
   * Szyfrowanie danych użytkowników w bazie danych.
   * Możliwość logowania się przez zewnętrzne serwisy (np. Google, Facebook).
8. **Integracja z urządzeniami mobilnymi**
   * Aplikacja może posiadać wersję mobilną (np. na iOS i Androida), aby użytkownicy mogli wprowadzać transakcje w czasie rzeczywistym.

**Architektura aplikacji**

1. **Frontend**
   * **ASP.NET Core + Blazor**: Blazor pozwala tworzyć aplikacje webowe, które działają na wszystkich platformach, wykorzystując C# zarówno po stronie klienta, jak i serwera.
   * **Alternatywnie: React lub Angular z WebAPI**: Możesz zdecydować się na frontend w technologii JavaScript (np. React lub Angular) i połączyć go z backendem opartym na ASP.NET Core.
2. **Backend**
   * **ASP.NET Core Web API**: API odpowiedzialne za logikę biznesową, autoryzację użytkowników, przechowywanie transakcji i zarządzanie danymi finansowymi.
   * **Entity Framework Core**: ORM do komunikacji z bazą danych (np. SQL Server) w celu przechowywania danych o użytkownikach, transakcjach, celach oszczędnościowych itp.
   * **SignalR**: Może być użyte do dynamicznych powiadomień w czasie rzeczywistym (np. o przekroczeniu budżetu).
3. **Baza danych**
   * **SQL Server / Azure SQL Database**: Relacyjna baza danych, która przechowuje dane o użytkownikach, transakcjach, kategoriach wydatków i celach oszczędnościowych.
   * **Alternatywnie: MongoDB**: Jeśli chcesz zbudować system bardziej elastyczny i skalowalny, MongoDB może być opcją.
4. **Integracja z systemami płatności**
   * **Stripe / PayPal**: Możesz zintegrować zewnętrzne systemy płatności w celu pobierania danych o transakcjach użytkowników (np. aby automatycznie importować wydatki).
   * **API bankowe**: Użyj API do integracji z bankami i uzyskiwania transakcji bezpośrednio z kont bankowych użytkowników.

**Planowanie rozwoju aplikacji**

1. **MVP (Minimum Viable Product)**:
   * Zacznij od podstawowych funkcji: rejestracja/logowanie, dodawanie transakcji, kategoryzowanie wydatków, proste wykresy analityczne, ustawianie budżetu miesięcznego.
   * Zaimplementuj prosty interfejs użytkownika, który będzie pozwalał użytkownikom łatwo dodawać wydatki i przychody.
2. **Wersja 2.0 (rozbudowa)**:
   * Dodanie powiadomień i przypomnień.
   * Integracja z API bankowymi lub systemami płatności (np. Stripe).
   * Zwiększenie możliwości analitycznych, np. generowanie szczegółowych raportów i wykresów.
   * Wprowadzenie celów oszczędnościowych z systemem powiadomień o postępach.
3. **Wersja 3.0 (mobilna)**:
   * Stworzenie wersji mobilnej (na iOS i Androida).
   * Synchronizacja danych między wersją webową i mobilną.
   * Możliwość robienia zdjęć paragonów i dodawania ich jako transakcji.

**Potrzebne technologie i narzędzia**

* **ASP.NET Core**: Do stworzenia backendu aplikacji.
* **Blazor (opcjonalnie)**: Do tworzenia aplikacji frontendowej z C#.
* **Entity Framework Core**: Do zarządzania bazą danych.
* **SignalR**: Do powiadomień w czasie rzeczywistym.
* **SQL Server lub NoSQL (np. MongoDB)**: Do przechowywania danych.
* **Azure lub AWS**: Do hostingu aplikacji w chmurze.
* **React Native lub Xamarin (dla aplikacji mobilnej)**: Jeśli planujesz aplikację mobilną, możesz użyć Xamarin lub React Native, które pozwalają pisać aplikacje na Androida i iOS w C#.

**Przykładowe wyzwania techniczne**

1. **Bezpieczeństwo danych**:
   * Upewnienie się, że dane finansowe użytkowników są odpowiednio szyfrowane i zabezpieczone.
   * Zastosowanie OAuth2 i OpenID Connect do bezpiecznego logowania.
2. **Integracja z bankami i systemami płatności**:
   * Wymaga to pracy z API dostarczanymi przez banki i systemy płatności, co może wiązać się z dodatkowymi trudnościami związanymi z autoryzacją i synchronizowaniem danych.
3. **Zarządzanie dużą ilością danych**:
   * Dobrze zaprojektowana baza danych i indeksowanie pozwoli na efektywne zarządzanie rosnącą ilością danych o transakcjach.