

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Dokumentacja

Autor:
Michał Bernardy
Adrian Gargula

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2024

Spis treści

1. Ogólne określenie wymagań	4
1.1. Konto	4
1.2. Logowanie biometryczne	4
1.3. Dokumentacja zakupów	5
1.4. Powiadomienia	5
1.5. Funkcjonalność aplikacji	5
1.6. Tryb offline	6
1.7. Baza danych	6
1.8. Design logo	6
2. Określenie wymagań szczegółowych	7
2.1. Specyfikacja techniczna .NET MAUI	7
2.2. Specyfikacja techniczna Aplikacji CYD	7
2.3. Funkcje i Moduły	8
2.3.1. Logowanie i Weryfikacja	8
2.3.2. Powiadomienia push	8
2.3.3. Dokumentacja wydatków	8
2.3.4. Kategorie wydatków	8
2.3.5. Miesięczne zestawienie	8
2.3.6. Baza Danych	9
3. Projektowanie	10
3.1. Przygotowanie Narzędzi do Tworzenia Aplikacji "CYD" w .NET MAUI . .	10
3.1.1. Środowisko programistyczne .NET MAUI	10
3.1.2. Visual Studio	10
3.1.3. Git	10
3.1.4. Emulator	11
3.1.5. Tworzenie Nowego Projektu .NET MAUI	11
3.2. Szkic layouta	12
3.2.1. Ekran Logowania	12
3.2.2. Ekran Rejestrowania	13

4. Implementacja	14
4.1. Klasa AppShell	14
4.1.1. Wyjaśnienie kodu	16
4.2. XAML Strony Logowania	18
4.2.1. Wyjaśnienie kodu XAML	21
4.2.2. Funkcjonalność	22
4.3. Implementacja klasy FirebaseAuthService	22
4.3.1. Wyjaśnienie kodu	23
4.4. Klasa SignUpPage	25
4.4.1. Wyjaśnienie kodu	27
4.5. Klasa CoinGeckoService	29
4.5.1. Wyjaśnienie kodu	30
5. Testowanie	33
5.1. Logowanie	33
5.2. Rejestracja	34
5.3. Resetowanie hasła	35
5.4. Menu boczne	35
5.5. Kryptowaluty	35
5.6. Kurs walut	36
5.7. Sensor światła	36
5.8. Dane logowania	37
5.9. Wydatki	38
6. Podręcznik użytkownika	39
6.1. Logowanie i rejestracja	39
6.2. Korzystanie z aplikacji	43
Literatura	51
Spis rysunków	52
Spis tabel	53
Spis listingów	54

1. Ogólne określenie wymagań

Aplikacja mobilna **CountYourDollars** w skrócie ”**CYD**” ma oferować użytkownikom łatwą i szybką możliwość zarządzania i monitorowania miesięcznych wydatków użytkownika. Narzędzie jest wyposażone w wiele funkcji takich jak logowanie, dokumentowanie zakupów, miesięczne podsumowanie wydatków oraz powiadomienia. Dzięki prostemu i intuicyjnemu interfejsowi, aplikacja jest przyjazna dla wszystkich użytkowników, w tym osób starszych, którzy mogą korzystać z niej bez obaw o trudności w obsłudze. **CountYourDollars** to kompleksowe rozwiązanie, które wspiera użytkowników w efektywnym zarządzaniu finansami osobistymi dzięki czemu mogą oni zapomnieć o magicznie znikających pieniądzach.

1.1. Konto

Aby skorzystać z pełnych możliwości aplikacji **CountYourDollars**, użytkownik musi najpierw założyć konto. Proces rejestracji jest prosty i intuicyjny, co pozwala na szybkie rozpoczęcie korzystania z narzędzia. Utworzenie konta gwarantuje bezpieczeństwo danych użytkownika, zapewniając, że wszystkie informacje finansowe oraz osobiste są odpowiednio chronione.

1.2. Logowanie biometryczne

CountYourDollars zapewnia najwyższy poziom bezpieczeństwa, umożliwiając użytkownikom logowanie za pomocą danych biometrycznych, takich jak odciski palców. To innowacyjne rozwiązanie gwarantuje, że dostęp do wrażliwych informacji finansowych jest ściśle ograniczony do uprawnionych użytkowników, co znacząco minimalizuje ryzyko nieautoryzowanego dostępu. Wprowadzenie logowania biometrycznego nie tylko podnosi standardy bezpieczeństwa, ale także znacząco zwiększa wygodę korzystania z aplikacji. Dzięki szybkiej i intuicyjnej metodzie uwierzytelniania użytkownicy mogą łatwo i bezpiecznie uzyskać dostęp do swojego konta, eliminując konieczność zapamiętywania skomplikowanych haseł.

1.3. Dokumentacja zakupów

Aplikacja będzie wymagała od użytkowników dołączenia dokumentu potwierdzającego wydatek. Dzięki temu każdy zapis będzie dokładnie udokumentowany, co pozwoli na lepszą kontrolę finansów oraz umożliwi łatwiejsze śledzenie wydatków. Użytkownicy będą mieli możliwość załączania zdjęć paragonów lub innych dowodów zakupu, co zapewni pełną transparentność i wiarygodność w zarządzaniu osobistymi finansami.

1.4. Powiadomienia

Oprogramowanie ma wbudowany system powiadomień push, który na bieżąco informuje użytkowników o zbliżaniu się do indywidualnie ustalonego limitu wydatków. Dzięki możliwości dostosowania limitu do własnych potrzeb, użytkownicy mogą efektywnie zarządzać swoim budżetem, a system powiadomień przypomina im o nadchodzących ograniczeniach. W ten sposób **CountYourDollars** nie tylko pomaga w monitorowaniu wydatków, ale także wspiera w osiąganiu finansowych celów, oferując na czas przypomnienia, które pozwalają na lepszą kontrolę nad osobistymi finansami.

1.5. Funkcjonalność aplikacji

Na początku korzystania z aplikacji użytkownik wprowadza swój miesięczny dochód, co stanowi fundament efektywnego zarządzania finansami. Interfejs aplikacji został zaprojektowany w sposób intuicyjny, umożliwiając łatwe przechodzenie między poszczególnymi miesiącami. Po kliknięciu na konkretny miesiąc, użytkownicy zyskują pełny wgląd w swoje wydatki, co pozwala na szczegółową analizę finansową. Na zakończenie każdego miesiąca aplikacja generuje podsumowanie wydatków, prezentując użytkownikowi informacje o tym, ile zostało wydane na różne kategorie, takie jak samochód, jedzenie, rozrywka, czynsz i inne. Dzięki temu użytkownicy mają możliwość skutecznego monitorowania swoich wydatków oraz podejmowania świadomych decyzji finansowych, co przyczynia się do lepszego zarządzania budżetem.

1.6. Tryb offline

CountYourDollars została zaprojektowana z myślą o zapewnieniu użytkownikom maksymalnej wygody i dostępności, nawet w sytuacjach, gdy nie mają połączenia internetowego. Funkcjonalność trybu offline oferuje użytkownikom podgląd do wydatków w obecnym miesiącu.

1.7. Baza danych

Aplikacja korzysta z chmurowej bazy danych Firebase, co pozwala na skuteczne przechowywanie, zarządzanie i dostęp do różnorodnych informacji związanych z finansami użytkowników. Firebase jest wszechstronnym rozwiązaniem, które oferuje wiele funkcji niezbędnych do prawidłowego funkcjonowania aplikacji.

1.8. Design logo



Rys. 1.1. Logo

Logo przedstawia symboliczny emblemat w kształcie okręgu z wyraźnym napisem "CYD" w jego wnętrzu. Dominująca kolorystyka w odcieniach niebieskiego nadaje mu nowoczesny i profesjonalny wygląd, co jest często kojarzone z technologią i finansami. Napis "COUNTYOURDOLLARS" poniżej skrótu to nazwa aplikacji która już na wstępie sugeruje że będzie ona służyć do zarządzania finansami. Cały projekt jest schludny i estetyczny, wykorzystując prostą, ale elegancką typografię.

2. Określenie wymagań szczegółowych

2.1. Specyfikacja techniczna .NET MAUI

- Język programowania: .NET MAUI pozwala programować w języku C#.
- Wspierane platformy: .NET MAUI umożliwia tworzenie aplikacji na platformy iOS, Android, macOS oraz Windows.
- Jednolity kod interfejsu: Dzięki .NET MAUI można tworzyć interfejsy użytkownika za pomocą wspólnego kodu dla wielu platform.
- Natywny dostęp: Pozwala na dostęp do natywnych API oraz funkcji systemowych każdej z platform przy użyciu specyficznych narzędzi i komponentów.
- Wspólne API: .NET MAUI oferuje dostęp do wspólnego zestawu API, co pozwala na udostępnienie logiki biznesowej między platformami.
- Narzędzia IDE: Do pracy z .NET MAUI można wykorzystać środowisko Visual Studio 2022 oraz Visual Studio for Mac.
- Biblioteki: .NET MAUI integruje się z ekosystemem .NET, umożliwiając dostęp do szerokiej gamy bibliotek oraz narzędzi, takich jak NuGet.
- Testowanie: Możliwość testowania aplikacji zarówno na rzeczywistych urządzeniach, jak i przy użyciu emulatorów oraz narzędzi takich jak .NET MAUI Test. ^[1]

2.2. Specyfikacja techniczna Aplikacji CYD

Aplikacja CYD to kompleksowe narzędzie stworzone w technologii .NET MAUI, dostępne na platformy Android, które ma na celu ułatwienie życia oraz dostarczenie istotnych informacji związanych z wydatkami. Aplikacja będzie wyposażona w funkcje powiadomień push, logowanie biometryczne, kategoryzowanie wydatków, podsumowywanie miesięczne oraz dokumentowanie wydatków poprzez zdjęcie. Poniżej przedstawiamy bardziej szczegółowe wymagania dotyczące każdej z tych funkcji

¹Programowanie urządzeń mobilnych [wykłady]

2.3. Funkcje i Moduły

2.3.1. Logowanie i Weryfikacja

- Aplikacja CYD umożliwia użytkownikom zalogowanie się za pomocą unikatowego identyfikatora oraz hasła, a także wykorzystanie zaawansowanych mechanizmów logowania biometrycznego, takich jak rozpoznawanie odcisków palców .
- W trakcie procesu logowania następuje weryfikacja tożsamości użytkownika, co zapewnia bezpieczeństwo danych oraz dostęp tylko do odpowiednich zasobów

2.3.2. Powiadomienia push

- Funkcja "Powiadomienia push" powiadamia użytkowników o kończącym się budżecie w danym miesiącu.
- Powiadomienia są wysyłane po przekroczeniu danych limitów ustawionych przez użytkownika.

2.3.3. Dokumentacja wydatków

- Każdy wydatek może być potwierdzany zdjęciem paragonu.
- Dzięki temu użytkownicy mogą dokładnie uwiecznić swoje wydatki nie musząc zbierać paragonów.

2.3.4. Kategorie wydatków

- Kategoryzowanie pozwala na podział obciążeń, użytkownik może sprawdzać jakie produkty pochłaniają daną ilość pieniędzy w skali miesiąca.

2.3.5. Miesięczne zestawienie

- Aplikacja dostarcza informacje na miesięcznego zestawienia wydatków.
- Dane do zestawienia są pobierane z bazy danych i prezentowane w tabeli z sumą wydatków podzieloną na kategorię.

2.3.6. Baza Danych

- Aplikacja korzysta z bazy danych do przechowywania informacji o użytkownikach, wydatkach i innych danych.
- Baza danych jest odpowiednio zabezpieczona i regularnie tworzone są jej kopie zapasowe, aby zapewnić integralność i dostępność danych.
- W aplikacji będzie wykorzystana baza danych Google Firebase. Firebase oferuje bazę danych typu NoSQL o nazwie Firebase Realtime Database oraz Firebase Firestore. Oba rozwiązania są skalowalne i oferują możliwość współpracy w czasie rzeczywistym.².

²Google Firebase [1].

3. Projektowanie

3.1. Przygotowanie Narzędzi do Tworzenia Aplikacji "CYD" w .NET MAUI

Tworzenie aplikacji mobilnych jest zadaniem wymagającym odpowiednich narzędzi i środowiska programistycznego. W przypadku aplikacji "CYD" w języku .NET MAUI, musimy odpowiednio skonfigurować narzędzia, aby móc sprawnie rozwijać projekt. W poniższym opisie przedstawiamy kroki przygotowania narzędzi, aby ułatwić proces tworzenia aplikacji dla platformy Android.

3.1.1. Środowisko programistyczne .NET MAUI

Rozpoczynamy od zainstalowania środowiska .NET MAUI na naszym komputerze. .NET MAUI to nowoczesne narzędzie umożliwiające tworzenie aplikacji wieloplatformowych z wykorzystaniem języka C oraz jednolitego kodu dla iOS, Androida, macOS i Windows.

3.1.2. Visual Studio

Następnie, jeśli nie mamy jeszcze zainstalowanego Visual Studio, pobieramy i instalujemy najnowszą wersję programu. Podczas instalacji zaleca się zaznaczenie opcji "Mobile development with .NET", co pozwoli na aktywację wsparcia dla .NET MAUI. To środowisko programistyczne dostarcza narzędzi do projektowania interfejsu, edycji kodu, debugowania i testowania aplikacji.

3.1.3. Git

Git to system kontroli wersji. Git to system, który pozwala programistom zapisywać wszystkie zmiany w pisanym kodzie – w taki sposób, aby niczego nie stracili. Programista w każdej chwili może wrócić do poprzedniej wersji, jeżeli zajdzie taka potrzeba. Dzięki podglądowi poprzednich wersji można prześledzić, jak program się rozwijał, cofnąć się, odzyskać przypadkowo utracone zmiany czy powrócić do wcześniejszych pomysłów. Z systemu kontroli wersji Git korzystają zarówno programiści w korporacyjnych zespołach, jak i ci pracujący samodzielnie. Warto wyrobić sobie ten dobry nawyk korzystania z systemu kontroli wersji – nie raz okaże się, że uratuje nam to skórę i zapobiegnie marnowaniu czasu³.

³CODECOOL [www\[2\]](#).

3.1.4. Emulator

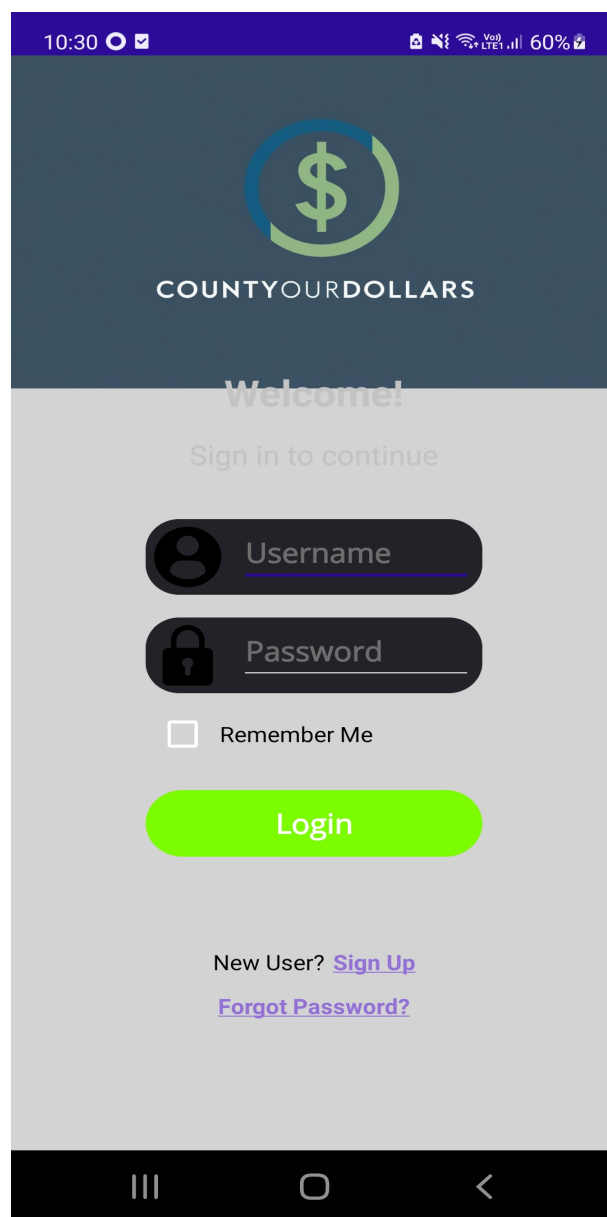
Warto także zadbać o narzędzia potrzebne podczas tworzenia aplikacji .NET MAUI. Przydatnym narzędziem jest emulator Androida do testowania aplikacji na różnych urządzeniach oraz fizyczne urządzenie Android.

3.1.5. Tworzenie Nowego Projektu .NET MAUI

Po zainstalowaniu narzędzi i skonfigurowaniu środowiska, możemy przejść do tworzenia nowego projektu .NET MAUI. W Visual Studio wybieramy opcję utworzenia nowego projektu .NET MAUI App. Możemy wybrać odpowiedni szablon aplikacji, który najlepiej odpowiada naszym potrzebom, na przykład Blank App, .NET MAUI Blazor App lub inne dostępne opcje. Szablony te dostarczają podstawową strukturę aplikacji i pozwalają na łatwe dodawanie nowych widoków i funkcjonalności.

3.2. Szkic layouta

3.2.1. Ekran Logowania

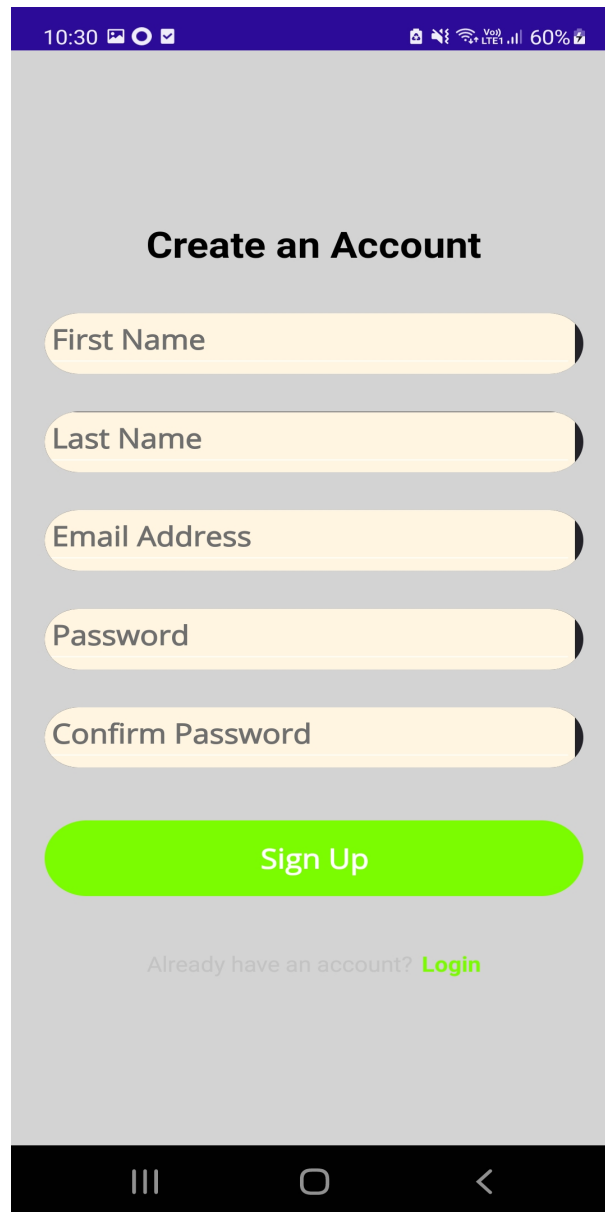


Rys. 3.1. Ekran logowania

Ń

Na rysunku.3.1(s.12). Użytkownik zobaczy ekran po uruchomieniu aplikacji. Na górze ekranu zŃnajduje się logo aplikacji. Poniżej logotypu jest formularz logowania, z polami na wprowadzenie loginu użytkownika i hasła oraz przyciskiem "Login" oraz "Sign Up" czyli logowanie oraz rejestrowanie się jest także opcja "Forgot Password?" czyli odzyskiwanie hasła. Opcja "Remember me" pozwala na zapamiętywanie użytkownika.

3.2.2. Ekran Rejestrowania



Rys. 3.2. Ekran Sign Up

Na rysunku.3.2(s.13). Użytkownik zobaczy ekran po kliknięciu sing up na stronie logowania. Na górze ekranu znajduje się informacja co użytkownik robi w danym momencie . Poniżej jest formularz rejestrowania, z niezbędnymi do utworzenia konta danymi. Gdy wypełnimy formularz możemy kliknąć "Sign Up" dzięki czemu utworzymy konto.

4. Implementacja

Rozdział "Implementacja" zawiera szczegółowe wyjaśnienie kodu źródłowego aplikacji, który obejmuje różne klasy i metody używane do realizacji funkcji aplikacji. Opisano w nim zarówno sposób implementacji interfejsu użytkownika, jak i mechanizmy autoryzacji użytkownika oraz zarządzania stanem aplikacji. Zawiera również omówienie kluczowych funkcjonalności, takich jak zmiana motywu aplikacji oraz wylogowanie użytkownika. Ponadto przedstawiono sposób, w jaki aplikacja zarządza interakcjami użytkownika i realizuje różne operacje związane z logowaniem, rejestracją i odzyskiwaniem hasła.

4.1. Klasa AppShell

Klasa AppShell 1 (s. 14) jest centralnym elementem aplikacji w frameworku .NET MAUI, pełniąc rolę głównej ramki interfejsu użytkownika. Dziedziczy po klasie Shell, która zapewnia podstawową strukturę aplikacji, w tym nawigację i zarządzanie widokami. AppShell pozwala na konfigurację globalnych ustawień aplikacji, takich jak motyw (ciemny/lekki), oraz integruje funkcje logowania i wylogowywania użytkownika. W praktyce jest to klasa, która odpowiada za prezentację i zarządzanie głównymi ekranami aplikacji oraz obsługę interakcji użytkownika.

```
1 using CommunityToolkit.Maui.Alerts;
2 using CommunityToolkit.Maui.Core;
3 using Font = Microsoft.Maui.Font;
4 using Firebase.Auth;
5 namespace CYD
6 {
7     public partial class AppShell : Shell
8     {
9         private FirebaseAuthService _authService;
10
11         public AppShell()
12         {
13             InitializeComponent();
14             _authService = new FirebaseAuthService();
15             var currentTheme = Application.Current!.UserAppTheme;
16             ThemeSegmentedControl.SelectedIndex = currentTheme ==
AppTheme.Light ? 0 : 1;
17         }
18
19         public static async Task DisplaySnackbarAsync(string
message)
```

```
20     {
21         CancellationTokenSource cancellationTokenSource = new
CancellationTokenSource();
22
23         var snackbarOptions = new SnackbarOptions
24         {
25             BackgroundColor = Color.FromArgb("#FF3300"),
26             TextColor = Colors.White,
27             ActionButtonTextColor = Colors.Yellow,
28             CornerRadius = new CornerRadius(0),
29             Font = Font.SystemFontOfSize(18),
30             ActionButtonFont = Font.SystemFontOfSize(14)
31         };
32
33         var snackbar = Snackbar.Make(message, visualOptions:
snackbarOptions);
34
35         await snackbar.Show(cancellationTokenSource.Token);
36     }
37
38     public static async Task DisplayToastAsync(string message)
39     {
40         // Toast is currently not working in MCT on Windows
41         if (OperatingSystem.IsWindows())
42             return;
43
44         var toast = Toast.Make(message, textSize: 18);
45
46         var cts = new CancellationTokenSource(TimeSpan.
FromSeconds(5));
47         await toast.Show(cts.Token);
48     }
49
50     private void SfSegmentedControl_SelectionChanged(object
sender, Syncfusion.Maui.Toolkit.SegmentedControl.
SelectionChangedEventArgs e)
51     {
52         Application.Current!.UserAppTheme = e.NewIndex == 0 ?
AppTheme.Light : AppTheme.Dark;
53     }
54
55     private async void OnLogoutButtonClicked(object sender,
EventArgs e)
56     {
57         await _authService.LogoutAsync();
```

```

58
59         // Przejście do ekranu logowania po wylogowaniu
60         Application.Current.MainPage = new NavigationPage(new
LoginPage());
61         SecureStorage.Remove("user_email");
62         SecureStorage.Remove("user_uid");
63     }
64 }
65 }

```

Listing 1. Klasa AppShell

4.1.1. Wyjaśnienie kodu

- Deklaracja klasy **AppShell** na Listing 3 (s. 16):

```

1     public partial class AppShell : Shell
2

```

Listing 2. Deklaracja klasy

Klasa **AppShell** dziedziczy po klasie **Shell** z frameworka .NET MAUI i jest odpowiedzialna za główny interfejs aplikacji. Zawiera ona funkcjonalności związane z autoryzacją użytkownika oraz obsługą interfejsu użytkownika.

- Konstruktor klasy **AppShell()** na Listing 3 (s. 16):

```

1     public AppShell()
2     {
3         InitializeComponent();
4         _authService = new FirebaseAuthService();
5         var currentTheme = Application.Current!.UserAppTheme;
6         ThemeSegmentedControl.SelectedIndex = currentTheme ==
AppTheme.Light ? 0 : 1;
7     }
8

```

Listing 3. Konstruktor klasy

Konstruktor klasy **AppShell** inicjalizuje komponenty strony przy pomocy **InitializeComponent()** oraz ustawia bieżący motyw aplikacji (jasny lub ciemny) na podstawie ustawień systemowych. Używa do tego obiektu **UserAppTheme**.

- Metoda **DisplaySnackBarAsync** na Listing 4 (s. 17):


```

1      public static async Task DisplaySnackbarAsync(string
      message)
2

```

Listing 4. Metoda DisplaySnackbarAsync

Ta statyczna metoda wyświetla powiadomienie typu snackbar z określonym komunikatem. Można dostosować wygląd snackbar, np. kolor tła, kolor tekstu, czcionkę itp. Powiadomienie jest wyświetlane przez określony czas, aż użytkownik je zamknie lub upłynie czas oczekiwania.

- Metoda DisplayToastAsync na Listing 5 (s. 17):

```

1      public static async Task DisplayToastAsync(string message)
2

```

Listing 5. Metoda DisplayToastAsync

Metoda ta wyświetla powiadomienie typu toast, które pojawia się na krótki czas na ekranie i znika automatycznie. Jeśli aplikacja działa na systemie Windows, ta funkcjonalność jest wyłączona, ponieważ obecnie nie działa ona poprawnie na tej platformie.

- Zmiana motywu aplikacji na Listing 6 (s. 17):

```

1      private void SfSegmentedControl_SelectionChanged(object
      sender, Syncfusion.Maui.Toolkit.SegmentedControl.
      SelectionChangedEventArgs e)
2      {
3          Application.Current!.UserAppTheme = e.NewIndex == 0 ?
      AppTheme.Light : AppTheme.Dark;
4      }
5

```

Listing 6. Zmiana motywu aplikacji

Ta metoda jest wywoływana po zmianie wyboru w kontrolce **SegmentedControl**. Na podstawie nowego wyboru (indeksu) motyw aplikacji jest zmieniany na jasny lub ciemny.

- Metoda OnLogoutButtonClicked na Listing 7 (s. 17):

```

1      private async void OnLogoutButtonClicked(object sender,
      EventArgs e)
2      {
3          await _authService.LogoutAsync();
4

```

```

5      // Przejście do ekranu logowania po wylogowaniu
6      Application.Current.MainPage = new NavigationPage(new
LoginPage());
7      SecureStorage.Remove("user_email");
8      SecureStorage.Remove("user_uid");
9  }
10

```

Listing 7. Metoda OnLogoutButtonClicked

Metoda **OnLogoutButtonClicked** jest wywoływana po kliknięciu przycisku wylogowania. Używa ona metody **LogoutAsync** z klasy **FirebaseAuthService** do wylogowania użytkownika. Następnie, użytkownik zostaje przekierowany do ekranu logowania, a dane dotyczące użytkownika są usuwane z **SecureStorage**.

4.2. XAML Strony Logowania

Strona logowania XAML ?? (s. ??) jest odpowiedzialna za prezentację interfejsu użytkownika, który umożliwia użytkownikom wprowadzenie danych do logowania, takich jak adres e-mail i hasło. Jest to widok, który służy jako punkt wejścia do aplikacji, umożliwiając użytkownikowi autoryzację przed uzyskaniem dostępu do innych funkcji. Strona ta jest zaprojektowana w technologii XAML, co umożliwia łatwą konfigurację i dostosowanie wyglądu interfejsu do różnych platform. Zawiera elementy interaktywne, takie jak pola tekstowe do wprowadzania danych oraz przycisk logowania, które po kliknięciu uruchamia proces weryfikacji użytkownika.

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
3     xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4     x:Class="CYD.Pages.LoginPage"
5     Title="LoginPage"
6     NavigationPage.HasNavigationBar="False">
7     <Grid BackgroundColor="LightGray">
8         <Grid.RowDefinitions>
9             <RowDefinition Height="220"/>
10            <RowDefinition Height="*/>
11            <RowDefinition Height="40"/>
12        </Grid.RowDefinitions>
13
14        <StackLayout Grid.Row="0" HeightRequest="335">
15            <Image Source="Resources/Images/logo.png"
16                HorizontalOptions="FillAndExpand"

```

```

17         VerticalOptions="FillAndExpand"
18         Aspect="AspectFill"/>
19     </StackLayout>
20
21     <Grid RowSpacing="5" Grid.Row="1" Margin="20,20,20,0">
22         <Grid.RowDefinitions>
23             <RowDefinition Height="*" />
24             <RowDefinition Height="50" />
25             <RowDefinition Height="50" />
26             <RowDefinition Height="Auto" />
27             <RowDefinition Height="40" />
28             <RowDefinition Height="40" />
29             <RowDefinition Height="*" />
30         </Grid.RowDefinitions>
31
32         <Label Grid.Row="1" Text="Welcome!"
33             HorizontalOptions="Center"
34             FontSize="Title"
35             FontAttributes="Bold"
36             Padding="10"
37             Margin="0" />
38         <Label Grid.Row="2" Text="Sign in to continue"
39             HorizontalOptions="Center"
40             FontSize="Medium" />
41
42         <StackLayout Grid.Row="3" Orientation="Vertical"
43             HorizontalOptions="CenterAndExpand">
44             <Border BackgroundColor="Transparent" Stroke="
45                 LightGreen" Padding="0" HorizontalOptions="FillAndExpand">
46                 <StackLayout Orientation="Horizontal"
47                     HorizontalOptions="FillAndExpand" VerticalOptions="
48                         CenterAndExpand" Spacing="5">
49                     <Border BackgroundColor="SkyBlue"
50                         HeightRequest="40" WidthRequest="40" Padding="0" Margin="5">
51                         <Image Source="Resources/Images/
52                             profile_user.png" Aspect="AspectFill" Margin="0" />
53                     </Border>
54                     <Entry x:Name="EmailEntry" Placeholder="
55                         Email" TextColor="Black" FontAttributes="Bold" VerticalOptions="
56                             Center"
57                         HorizontalOptions="FillAndExpand"
58                         HeightRequest="40" Margin="0,0,5,0" />
59                 </StackLayout>
60             </Border>

```

```

53         <Border BackgroundColor="Transparent" Stroke="
LightGreen" Margin="0,15,0,0" Padding="0" HorizontalOptions="
FillAndExpand">
54             <StackLayout Orientation="Horizontal"
HorizontalOptions="FillAndExpand" VerticalOptions="
CenterAndExpand" Spacing="5">
55                 <Border BackgroundColor="SkyBlue"
HeightRequest="40" WidthRequest="40" StrokeShape="RoundRectangle
" Padding="0" Margin="5">
56                     <Image Source="Resources/Images/
password.png" Aspect="AspectFill" Margin="0"/>
57                 </Border>
58                 <Entry x:Name="PasswordEntry" Placeholder="
Password" IsPassword="True" HorizontalOptions="FillAndExpand"
59                     VerticalOptions="Center" TextColor="
Black" FontAttributes="Bold" HeightRequest="40" Margin="0,0,5,0"
/>
60             </StackLayout>
61         </Border>
62
63         <StackLayout Orientation="Horizontal" Margin="
0,5,0,0" Padding="0">
64             <CheckBox IsChecked="False"/>
65             <Label Text="Remember Me" TextColor="Black"
FontSize="Small" VerticalTextAlignment="Center"
HorizontalTextAlignment="Center"/>
66         </StackLayout>
67
68         <Button Text="Login" Clicked="OnLoginButtonClicked"
BackgroundColor="LawnGreen" TextColor="White" FontAttributes="
Bold"
69             CornerRadius="30" WidthRequest="200" Margin
="0,15,0,0"/>
70
71         <StackLayout Orientation="Horizontal"
HorizontalOptions="Center" VerticalOptions="Center" Margin="
0,60,0,0" Padding="10,0" Spacing="5">
72             <Label Text="New User?" TextColor="Black"
FontSize="Small" VerticalOptions="Center" HorizontalOptions="
Start"/>
73             <Label Text="Sign Up" TextColor="MediumPurple"
FontAttributes="Bold" TextDecorations="Underline" FontSize="
Small"
74                 VerticalOptions="Center"
HorizontalOptions="Start">

```

```

75         <Label.GestureRecognizers>
76             <TapGestureRecognizer Tapped="
OnSignUpTapped"/>
77         </Label.GestureRecognizers>
78     </Label>
79 </StackLayout>
80
81     <StackLayout Orientation="Horizontal"
HorizontalOptions="Center" VerticalOptions="Center" Margin="
0,10,0,0" Padding="10,0" Spacing="5">
82         <Label Text="Forgot Password?" TextColor="
MediumPurple" FontSize="Small" FontAttributes="Bold"
TextDecorations="Underline"
83             VerticalOptions="Center"
HorizontalOptions="Center">
84             <Label.GestureRecognizers>
85                 <TapGestureRecognizer Tapped="
OnForgotPasswordTapped"/>
86             </Label.GestureRecognizers>
87         </Label>
88     </StackLayout>
89
90 </StackLayout>
91 </Grid>
92 </Grid>
93 </ContentPage>

```

Listing 8. XAML Strony Logowania

4.2.1. Wyjaśnienie kodu XAML

- **Struktura Layoutu:** Strona logowania jest oparta na układzie **Grid**, który dzieli ekran na trzy główne części: nagłówek, sekcję logowania oraz stopkę.
 - Wiersz pierwszy (**Grid.Row="0"**) zawiera logo aplikacji.
 - Wiersz drugi (**Grid.Row="1"**) zawiera pola logowania oraz inne elementy interfejsu, takie jak przyciski i pola wyboru.
 - Wiersz trzeci (**Grid.Row="2"**) jest przeznaczony na przyciski oraz dodatkowe informacje o rejestracji i odzyskiwaniu hasła.
- **Logo Aplikacji:** Logo znajduje się w górnej części strony, a jego źródło to `Resources/Images/logo.png`. Jest ono wyświetlane w sposób responsywny,

wypełniając dostępne miejsce, dzięki ustawieniu `HorizontalOptions="FillAndExpand"` oraz `VerticalOptions="FillAndExpand"`.

- **Formularz logowania:** Formularz składa się z dwóch głównych pól wejściowych: jedno dla adresu e-mail oraz drugie dla hasła. Każde z tych pól znajduje się w **StackLayout**, który zawiera również ikony (np. ikona użytkownika przy polu e-mail i ikona kłódki przy polu hasła).
 - `Entry` to pole wprowadzania danych, atrybut `IsPassword="True"` dla pola hasła zapewnia ukrycie tekstu.
 - `Border` wokół każdego pola wejściowego daje efekt konturu.
- **Opcje logowania i rejestracji:** Poza polami logowania, aplikacja oferuje możliwość zapamiętania danych użytkownika (z pomocą `CheckBox`), a także przyciski:
 - `Button` - przycisk logowania.
 - `Label` z gestem dotknięcia, który przekierowuje użytkownika do strony rejestracji (`OnSignUpTapped`).
 - `Label` z gestem dotknięcia, który umożliwia odzyskiwanie hasła (`OnForgotPasswordTapped`).

4.2.2. Funkcjonalność

- **OnLoginButtonClicked:** Metoda ta zostanie wywołana po kliknięciu przycisku logowania. Odpowiada za wykonanie logowania, weryfikację danych użytkownika i dalsze przekierowanie w aplikacji. - **OnSignUpTapped:** Akcja odpowiedzialna za przekierowanie do strony rejestracji. - **OnForgotPasswordTapped:** Akcja odpowiedzialna za przekierowanie do strony odzyskiwania hasła.

4.3. Implementacja klasy `FirebaseAuthService`

Klasa `FirebaseAuthService` 9 (s. 23) odpowiada za integrację aplikacji z usługą `Firebase Authentication`, umożliwiającą zarządzanie procesami logowania, rejestracji i wylogowywania użytkowników. Jest to klasa, która zapewnia bezpieczne uwierzytelnianie użytkowników, umożliwiając aplikacji komunikację z `Firebase` w celu weryfikacji danych logowania oraz zarządzania sesjami użytkowników. Implementacja tej klasy zawiera metody do logowania, rejestracji nowych użytkowników, a także funkcje do zarządzania tokenami uwierzytelniającymi. Dzięki tej klasie aplikacja może zapewnić bezpieczny dostęp do swoich zasobów i umożliwić użytkownikom dostęp tylko po pomyślnym zalogowaniu.

```

1 public class FirebaseAuthService
2 {
3     private readonly FirebaseAuthProvider _authProvider;
4     private readonly string _firebaseApiKey = "
AlzaSyCsPbeDGIi0slQWhIcPr2hI0NShLn7dE7M";
5     private FirebaseAuthLink _authLink;
6
7     public FirebaseAuthService()
8     {
9         _authProvider = new FirebaseAuthProvider(new FirebaseConfig
(_firebaseApiKey));
10    }
11
12    public async Task<string> SignInWithEmailAndPasswordAsync(
string email, string password)
13    {
14        try
15        {
16            _authLink = await _authProvider.
SignInWithEmailAndPasswordAsync(email, password);
17            await SecureStorage.SetAsync("user_email", email);
18            await SecureStorage.SetAsync("user_uid", _authLink.User
.LocalId);
19            string token = _authLink.FirebaseToken;
20            return token;
21        }
22        catch (Exception ex)
23        {
24            return $"Error: {ex.Message}";
25        }
26    }
27 }

```

Listing 9. Klasa FirebaseAuthService

4.3.1. Wyjaśnienie kodu

- Deklaracja klasy FirebaseAuthService na Listing 9 (s. 23):

```

1     public class FirebaseAuthService
2

```

Listing 10. Deklaracja klasy

Klasa **FirebaseAuthService** odpowiada za autentykację użytkownika w Firebase przy użyciu emaila i hasła.

- **Konstruktor klasy FirebaseAuthService()** na **Listing 11** (s. 24):

```
1 public FirebaseAuthService()  
2 {  
3     _authProvider = new FirebaseAuthProvider(new  
FirebaseConfig(_firebaseApiKey));  
4 }  
5
```

Listing 11. Konstruktor klasy

Konstruktor tworzy obiekt **FirebaseAuthProvider**, który będzie używany do wykonywania operacji autoryzacji. Przekazuje mu klucz API Firebase, który pozwala na dostęp do usług Firebase.

- **Metoda SignInWithEmailAndPasswordAsync** na **Listing 12** (s. 24):

```
1 public async Task<string> SignInWithEmailAndPasswordAsync(  
string email, string password)  
2
```

Listing 12. Metoda SignInWithEmailAndPasswordAsync

Metoda ta umożliwia użytkownikowi logowanie się przy użyciu adresu e-mail i hasła. Zwraca token autoryzacyjny, jeśli logowanie zakończy się powodzeniem.

- **Proces logowania** na **Listing 13** (s. 24):

```
1 try  
2 {  
3     _authLink = await _authProvider.  
SignInWithEmailAndPasswordAsync(email, password);  
4     await SecureStorage.SetAsync("user_email", email);  
5     await SecureStorage.SetAsync("user_uid", _authLink.User  
.LocalId);  
6     string token = _authLink.FirebaseToken;  
7     return token;  
8 }  
9 catch (Exception ex)  
10 {  
11     return $"Error: {ex.Message}";  
12 }  
13
```

Listing 13. Proces logowania

W tej części kodu:

- **SignInWithEmailAndPasswordAsync** jest wywoływana na obiekcie **_authProvider**, który próbuje zalogować użytkownika.
- Jeśli logowanie jest pomyślne, przechowujemy dane użytkownika (adres e-mail oraz unikalny identyfikator **user_uid**) w **SecureStorage**.
- Na końcu zwracamy token autoryzacyjny **_authLink.FirebaseToken**.

• Obsługa błędów na Listing 14 (s. 25):

```

1      catch (Exception ex)
2      {
3          return $"Error: {ex.Message}";
4      }
5  
```

Listing 14. Obsługa błędów

W przypadku wystąpienia błędu, zostanie on przechwycony przez blok **catch**, a metoda zwróci komunikat o błędzie.

4.4. Klasa SignUpPage

Klasa **SignUpPage** 15 (s. 25) jest odpowiedzialna za interfejs użytkownika, który umożliwia nowym użytkownikom rejestrację w aplikacji. Zawiera formularz rejestracyjny, w którym użytkownicy mogą wprowadzić swoje dane, takie jak adres e-mail, hasło oraz inne wymagane informacje. Strona ta obsługuje proces rejestracji, weryfikując dane wejściowe użytkownika oraz wysyłając je do systemu autoryzacji, np. **Firebase**, w celu utworzenia nowego konta. Dzięki tej klasie użytkownicy mogą założyć nowe konto w aplikacji, co umożliwia im późniejsze logowanie i korzystanie z jej funkcji. Klasa **SignUpPage** jest kluczowym elementem procesu rejestracji i zapewnia łatwy dostęp do funkcji tworzenia konta.

```

1  using Microsoft.Maui.Controls;
2
3  namespace CYD.Pages;
4
5  public partial class SignUpPage : ContentPage
6  {
7      private FirebaseAuthService _authService;
8      public SignUpPage()
9      {
10         InitializeComponent();
11         _authService = new FirebaseAuthService();
12     }

```

```
13
14 private async void OnSignUpClicked(object sender, EventArgs e)
15 {
16     string firstName = FirstNameEntry.Text;
17     string lastName = LastNameEntry.Text;
18     string email = EmailEntry.Text;
19     string password = PasswordEntry.Text;
20     string confirmPassword = ConfirmPasswordEntry.Text;
21
22     if (string.IsNullOrEmpty(firstName) || string.IsNullOrEmpty(
23 lastName) ||
24         string.IsNullOrEmpty(email) || string.IsNullOrEmpty(
25 password) ||
26         string.IsNullOrEmpty(confirmPassword))
27     {
28         await DisplayAlert("Error", "Please fill in all fields
29 .", "OK");
30         return;
31     }
32
33     if (password != confirmPassword)
34     {
35         await DisplayAlert("Error", "Passwords do not match.",
36 "OK");
37         return;
38     }
39
40     if (password.Length < 6)
41     {
42         await DisplayAlert("Error", "Password must be at least 6
43 signs!", "OK");
44         return;
45     }
46
47     string result = await _authService.\textbf{
48 RegisterWithEmailAndPasswordAsync}(email, password);
49
50     if (!string.IsNullOrEmpty(result) && result.StartsWith("
51 Error"))
52     {
53         await DisplayAlert("Error", "Unable to register account
54 ", "OK");
55         return;
56     }
57     else
```

```

50     {
51         await DisplayAlert("Registration Success", "Account
created successfully!", "OK");
52         await Navigation.PopAsync();
53
54         EmailEntry.Text = string.Empty;
55         PasswordEntry.Text = string.Empty;
56         ConfirmPasswordEntry.Text = string.Empty;
57     }
58
59 }
60
61 private async void OnLoginTapped(object sender, EventArgs e)
62 {
63     await Navigation.PopAsync();
64 }
65 }

```

Listing 15. Klasa SignUpPage

4.4.1. Wyjaśnienie kodu

- Deklaracja klasy **SignUpPage** na Listing 15 (s. 25):

```

1     public partial class SignUpPage : ContentPage
2

```

Listing 16. Deklaracja klasy

Klasa **SignUpPage** odpowiada za formularz rejestracji użytkownika w aplikacji. Dziedziczy po klasie **ContentPage**, co oznacza, że jest stroną w aplikacji opartej na technologii Xamarin (MAUI).

- Konstruktor klasy **SignUpPage()** na Listing 17 (s. 27):

```

1     public SignUpPage()
2     {
3         InitializeComponent();
4         _authService = new FirebaseAuthService();
5     }
6

```

Listing 17. Konstruktor klasy

Konstruktor inicjalizuje komponenty strony przy pomocy **InitializeComponent()** oraz tworzy obiekt **FirebaseAuthService**, który będzie odpowiedzialny za logikę rejestracji użytkownika za pomocą Firebase.

- **Metoda OnSignUpClicked na Listing 18 (s. 28):**

```
1 private async void OnSignUpClicked(object sender, EventArgs  
2 e)
```

Listing 18. Metoda OnSignUpClicked

Metoda ta jest wywoływana, gdy użytkownik kliknie przycisk rejestracji. Wykonuje ona szereg sprawdzeń:

- Sprawdza, czy wszystkie pola zostały wypełnione.
- Sprawdza, czy hasła się zgadzają.
- Sprawdza, czy hasło ma odpowiednią długość (przynajmniej 6 znaków).

- **Sprawdzanie i rejestracja użytkownika na Listing 19 (s. 28):**

```
1 string result = await _authService.\textbf{  
RegisterWithEmailAndPasswordAsync}(email, password);  
2  
3 if (!string.IsNullOrEmpty(result) && result.StartsWith("Error"))  
4 {  
5     await DisplayAlert("Error", "Unable to register account", "OK");  
6     return;  
7 }  
8 else  
9 {  
10     await DisplayAlert("Registration Success", "Account created successfully!", "OK");  
11     await Navigation.PopAsync();  
12  
13     EmailEntry.Text = string.Empty;  
14     PasswordEntry.Text = string.Empty;  
15     ConfirmPasswordEntry.Text = string.Empty;  
16 }  
17
```

Listing 19. Sprawdzanie i rejestracja użytkownika

Jeśli wszystkie dane są poprawne, wywoływana jest metoda **RegisterWithEmailAndPasswordAsync** z klasy **FirebaseAuthService**, która próbuje zarejestrować użytkownika. Jeśli rejestracja zakończy się niepowodzeniem, wyświetli się komunikat o błędzie. W przeciwnym razie użytkownik zostaje poinformowany o sukcesie i strona rejestracji zostaje zamknięta.

- Metoda **OnLoginTapped** na Listing 20 (s. 29):

```

1     private async void OnLoginTapped(object sender, EventArgs e
2     )
3     {
4         await Navigation.PopAsync();
5     }

```

Listing 20. Metoda OnLoginTapped

Metoda **OnLoginTapped** umożliwia użytkownikowi powrót do strony logowania, klikając w odpowiedni element interfejsu.

4.5. Klasa CoinGeckoService

Klasa CoinGeckoService 21 (s. 29) jest odpowiedzialna za integrację aplikacji z API CoinGecko, które dostarcza dane o kryptowalutach, takie jak aktualne kursy, historia cen czy informacje o tokenach. Dzięki tej klasie aplikacja może pobierać informacje na temat najnowszych cen kryptowalut, przeliczać wartości walut w czasie rzeczywistym oraz analizować dane historyczne. Klasa ta korzysta z metod API CoinGecko, umożliwiając aplikacji dostęp do szerokiej gamy danych o rynku kryptowalut, a także zapewnia efektywną obsługę zapytań i zarządzanie odpowiedziami z API. Dzięki CoinGeckoService użytkownicy aplikacji mogą śledzić zmiany w cenach kryptowalut oraz podejmować bardziej świadome decyzje inwestycyjne na podstawie bieżących danych rynkowych.

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Net.Http.Json;
7
8 namespace CYD.Services
9 {
10     public class CoinGeckoService
11     {
12         private readonly HttpClient _httpClient;
13
14         public CoinGeckoService()
15         {
16             _httpClient = new HttpClient
17             {

```

```

18         BaseAddress = new Uri("https://api.coingecko.com/
api/v3/")
19     };
20 }
21
22     public async Task<List<CryptoCurrency>>
GetTopCryptocurrenciesAsync(int count = 10)
23     {
24         try
25         {
26             var result = await _httpClient.GetFromJsonAsync<
List<CryptoCurrency>>($"coins/markets?vs_currency=usd&order=
market_cap_desc&per_page={count}&page=1&sparkline=false");
27             return result ?? new List<CryptoCurrency>();
28         }
29         catch (Exception ex)
30         {
31             Console.WriteLine($"Error fetching data from
CoinGecko: {ex.Message}");
32             return new List<CryptoCurrency>();
33         }
34     }
35 }
36
37 public class CryptoCurrency
38 {
39     public string Id { get; set; }
40     public string Symbol { get; set; }
41     public string Name { get; set; }
42     public decimal Current_price { get; set; }
43     public string Image { get; set; }
44 }
45 }

```

Listing 21. Klasa CoinGeckoService

4.5.1. Wyjaśnienie kodu

- Deklaracja klasy CoinGeckoService na Listing 21 (s. 29):

```

1     public class CoinGeckoService
2

```

Listing 22. Deklaracja klasy CoinGeckoService

Klasa **CoinGeckoService** odpowiada za komunikację z API CoinGecko w celu pobierania danych o kryptowalutach.

- Konstruktor klasy **CoinGeckoService()** na **Listing 23** (s. 31):

```

1      public CoinGeckoService()
2      {
3          _httpClient = new HttpClient
4          {
5              BaseAddress = new Uri("https://api.coingecko.com/
api/v3/")
6          };
7      }
8  
```

Listing 23. Konstruktor klasy CoinGeckoService

Konstruktor inicjalizuje obiekt **HttpClient** z ustawioną bazową ścieżką **https://api.coingecko.com** co umożliwia wykonywanie zapytań do API CoinGecko.

- Metoda **GetTopCryptocurrenciesAsync** na **Listing 24** (s. 31):

```

1      public async Task<List<Cryptocurrency>>
GetTopCryptocurrenciesAsync(int count = 10)
2  
```

Listing 24. Metoda GetTopCryptocurrenciesAsync

Metoda ta pobiera dane o najpopularniejszych kryptowalutach z API CoinGecko. Przyjmuje parametr **count**, który określa liczbę kryptowalut do pobrania (domyślnie 10).

- Pobieranie danych z API na **Listing 25** (s. 31):

```

1      var result = await _httpClient.GetFromJsonAsync<List<
Cryptocurrency>>($"coins/markets?vs_currency=usd&order=
market_cap_desc&per_page={count}&page=1&sparkline=false");
2      return result ?? new List<Cryptocurrency>();
3  
```

Listing 25. Pobieranie danych z API

Dane są pobierane za pomocą metody **GetFromJsonAsync** z biblioteki **System.Net.Http.Json**. Jeśli żądanie zwróci **null**, metoda zwróci pustą listę.

- Obsługa błędów na **Listing 26** (s. 32):

```
1      catch (Exception ex)
2      {
3          Console.WriteLine($"Error fetching data from CoinGecko:
4          {ex.Message}");
5          return new List<Cryptocurrency>();
6      }
```

Listing 26. Obsługa błędów

W przypadku wystąpienia błędu, metoda wyświetla komunikat w konsoli i zwraca pustą listę kryptowalut.

- **Klasa Cryptocurrency** na **Listing 27** (s. 32):

```
1      public class Cryptocurrency
2      {
3          public string Id { get; set; }
4          public string Symbol { get; set; }
5          public string Name { get; set; }
6          public decimal Current_price { get; set; }
7          public string Image { get; set; }
8      }
9
```

Listing 27. Klasa Cryptocurrency

Klasa **Cryptocurrency** definiuje strukturę danych dla kryptowalut, zawierając takie informacje jak:

- **Id** - unikalny identyfikator kryptowaluty.
- **Symbol** - skrócony symbol kryptowaluty.
- **Name** - pełna nazwa kryptowaluty.
- **Current_price** - aktualna cena kryptowaluty w USD.
- **Image** - URL do obrazu kryptowaluty.

5. Testowanie

Testowanie aplikacji mobilnej jest kluczowym etapem procesu jej rozwoju, który pozwala na weryfikację jakości, funkcjonalności oraz wydajności oprogramowania. W dynamicznie zmieniającym się środowisku mobilnym, gdzie różnorodność urządzeń, systemów operacyjnych i warunków użytkowania jest ogromna, testy odgrywają kluczową rolę w zapewnieniu satysfakcji użytkowników oraz stabilności aplikacji. Niniejszy rozdział omawia metody, narzędzia oraz najlepsze praktyki w zakresie testowania aplikacji mobilnych.

5.1. Logowanie

Funkcja	Typ danych	Nie działa/Działa
Logowanie	Poprawne dane	Działa
	poprawny email, błędne hasło	Nie działa
	błędny e-mail, poprawne hasło	Nie działa
	błędne dane	Nie działa
Logowanie biometryczne	Poprawny palec	Działa
	Niepoprawny palec	Nie działa

Tab. 5.1. Logowanie

W tabelce 5.1 (s. 33) widzimy:

- Logowanie przebiega poprawnie, jeśli dane wprowadzone przez użytkownika są poprawne. W przypadku podania niepoprawnego adresu e-mail lub hasła logowanie zostanie odrzucone.
- Logowanie biometryczne działa poprawnie, jeśli użytkownik wprowadzi poprawny palec. W przypadku podania nieprawidłowego palca logowanie zostanie odrzucone.

5.2. Rejestracja

Funkcja	Typ danych	Nie działa/Działa
Rejestracja	Poprawne dane	Działa
	poprawny email,błędne hasło	Nie działa
	błędny e-mail,poprawne hasło	Nie działa
	Hasło tylko małe litery	Działa
	Hasło tylko duże litery	Działa
	Hasło tylko cyfry	Działa
	Zajety e-mail	Nie działa

Tab. 5.2. Rejestracja

W tabelce 5.2 (s. 34) widzimy:

- Rejestracja przebiega poprawnie, jeśli dane wprowadzone przez użytkownika są poprawne. W przypadku podania niepoprawnego adresu e-mail lub hasła rejestracja zostanie przerwana.
- Hasło może składać się tylko z małych liter, tylko z dużych liter, tylko z liczb lub z kombinacji małych, dużych liter i cyfr.
- Adres e-mail nie może być już zajęty. W przypadku podania adresu e-mail, który jest już używany przez innego użytkownika, rejestracja zostanie przerwana.

5.3. Resetowanie hasła

Funkcja	Typ danych	Nie działa/Działa
Resetowanie hasła	Wysyłanie linku z resetem	Nie działa
	Wysyłanie na zły e-mail	Nie działa
	Wysyłanie na dobry e-mail	Nie działa

Tab. 5.3. Resetowanie hasła

W tabelce 5.7 (s. 36) widzimy:

- Żadna funkcja nie działa poprawnie.

5.4. Menu boczne

Funkcja	Typ danych	Nie działa/Działa
Menu boczne	Przycisk	Działa
	Przesunięcie w bok	Działa
	Przyciski w menu	Działa

Tab. 5.4. Menu boczne

W tabelce 5.4 (s. 35) widzimy:

- Menu boczne działa poprawnie, jeśli użytkownik kliknie lub przesunie w bok.

5.5. Kryptowaluty

Funkcja	Typ danych	Nie działa/Działa
Kurs kryptowalut	Aktualizacja cen	Działa

Tab. 5.5. Kryptowaluty

W tabelce 5.7 (s. 36) widzimy:

- Funkcja "Aktualizacja cen" działa poprawnie.

5.6. Kurs walut

Funkcja	Typ danych	Nie działa/Działa
Kurs walut	Pobieranie wartości walut	Nie działa

Tab. 5.6. Kurs walut

W tabelce 5.7 (s. 36) widzimy:

- Funkcja "Pobieranie wartości walut" nie działa poprawnie.

5.7. Sensor światła

Funkcja	Typ danych	Nie działa/Działa
Sensor światła	Zmiana motywu zależności od naświetlenia	Nie działa
Przycisk do zmiany motywu	Zmiana motywu	Działa

Tab. 5.7. Sensor światła

W tabelce 5.7 (s. 36) widzimy:

- Funkcja "Zmiana motywu zależności od naświetlenia" nie działa poprawnie.
- Funkcja "Zmiana motywu" działa poprawnie.

5.8. Dane logowania

Funkcja	Typ danych	Nie działa/Działa
Dane logowania	Zmiana e-mail na błędny	Nie działa
	Zmiana e-mail na poprawny	Działa
	Zmiana hasła	Działa
	Zmiana hasła błąd	Działa
	Zmiana hasła znak specjalny	Działa
	Zmiana hasła jedna litera	Działa
	Zmiana hasła tylko duże litery	Działa
	Zmiana hasła tylko małe litery	Działa

Tab. 5.8. Dane logowania

W tabelce 5.9 (s. 38) widzimy:

- Funkcja "Zmiana e-mail na błędny" nie działa.
- Funkcja "Zmiana e-mail na poprawny" działa poprawnie.
- Funkcja "Zmiana hasła" działa poprawnie, niezależnie od typu danych.

5.9. Wydatki

Funkcja	Typ danych	Nie działa/Działa
Wydatki	Dodanie wydatków	Działa
	Dodanie kategori	Działa
	Dodanie kwoty	Działa
	Odświeżanie	Działa

Tab. 5.9. Wydatki

W tabelce 5.9 (s. 38) widzimy:

- Funkcja "Dodanie wydatków" działa poprawnie.
- Funkcja "Dodanie kategori" działa poprawnie.
- Funkcja "Dodanie kwoty" działa poprawnie.
- Funkcja "Odświeżanie" działa poprawnie.

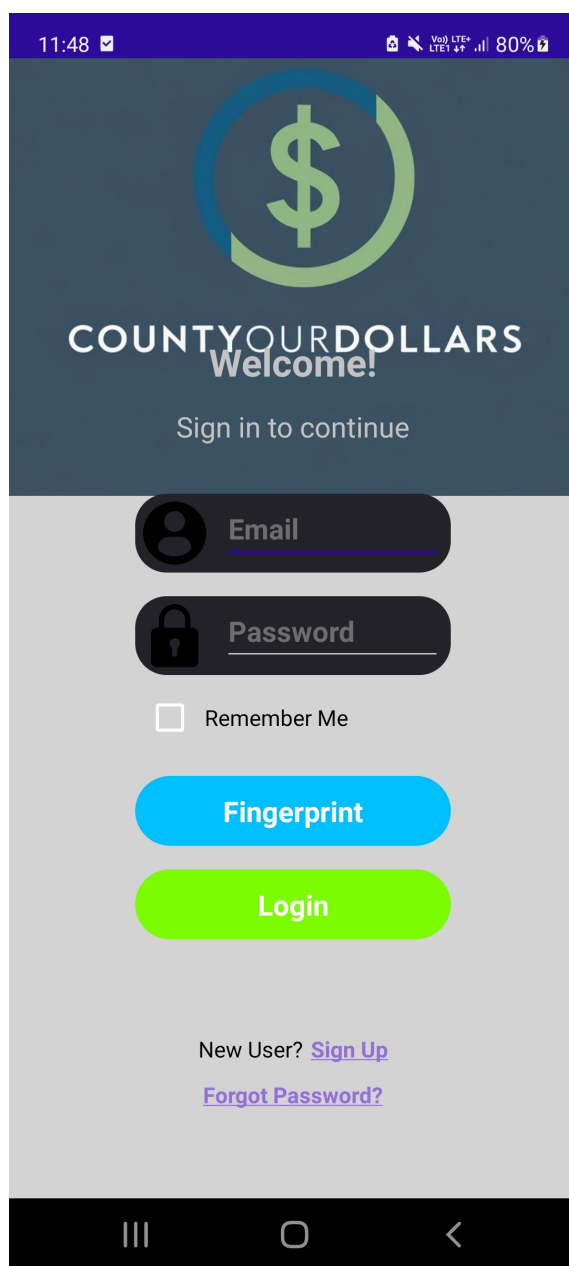
6. Podręcznik użytkownika

Podręcznik użytkownika to rozdział, który ma na celu pomóc użytkownikowi w pełnym zrozumieniu funkcji i możliwości aplikacji. Zawiera szczegółowe instrukcje dotyczące obsługi, nawigacji oraz rozwiązywania potencjalnych problemów, z jakimi użytkownik może się spotkać. Jego celem jest zapewnienie łatwego i efektywnego korzystania z aplikacji poprzez wyjaśnienie, jak wykonywać różne czynności, jak logowanie, rejestracja, dodawanie wydatków czy zarządzanie kryptowalutami.

6.1. Logowanie i rejestracja

Po uruchomieniu aplikacji użytkownik zostaje przeniesiony na stronę logowania 40 6.1., na której znajdują się następujące elementy:

- **Pole do wpisania adresu e-mail:** Użytkownik wpisuje swój adres e-mail.
- **Pole do wpisania hasła:** Użytkownik wpisuje swoje hasło.
- **Przycisk „Login”:** Umożliwia zalogowanie się po wpisaniu poprawnych danych.
- **Przycisk „Fingerprint”:** Umożliwia zalogowanie się za pomocą odcisku palca (jeśli urządzenie obsługuje tę funkcjonalność).
- **Link „Sign Up”:** Kliknięcie w ten podkreślony tekst przenosi użytkownika do strony rejestracji.
- **Link „Forgot Password?”:** Kliknięcie w ten podkreślony tekst przenosi użytkownika na stronę odzyskiwania hasła.



Rys. 6.1. Strona Logowania (LoginPage)

Na stronie rejestracji 6.2. użytkownik może wprowadzić swoje dane:

- **First Name:** Imię użytkownika.
- **Last Name:** Nazwisko użytkownika.
- **Email Address:** Adres e-mail użytkownika.
- **Password:** Hasło użytkownika.
- **Confirm Password:** Potwierdzenie hasła.

Jeśli którekolwiek z tych pól pozostanie puste, po kliknięciu przycisku „Sign Up” użytkownik otrzyma komunikat o błędzie, informujący o konieczności uzupełnienia wszystkich pól.

Na dole strony znajduje się również link „Login”, który przenosi użytkownika z powrotem do strony logowania.

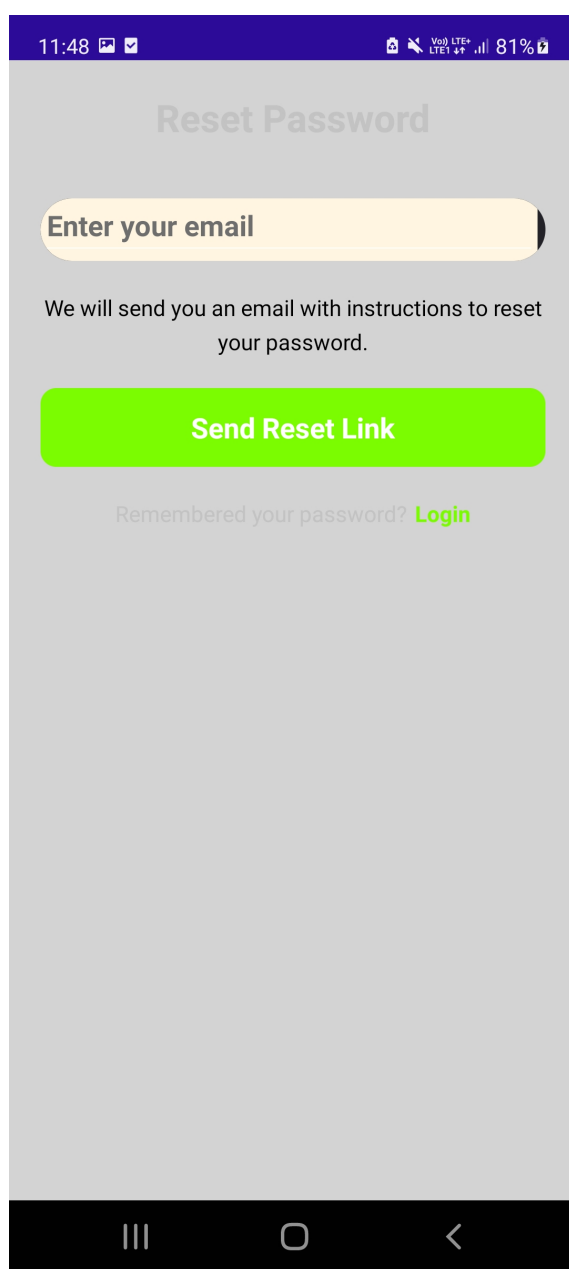
Po pomyślnym wypełnieniu formularza i kliknięciu przycisku „Sign Up”, użytkownik zostaje zarejestrowany w systemie za pomocą Firebase Authentication i przekierowany na stronę logowania.

Rys. 6.2. Strona Rejestracji (SignUpPage)

Na stronie odzyskiwania hasła⁴² 6.3. użytkownik może wprowadzić swój adres

e-mail, aby otrzymać link do zresetowania hasła.

- **Email:** Pole, w którym użytkownik wprowadza swój adres e-mail.
- **Send Reset Link:** Przycisk wysyłający link do resetowania hasła na podany adres e-mail.
- **Link „Remember your password?Login”:** Kliknięcie w ten zielony tekst przenosi użytkownika z powrotem na stronę logowania.

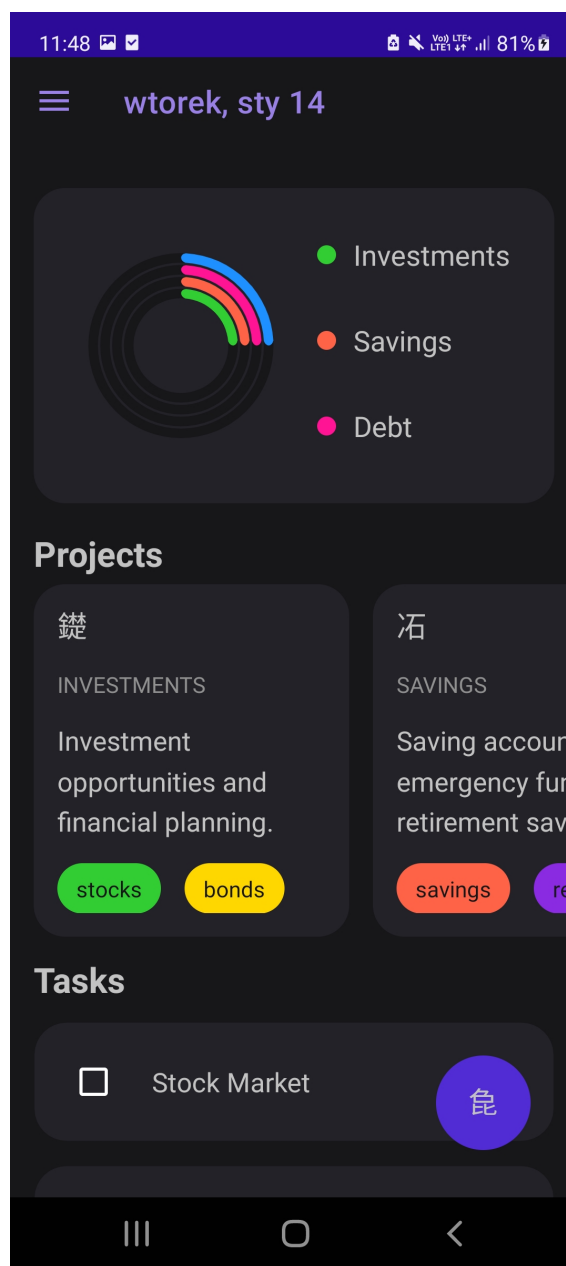


Rys. 6.3. Strona Odzyskiwania Hasła (ForgotPasswordPage)

6.2. Korzystanie z aplikacji

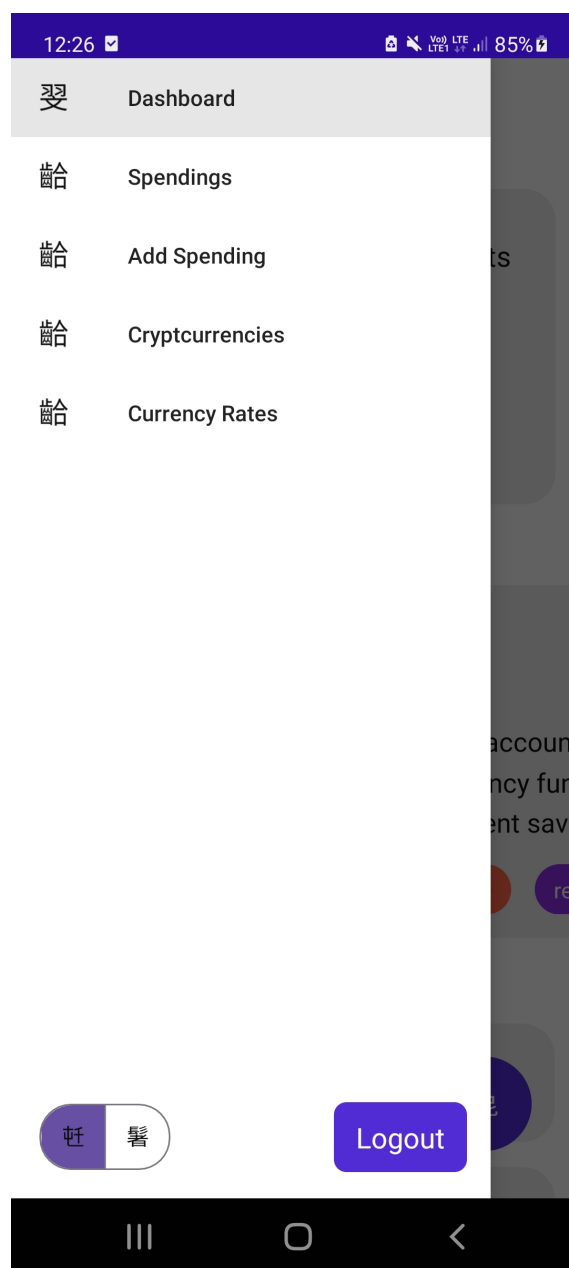
Po zalogowaniu się użytkownik zostaje przeniesiony na stronę główną 43 6.4., gdzie wyświetlane są dane dotyczące jego finansów. W tej sekcji znajdują się kategorie, takie jak:

- **Investments:** Informacje o inwestycjach .
- **Saving:** Jak można zaoszczędzić.
- **Debt:** Informacje o zadłużeniach.



Rys. 6.4. Strona Główna (MainPage)

Na głównym ekranie aplikacji (AppShell)⁴⁴ 6.5. dostępne są przejścia do różnych stron, takich jak Dashboard, Spendings, Add Spending, Cryptocurrencies oraz Currency Rates. Na dole ekranu znajduje się przełącznik umożliwiający zmianę motywu z jasnego na ciemny oraz przycisk „Logout”, który wylogowuje użytkownika z aplikacji.

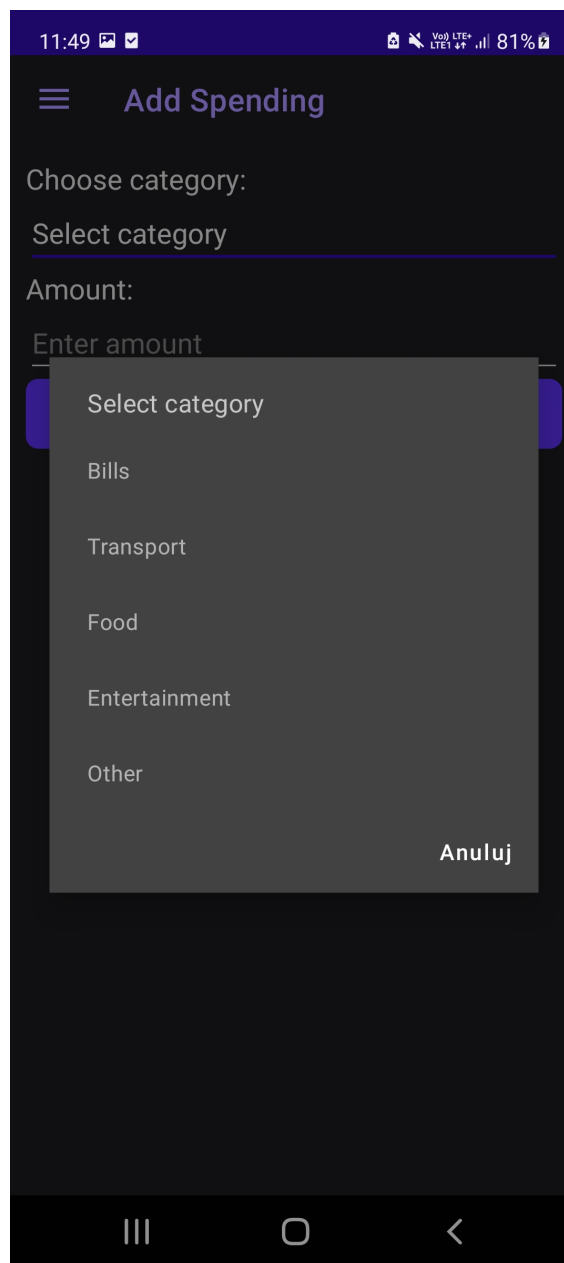


Rys. 6.5. AppShell

Na stronie dodawania wydatków ⁴⁵ 6.6. przechodzimy na nią poprzez kliknięcie w appshell Add Spending użytkownik może dodać nowe wydatki, wybierając jedną z dostępnych kategorii:

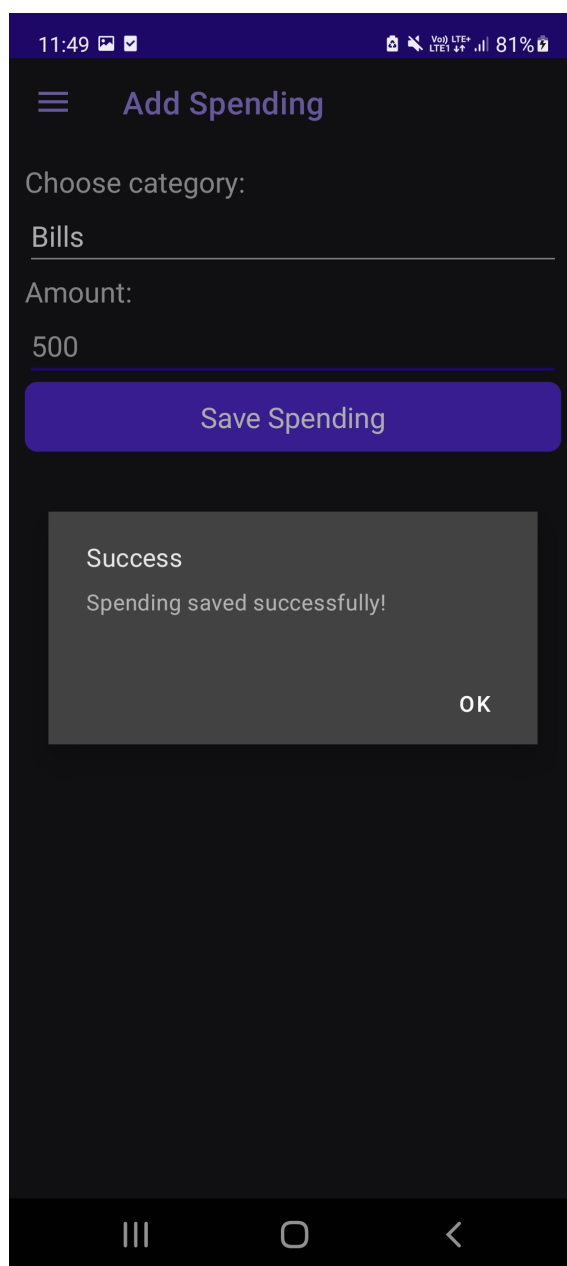
- **Bills:** Kategorie wydatków związane z rachunkami.

- **Transport:** Wydatki na transport.
- **Food:** Wydatki na jedzenie.
- **Entertainment:** Wydatki na rozrywkę.
- **Other:** Inne wydatki.



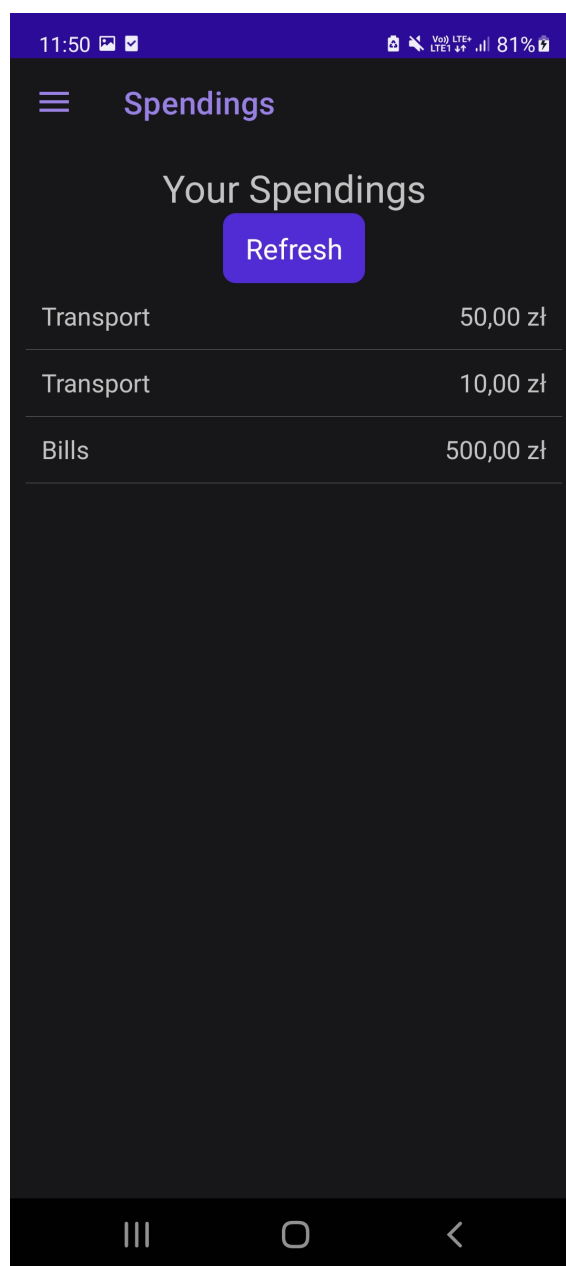
Rys. 6.6. Strona Dodawania Wydatków (AddSpendingPage)

Na tej samej stronie użytkownik wprowadza wartość wydatku (Amount)46 6.7.. Po kliknięciu przycisku „Save Spending” użytkownik otrzymuje komunikat informujący o pomyślnym zapisaniu wydatku.



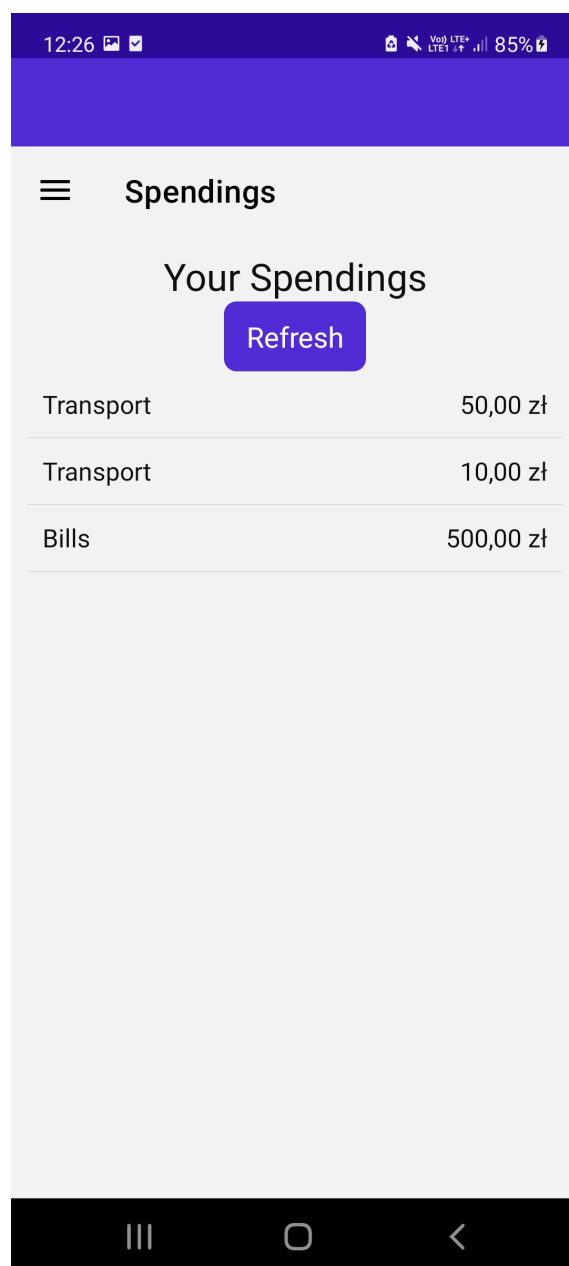
Rys. 6.7. Wpisywanie Wartości Wydatku (AddSpendingPage)

Po dodaniu wydatków w sekcji Add Spending, użytkownik może przejść do strony Spendings⁴⁷ 6.8., gdzie wyświetlana jest szczegółowa tabela z dotychczasowymi wydatkami. Tabela zawiera takie informacje, jak kategoria wydatku, jego kwota. Dzięki temu użytkownicy mogą łatwo śledzić swoje wydatki w danym okresie, co pomaga w utrzymaniu kontroli nad finansami. W przypadku gdy dane są aktualizowane lub użytkownik chce sprawdzić najnowsze wydatki, dostępny jest przycisk „Refresh”. Po jego kliknięciu strona odświeża się, prezentując najnowsze informacje, co pozwala na bieżąco monitorować zmiany w wydatkach.



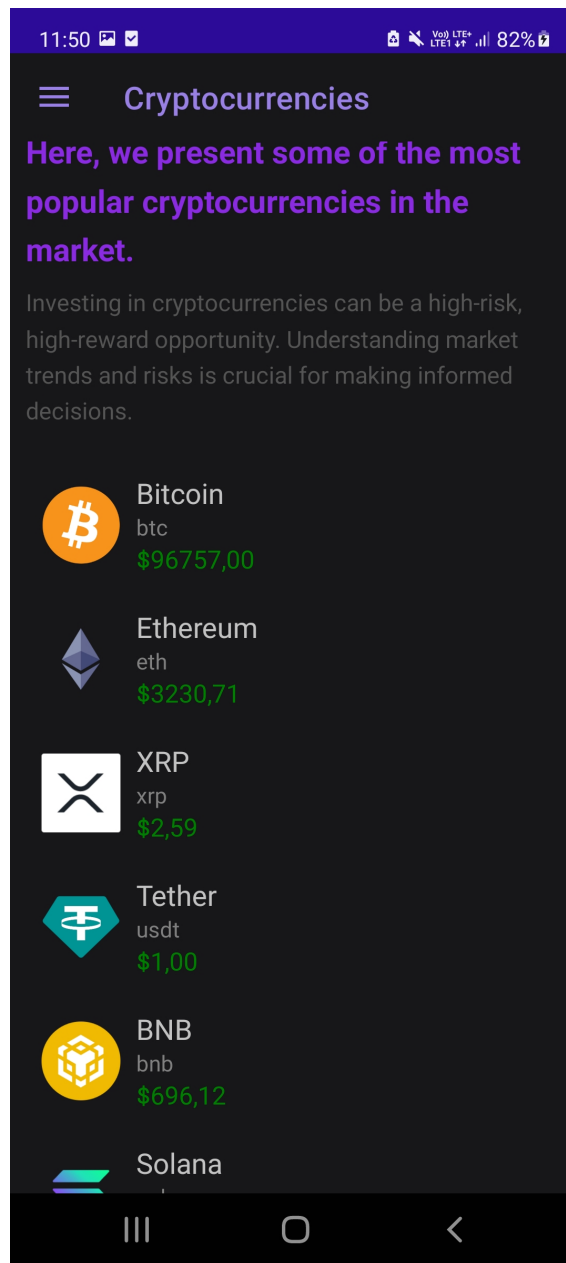
Rys. 6.8. Strona Wydatków (SpendingPage)

Strona Wydatków wyświetlana w trybie jasnym⁴⁸ 6.9 jest zaprojektowana w sposób przejrzysty i intuicyjny. Użytkownicy mogą łatwo zorientować się w swojej historii wydatków dzięki jasnemu tłu oraz czytelnym kolumnom w tabeli. Jest to przyjazny interfejs, w którym można szybko poruszać się po poszczególnych sekcjach, dodając nowe wydatki, edytując istniejące lub usuwając te, które już nie są potrzebne. Taki widok sprzyja efektywnemu zarządzaniu osobistymi finansami, zapewniając użytkownikowi wygodę i komfort w codziennym użytkowaniu aplikacji.



Rys. 6.9. Strona Wydatków w Trybie Jasnym (SpendingPage)

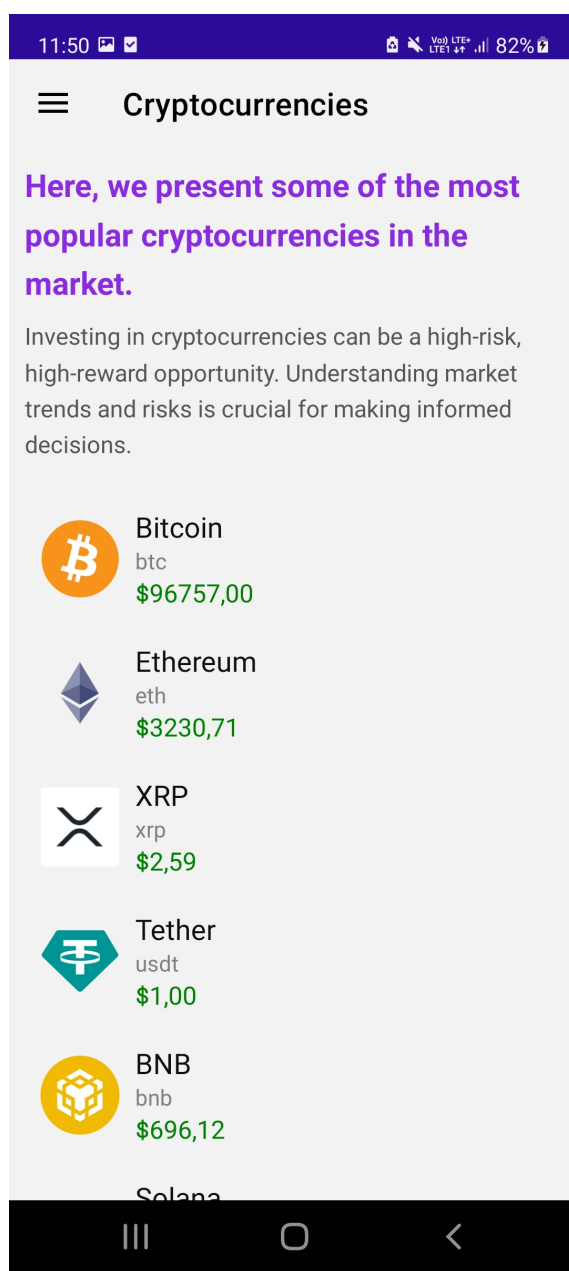
Na stronie Cruptocurrencies 49 6.10. użytkownicy mają możliwość zapoznania się z podstawowymi informacjami na temat kryptowalut, które stanowią jedną z najnowszych i najbardziej innowacyjnych form cyfrowych środków płatniczych. Można tu znaleźć szczegóły dotyczące ich funkcjonowania, historii, oraz rosnącej roli w globalnej gospodarce. Ponadto, strona prezentuje aktualne ceny najbardziej popularnych kryptowalut, takich jak Bitcoin, Ethereum, czy Binance Coin, umożliwiając użytkownikom śledzenie ich wartości w czasie rzeczywistym. Dzięki temu użytkownicy mogą podejmować bardziej świadome decyzje inwestycyjne, bazując na bieżących informacjach rynkowych.



Rys. 6.10. Strona Kryptowalut (CryptocurrenciesPage)

Strona Kryptowalut wyświetlana jest w standardowym trybie ciemnym, co umożliwia łatwe i wygodne przeglądanie treści. W tym widoku użytkownicy mogą zobaczyć listę kryptowalut wraz z ich bieżącymi cenami oraz dodatkowymi informacjami, takimi jak zmiany wartości w skali ostatnich godzin, dni czy tygodni. Interfejs jest zaprojektowany w sposób przejrzysty, z intuicyjnym układem, który sprawia, że korzystanie z platformy jest wygodne, nawet dla osób, które dopiero zaczynają swoją przygodę z kryptowalutami. Strona Kryptowalut może być również wyświetlana w trybie jasnym 50 6.11, który zmienia kolorystykę strony na bardziej stonowaną, z dominującymi jasnymi tonami, co jest szczególnie korzystne w warunkach wysokiego

oświecenia.



Rys. 6.11. Strona Kryptowalut (CryptocurrenciesPage – Tryb Jasny)

Bibliografia

- [1] *Firebase*. URL: <https://firebase.google.com/>.
- [2] *CODECOOL*. URL: <https://codecool.com/pl/blog/github-czym-jest-i-do-czego-sluzy/>.

Spis rysunków

1.1. Logo	6
3.1. Ekran logowania	12
3.2. Ekran Sign Up	13
6.1. Strona Logowania (LoginPage)	40
6.2. Strona Rejestracji (SignUpPage)	41
6.3. Strona Odzyskiwania Hasła (ForgotPasswordPage)	42
6.4. Strona Główna (MainPage)	43
6.5. AppShell	44
6.6. Strona Dodawania Wydatków (AddSpendingPage)	45
6.7. Wpisywanie Wartości Wydatku (AddSpendingPage)	46
6.8. Strona Wydatków (SpendingPage)	47
6.9. Strona Wydatków w Trybie Jasnym (SpendingPage)	48
6.10. Strona Kryptowalut (CryptocurrenciesPage)	49
6.11. Strona Kryptowalut (CryptocurrenciesPage – Tryb Jasny)	50

Spis tabel

5.1. Logowanie	33
5.2. Rejestracja	34
5.3. Resetowanie hasła	35
5.4. Menu boczne	35
5.5. Kryptowaluty	35
5.6. Kurs walut	36
5.7. Sensor światła	36
5.8. Dane logowania	37
5.9. Wydatki	38

Spis listingów

1.	Klasa AppShell	14
2.	Deklaracja klasy	16
3.	Konstruktor klasy	16
4.	Metoda DisplaySnackBarAsync	17
5.	Metoda DisplayToastAsync	17
6.	Zmiana motywu aplikacji	17
7.	Metoda OnLogoutButtonClicked	17
8.	XAML Strony Logowania	18
9.	Klasa FirebaseAuthService	23
10.	Deklaracja klasy	23
11.	Konstruktor klasy	24
12.	Metoda SignInWithEmailAndPasswordAsync	24
13.	Proces logowania	24
14.	Obsługa błędów	25
15.	Klasa SignUpPage	25
16.	Deklaracja klasy	27
17.	Konstruktor klasy	27
18.	Metoda OnSignUpClicked	28
19.	Sprawdzanie i rejestracja użytkownika	28
20.	Metoda OnLoginTapped	29
21.	Klasa CoinGeckoService	29
22.	Deklaracja klasy CoinGeckoService	30
23.	Konstruktor klasy CoinGeckoService	31
24.	Metoda GetTopCryptocurrenciesAsync	31
25.	Pobieranie danych z API	31
26.	Obsługa błędów	32
27.	Klasa CryptoCurrency	32