Dokumentacja techniczna do aplikacji “Aplikacja do zarządzania listą zakupów dla domowników” autorstwa Jakub Wasielewski 138442 oraz Daniel Wardak 138436.

1. Tytuł projektu:

Aplikacja do zarządzania listą zakupów dla domowników

2. Krótki opis działania projektu:

Aplikacja umożliwia domownikom tworzenie i zarządzanie listami zakupów w środowisku lokalnym. Każdy użytkownik może się zarejestrować, a następnie dodawać produkty do swojej indywidualnej listy zakupów. Dorośli użytkownicy mają możliwość wglądu do pełnej listy zakupów, generowania zbiorczej listy produktów (z sumowaniem ilości tych samych produktów) oraz jej wydrukowania. Dorośli mogą również zobaczyć, które produkty były przypisane do poszczególnych użytkowników.

3. Autorzy projektu:

- Jakub Wasielewski 138442
- Daniel Wardak 138436

4. Specyfikacja wykorzystanych technologii:

- Platforma: .NET 8, ASP.NET Core:

Aplikacja będzie oparta na języku C# i stworzona przy użyciu ASP.NET Core w wersji 8, co zapewnia wsparcie dla najnowszych funkcji platformy .NET.

ORM: Entity Framework Core

- Aplikacja korzysta z Entity Framework Core jako ORM (Object-Relational Mapping) do komunikacji z bazą danych. Dzięki temu nie jest wymagane bezpośrednie pisanie zapytań SQL, a operacje na danych są wykonywane za pomocą obiektów i klas w C#.

Baza danych: Aplikacja wykorzystuje bazę danych SQLite. Plik bazy danych ShoppingApp.db znajduje się w folderze Data. Baza jest trwała i przechowuje dane użytkowników oraz list zakupów.

- Dla uproszczenia i wyeliminowania konieczności konfiguracji zewnętrznej bazy danych, aplikacja wykorzystuje wbudowaną bazę danych w pamięci (In-Memory Database) dostarczaną przez Entity Framework Core. Pozwala to na przechowywanie danych podczas działania aplikacji bez konieczności instalowania dodatkowego oprogramowania.
  - Frontend: HTML, CSS:

Formularze użytkowników, tabele do raportów oraz inne elementy interfejsu będą tworzone z wykorzystaniem HTML i CSS, co pozwala na intuicyjną prezentację danych i generowanie czytelnych raportów.

5. Instrukcje pierwszego uruchomienia projektu:

01. Skopiuj repozytorium

Skopiuj projekt z publicznego repozytorium git:
https://github.com/Dancio312/Dokumentacja-Aplikacja-do-zarz-dzania-list-zakup-w-dla-domownik-w

02. Zainstaluj wymagane oprogramowanie

Upewnij się, że masz zainstalowane:

- .NET w wersji 8 (do uruchomienia aplikacji ASP.NET Core).

03. Uruchomienie aplikacji

04. Pierwsze logowanie

Przed założeniem konta użytkownika upewnij się, że baza danych została prawidłowo przygotowana. W tym celu wykonaj następujące polecenia w terminalu w katalogu projektu:
Add-Migration InitialCreate
Update-Database
To zapewni, że struktura bazy danych jest zgodna z aktualnymi modelami.
Pierwsze logowanie
Użyj formularza rejestracji, aby utworzyć konto użytkownika i przetestować działanie aplikacji

6. Opis struktury projektu:

01. Controllers

Zawiera kontrolery odpowiedzialne za obsługę logiki aplikacji oraz zarządzanie interakcjami między widokami, modelami a ORM.

- **AccountController.cs**

Zarządza rejestracją i logowaniem użytkowników.

- **HomeController.cs**

Odpowiada za stronę główną, politykę prywatności.

- **ProductsController.cs**

Zarządzanie produktami: wyświetlanie, dodawanie, edycja, usuwanie.

- **ShoppingListsController.cs**

Zarządzanie listami zakupów: wyświetlanie, tworzenie, edycja, usuwanie oraz generowanie zbiorczych list.

02. Models

Zawiera klasy modelowe, które odwzorowują struktury danych używane w aplikacji i odpowiadają tabelą w bazie danych. Przykładowe modele:

- **ErrorViewModel.cs**

Model do obsługi błędów aplikacji.

- **Product.cs**

Model produktu.

- **RegisterViewModel.cs**

Model do obsługi rejestracji użytkowników.

- **ShoppingList.cs**

Model listy zakupów.

03. Data

Przechowuje klasę kontekstu bazy danych, np. ApplicationDbContext, która zarządza danymi aplikacji przy użyciu Entity Framework Core.

- **Migrations**

Zawiera migrację bazy danych:
20241214055640_InitialCreate.cs

- **ApplicationDbContext.cs**

Klasa kontekstu bazy danych, zawiera:
Product : DbSet<Product>
ShoppingList : DbSet<ShoppingList>

- **ShoppingApp.db**, który przechowuje dane aplikacji jest to baza danych.
- **ApplicationDbContextModelSnapshot.cs**

Odpowiada za snapshot modelu bazy danych.

04. Views

Zawiera widoki odpowiadające kontrolerom i modelom danych. Każdy widok jest odpowiedzialny za wyświetlanie danych użytkownikowi i przyjmowanie jego interakcji. Widoki są tworzone przy użyciu HTML.

- **Account**

Zawiera widok rejestracji użytkownika:
Register.cshtml

- **Home**

Widoki dla strony głównej i polityki prywatności:
Index.cshtml
Privacy.cshtml

- **Products**

Widoki do zarządzania produktami:
Create.cshtml
Delete.cshtml
Details.cshtml
Edit.cshtml
Index.cshtml

- **Shared**

Wspólne widoki i layout aplikacji:
_Layout.cshtml
_LoginPartial.cshtml
_ValidationScriptsPartial.cshtml
Error.cshtml

- **ShoppingLists**

Widoki do zarządzania listami zakupów:
Create.cshtml
Delete.cshtml
Details.cshtml
Edit.cshtml
GenerateFamilyList.cshtml
Index.cshtml

- **Globalne**

Widoki globalne dla aplikacji:
_ViewImports.cshtml

_ViewStart.cshtml

05. wwwroot

Folder ten zawiera zasoby statyczne, takie jak pliki CSS, obrazy i inne zasoby wykorzystywane w interfejsie użytkownika.

- **css**

Zawiera plik stylów aplikacji: site.css.

- **js**

Zawiera skrypt JavaScript aplikacji: site.js.

- **lib**

Zawiera biblioteki frontendu:
bootstrap – Framework CSS i JS.
jquery – Biblioteka JavaScript.
jquery-validation – Walidacja formularzy.
jquery-validation-unobtrusive – Walidacja unobtrusive.

favicon.ico

Ikona aplikacji.

06. appsettings.json

Konfiguracja bazy danych, logowania i hostów.

- **appsettings.Development.json**

Konfiguracja logowania dla środowiska deweloperskiego.

07. Program.cs

Punkt wejściowy aplikacji, który uruchamia serwer ASP.NET Core. Konfiguruje kluczowe usługi, takie jak:

- Entity Framework Core – do obsługi bazy danych.
- Identity z obsługą ról – umożliwia uwierzytelnianie użytkowników i przypisywanie ról (Parent i Child).
- Routing – definiuje domyślną trasę aplikacji.
- Ustawiono domyślną kulturę na "pl-PL" dla formatów liczbowych, dat oraz walut.

Dodatkowo konfiguruje środowiska deweloperskie i produkcyjne, obsługę błędów, HTTPS, pliki statyczne oraz uwierzytelnianie i autoryzację.

7. Wylistowane wszystkie modele, do każdego z nich krótki opis, po co dany model jest, wszystkie pola w modelach wraz z opisem:

1. ErrorViewModel

Opis: Model do obsługi błędów aplikacji.
Pola:
RequestId: string – Identyfikator żądania.
ShowRequestId: bool – Flaga wskazująca, czy wyświetlić identyfikator żądania.

2. Product

Opis: Reprezentuje produkt, który może być dodany do listy zakupów.
Pola:

Id: int – Unikalny identyfikator produktu (klucz główny).
Name: string – Nazwa produktu.
Value (decimal) – Wartość produktu, ograniczona zakresem 0,01–100,99 z formatowaniem jako waluta.

3. RegisterViewModel

Opis: Model używany do rejestracji użytkowników.
Pola:
Email: string – Adres e-mail użytkownika.
Password: string – Hasło użytkownika.
Role: string – Rola przypisana użytkownikowi (np. „Parent” lub „Child”).

4. ShoppingList

Opis: Reprezentuje listę zakupów użytkownika, łączy użytkownika z produktami.
Pola:
Id: int – Unikalny identyfikator listy zakupów (klucz główny).
ProductId: int – Klucz obcy do produktu.
Product: Product – Powiązany produkt.
UserId: string – Klucz obcy do użytkownika.
User: IdentityUser – Powiązany użytkownik.

8. **Wylistowane kontrolery wraz z metodami:** metoda musi być opisana swoją nazwą, dopuszczalnymi metodami HTTP, przyjmowanymi parametrami, opisem działania i zwracanymi danymi; w przypadku standardowych i oczywistych metod opis może być skrócony:

1.AccountController

Register() : IActionResult – Wyświetla formularz rejestracji użytkownika.
Register(RegisterViewModel) : Task<IActionResult> – Obsługuje rejestrację użytkownika na podstawie danych z RegisterViewModel.

2. HomeController

Index() : IActionResult – Wyświetla stronę główną aplikacji.
Privacy() : IActionResult – Wyświetla politykę prywatności.
Error() : IActionResult – Obsługuje błędy aplikacji.

3. ProductsController

Index() : Task<IActionResult> – Wyświetla listę produktów.
Details(int?) : Task<IActionResult> – Wyświetla szczegóły produktu na podstawie ID.
Create() : IActionResult – Wyświetla formularz tworzenia nowego produktu.
Create(Product) : Task<IActionResult> – Tworzy nowy produkt w bazie danych.
Edit(int?) : Task<IActionResult> – Wyświetla formularz edycji produktu.
Edit(int, Product) : Task<IActionResult> – Zapisuje zmiany w edytowanym produkcie.
Delete(int?) : Task<IActionResult> – Wyświetla stronę potwierdzenia usunięcia produktu.
DeleteConfirmed(int) : Task<IActionResult> – Usuwa produkt z bazy danych.

ProductExists(int) : bool – Sprawdza, czy produkt o podanym ID istnieje.

4. ShoppingListsController

Index() : Task<IActionResult> – Wyświetla listę zakupów użytkownika.
Details(int?) : Task<IActionResult> – Wyświetla szczegóły listy zakupów na podstawie ID.
Create() : IActionResult – Wyświetla formularz tworzenia nowej listy zakupów.
Create(ShoppingList) : Task<IActionResult> – Tworzy nową listę zakupów w bazie danych.
Edit(int?) : Task<IActionResult> – Wyświetla formularz edycji listy zakupów.
Edit(int, ShoppingList) : Task<IActionResult> – Zapisuje zmiany w edytowanej liście zakupów.
Delete(int?) : Task<IActionResult> – Wyświetla stronę potwierdzenia usunięcia listy zakupów.
DeleteConfirmed(int) : Task<IActionResult> – Usuwa listę zakupów z bazy danych.
ShoppingListExists(int) : bool – Sprawdza, czy lista zakupów o podanym ID istnieje.
GenerateFamilyList() : Task<IActionResult> – Generuje zbiorczą listę zakupów dla rodziny.
ClearShoppingLists() : Task<IActionResult> – Usuwa wszystkie listy zakupów.

9. **Opis systemu użytkowników:** czy i jakie role w systemie funkcjonują oraz jak nadać rolę użytkownikowi, jakie możliwości mają użytkownicy zalogowani w odróżnieniu od gości, jakie informacje są powiązane z konkretnym użytkownikiem, a jakie są globalne:

01. Role w systemie

W aplikacji do zarządzania listą zakupów funkcjonują dwie główne role użytkowników:

Użytkownik standardowy (dziecko):
Osoba, która korzysta z aplikacji do tworzenia i zarządzania własnymi listami zakupów.
Możliwości:

- Rejestracja i logowanie do systemu.
- Tworzenie własnych list zakupów.
- Dodawanie i edytowanie produktów do swoich list zakupów.
- Przeglądanie aktualnych list zakupów.
- Możliwość edytowania i usuwania swoich produktów z listy.

Użytkownik dorosły:
Osoba, która ma pełny dostęp do wszystkich funkcji aplikacji, w tym dodatkowe możliwości administracyjne.
Możliwości:

- Wszystkie funkcje dostępne dla użytkowników standardowych.
- Przeglądanie list zakupów wszystkich użytkowników w rodzinie.
- Generowanie, drukowanie i usuwanie zbiorczych list zakupów.

02. Nadawanie ról użytkownikom

Rola użytkownika jest przypisywana podczas rejestracji.
System użytkowników w aplikacji działa w oparciu o Microsoft.Identity. Dodatkowo, aplikacja wykorzystuje własny model i kontroler, które umożliwiają rejestrację użytkowników z wyborem roli jako „Rodzic" lub „Dziecko". Dzięki temu role użytkowników są dostosowane do ich funkcjonalności w aplikacji.

03. Możliwości użytkowników

Zalogowani użytkownicy:

- Mogą zarządzać swoimi listami zakupów, dodawać, edytować i usuwać produkty.
- Mogą przeglądać listy zakupów innych użytkowników (w przypadku użytkowników dorosłych).
- Mogą generować zbiorcze listy zakupów do wydrukowania (użytkownicy dorośli).

Goście:

- Nie mają dostępu do żadnych funkcji aplikacji.
- Nie mogą przeglądać list zakupów ani dodawać produktów.
- Muszą zarejestrować się i zalogować, aby uzyskać dostęp do funkcjonalności aplikacji.

04. Informacje powiązane z użytkownikami

Specyficzne dla użytkownika:

- Listy zakupów, które są przypisane do danego użytkownika.
- Rola przypisana do użytkownika.

Informacje globalne:

- Produkty, które są dostępne do dodania na listy zakupów.
- Zbiorcza lista zakupów (dla dorosłych).

10. Krótka charakterystyka najciekawszych funkcjonalności:

01. Zarządzanie listami zakupów:

- Użytkownicy mogą tworzyć i zarządzać listą zakupów. Funkcjonalność ta umożliwia przypisywanie produktów do listy, co ułatwia organizację zakupów i pozwala na szybsze planowanie.

02. Wielu użytkowników:

- Aplikacja umożliwia korzystanie z jednej instancji przez różnych użytkowników w ramach jednej rodziny.

03. Generowanie zbiorczych list zakupów:

- Użytkownicy dorośli mają możliwość generowania i drukowania zbiorczych list zakupów, które zawierają wszystkie potrzebne produkty, zliczając ilości z różnych list. Dzięki temu użytkownik ma podgląd, co trzeba kupić bez zbędnych duplikatów.

04. Prosta nawigacja i interfejs:

- Aplikacja będzie posiadać przejrzysty interfejs użytkownika, który pozwala na łatwe dodawanie produktów i edytowanie list. Użytkownicy będą mogli korzystać z intuicyjnych formularzy HTML oraz responsywnych widoków.

05. Walidacja danych:

- Podczas dodawania i edytowania produktów użytkownicy będą informowani o błędach (np. brak wymaganych pól czy niewłaściwy format danych). Dzięki temu proces dodawania produktów będzie bardziej zautomatyzowany i mniej podatny na błędy.

06. Możliwość modyfikacji i usuwania:

- Użytkownicy będą mieli możliwość edytowania istniejących produktów oraz usuwania ich z list zakupów, co daje pełną kontrolę nad tym, co jest aktualnie planowane do zakupu.

07. Brak konieczności instalacji zewnętrznych baz danych

- Dzięki wykorzystaniu Entity Framework Core z In-Memory Database, aplikacja nie wymaga instalacji ani konfiguracji zewnętrznych serwerów baz danych.
- Uproszczone uruchomienie aplikacji.
- Łatwiejsze testowanie i rozwijanie aplikacji bez dodatkowych zależności.