Specyfikacja wymagań systemowych aplikacji „Dycha”

Autorzy: Aleksander Kominiak, Bartosz Goździkiewicz, Iga Kaczmarek, Jakub Gorządek, Kacper Domiński

Wersja dokumentu: 1.3
Data aktualizacji: 05.05.2025

Historia wersji:

- **1.0** – 15.04.2025: Utworzenie wstępnej wersji dokumentu (SRS) na podstawie założeń początkowych.
- **1.1** – 27.04.2025: Aktualizacja wymagań po wstępnej analizie (m.in. dodano opis funkcji kalkulatora).
- **1.2** – 03.05.2025: Korekta dokumentu po konsultacjach (m.in. modyfikacja sekcji 3.1).
- **1.3** – 05.05.2025: Aktualizacja dokumentu po studium wykonalności i implementacji.

Spis treści

1. Wstęp

1.1. Cel

Celem niniejszego dokumentu jest przedstawienie kluczowych założeń oraz szczegółowej specyfikacji wymagań systemowych dla aplikacji **„Dycha”**. Dokument jest przeznaczony dla członków zespołu projektowego odpowiedzialnych za implementację i testowanie systemu oraz dla interesariuszy pragnących uzyskać pełny obraz funkcjonalności i ograniczeń aplikacji. Specyfikacja ta służy jako materiał referencyjny podczas bieżącej realizacji projektu oraz ewentualnych przyszłych modyfikacji. Jeśli nie zaznaczono inaczej, wszystkie wyspecyfikowane tu wymagania dotyczą wersji 1.0 oprogramowania.

1.2. Zakres projektu

Zakres projektu obejmuje zaprojektowanie, implementację i wdrożenie stacjonarnej aplikacji inwestycyjnej **„Dycha”**. Celem aplikacji jest wsparcie indywidualnych inwestorów giełdowych (oraz entuzjastów rynku finansowego) w monitorowaniu swojego portfela oraz w analizie potencjalnych zysków/strat. System koncentruje się na dostarczeniu kluczowych funkcjonalności niezbędnych do oceny efektywności inwestycji giełdowych, **eliminuje nadmiarowe złożone opcje** obecne w profesjonalnych platformach, stawiając na prostotę i użyteczność. Zakres prac obejmuje integrację z zewnętrznymi źródłami danych rynkowych, implementację lokalnego interfejsu użytkownika na komputery PC oraz zapewnienie bezpiecznego przechowywania danych użytkownika w chmurze.

1.3. Przegląd produktu

1.3.1. Perspektywa produktu

Aplikacja **„Dycha”** jest samodzielnym systemem desktopowym, działającym jako narzędzie do **wspomagania inwestycji giełdowych** dla użytkowników indywidualnych. Nie jest częścią większego systemu korporacyjnego ani nie wymaga specjalistycznej infrastruktury – do działania potrzebuje jedynie typowego komputera PC z systemem Windows oraz połączenia internetowego. Aplikacja wykorzystuje dane giełdowe dostarczane przez zewnętrzne API i prezentuje je użytkownikowi wraz z analizami opartymi o te dane.

Pod względem **środowiska technicznego**, system został zbudowany jako aplikacja desktopowa w języku C# z wykorzystaniem technologii **Windows Forms**. Dane aplikacji (np. informacje o portfelu inwestycyjnym użytkownika) są przechowywane w chmurze – w **bazie danych Firebase** firmy Google – co zapewnia szybkie i bezpieczne przechowywanie oraz synchronizację danych między urządzeniami. Aplikacja nie korzysta z lokalnej bazy danych; wszystkie dane trwałe są przetrzymywane w usłudze chmurowej.

Z perspektywy biznesowej, **„Dycha”** ma stanowić wsparcie dla inwestorów przy podejmowaniu decyzji. Dostarcza ona narzędzia do monitorowania portfela inwestycyjnego, analizy historycznych notowań oraz obliczania średniej ceny zakupu akcji i potencjalnych zysków lub strat przy różnych scenariuszach sprzedaży. W odróżnieniu od rozbudowanych platform (takich jak np. TradingView czy aplikacje maklerskie), **„Dycha”** skupia się na najważniejszych funkcjach potrzebnych typowemu użytkownikowi, eliminując zbędne elementy mogące utrudniać obsługę początkującym inwestorom.

1.3.2. Funkcje produktu

Tabela poniżej przedstawia główne funkcje aplikacji „Dycha” wraz z ich opisem:

Nazwa funkcji	Opis funkcjonalności
Rejestracja konta	Umożliwia utworzenie nowego konta użytkownika za pomocą adresu e-mail i hasła (uwierzytelnianie przez Firebase). Nowo zarejestrowany użytkownik uzyskuje dostęp do wszystkich funkcji aplikacji.
Logowanie	Umożliwia zalogowanie się zarejestrowanemu użytkownikowi przy użyciu adresu e-mail (loginu) i hasła. Po pomyślnej weryfikacji użytkownik zostaje przeniesiony na główny ekran aplikacji.
Wylogowanie	Zamyka bieżącą sesję użytkownika i powraca do ekranu logowania. Służy zapewnieniu, że nieuprawnione osoby nie uzyskają dostępu do danych, gdy użytkownik zakończy pracę.
Wyszukiwanie instrumentu	Umożliwia znalezienie konkretnej akcji lub innego instrumentu finansowego po nazwie lub tickerze. Aplikacja pobiera aktualne **dane rynkowe** (np. bieżący kurs, zmienność, notowania historyczne) z zewnętrznego API i prezentuje je użytkownikowi.
Wyświetlanie notowań i wykresu	Prezentuje szczegółowe informacje o wybranym instrumencie finansowym, w tym aktualną cenę, zmiany (dzienne, procentowe), oraz interaktywny wykres ceny. Wykres generowany jest poprzez osadzenie strony Google Finance w aplikacji (kontrolka WebBrowser). Dzięki temu użytkownik może obserwować aktualne notowania i historyczne trendy cen bezpośrednio w aplikacji.
Dodanie transakcji (zakup)	Umożliwia dodanie pozycji do **portfela** użytkownika, symulując zakup akcji/ETF. Użytkownik określa instrument (np. wybierając go na ekranie głównym) oraz wolumen (liczbę jednostek) do zakupu. System pobiera aktualną cenę rynkową z API i zapisuje nową transakcję w bazie danych użytkownika (Firebase), uwzględniając cenę zakupu, wolumen i datę. Nie ma mechanizmu rzeczywistej transakcji finansowej – funkcja służy do ewidencji i analizy inwestycji użytkownika.
Przegląd portfela	Umożliwia wgląd w posiadany **portfel inwestycyjny**. Użytkownik widzi listę dodanych pozycji (np. zakupionych akcji) wraz z kluczowymi informacjami: symbolem instrumentu, wolumenem, średnią ceną zakupu, bieżącą wartością rynkową, zyskiem/stratą netto oraz procentową stopą zwrotu dla każdej pozycji. Podsumowanie portfela prezentuje łączną wartość konta oraz sumaryczny zysk/stratę.
Kalkulator inwestycyjny	Umożliwia przeprowadzenie symulacji potencjalnego zwrotu z inwestycji. Użytkownik podaje kwotę, jaką planuje zainwestować w wybrany instrument, oraz horyzont czasowy (liczbę dni inwestycji). Na tej podstawie aplikacja, wykorzystując historyczne dane cenowe instrumentu z API, estymuje przewidywaną stopę zwrotu (w %) oraz potencjalny zysk lub stratę

Nazwa funkcji	Opis funkcjonalności
	(w $) po zadanym okresie. Dodatkowo kalkulator wyświetla oszacowanie poziomu **ryzyka** (np. komunikaty typu “niskie/średnie/wysokie ryzyko”) na podstawie zmienności kursu danego instrumentu.
Podgląd szczegółów pozycji	(Opcjonalna) Umożliwia wyświetlenie szczegółowych informacji o wybranej pozycji z portfela. Po zaznaczeniu pozycji z listy portfela, użytkownik może podejrzeć jej szczegóły lub wygenerować dla niej wykres (funkcjonalność realizowana poprzez przekierowanie do ekranu głównego z załadowanym wykresem danego instrumentu).

1.3.3. Charakterystyka użytkowników

Aplikacja **„Dycha”** jest przeznaczona dla pojedynczej **kategorii użytkownika** – typowego inwestora giełdowego (osoby inwestującej indywidualnie na rynku akcji lub zainteresowanej analizą rynku). System nie wprowadza podziału na role użytkowników – wszyscy zalogowani mają taki sam dostęp do wszystkich funkcji aplikacji.

Charakterystyka docelowego użytkownika:

- Podstawowa wiedza z zakresu obsługi komputera i Internetu (wymagana do instalacji aplikacji na Windows oraz korzystania z niej).
- Ogólna orientacja w terminologii finansowej – przydatna, choć niekonieczna, ponieważ aplikacja prezentuje informacje w sposób zrozumiały nawet dla początkujących inwestorów (np. pokazuje zysk/stratę w wartościach bezwzględnych i procentach).
- Dostęp do komputera PC z systemem Windows oraz stałe połączenie z Internetem (wymóg techniczny, by aplikacja mogła pobierać aktualne dane giełdowe).
- **Liczba użytkowników**: Aplikacja jest projektowana z myślą o obsłudze wielu niezależnych użytkowników (każdy posiada własne konto i odrębny portfel w bazie danych). W bieżącej implementacji brak jest jednak rozbudowanych funkcji administracyjnych – wszyscy użytkownicy mają tę samą rolę i korzystają z aplikacji indywidualnie.

Należy zaznaczyć, że **„Dycha”** może być używana zarówno przez inwestorów indywidualnych rozpoczynających przygodę z giełdą, jak i bardziej doświadczonych entuzjastów czy analityków finansowych szukających prostego narzędzia do szybkich kalkulacji. Niemniej, wersja 1.0 nie rozróżnia typów użytkowników w systemie – wszyscy korzystają z tych samych funkcjonalności na równych prawach.

1.4. Definicje

W dokumencie zastosowano następujące skróty i terminy:

- **API** – (Application Programming Interface) Interfejs programistyczny aplikacji. W kontekście projektu „Dycha” odnosi się do zewnętrznego serwisu udostępniającego dane rynkowe online. Aplikacja korzysta z **Finnhub Stock API** do pobierania aktualnych notowań giełdowych (akcje, fundusze ETF itp.).

- **Firebase** – Platforma firmy Google oferująca usługi backend w chmurze (m.in. bazę danych NoSQL oraz uwierzytelnianie). W projekcie wykorzystana do przechowywania danych aplikacji (konto użytkownika, portfel inwestycyjny itp.) w sposób zdalny i synchronizowany między urządzeniami.
- **Windows Forms** – Biblioteka (framework) GUI dla języka C#/.NET umożliwiająca tworzenie aplikacji okienkowych dla systemu Windows. Aplikacja Dycha została zbudowana jako **Windows Forms App**, co warunkuje jej dostępność wyłącznie na systemach z rodziny Windows.
- **Portfel** – zbiór wszystkich posiadanych przez użytkownika pozycji inwestycyjnych (np. akcji określonych spółek lub jednostek funduszy ETF), wprowadzonych do aplikacji w celu monitorowania wyników.
- **Wolumen** – liczba jednostek danego instrumentu finansowego (np. liczba akcji) w jednej transakcji lub pozycji.
- **Średnia cena zakupu** – średni koszt zakupu pojedynczej jednostki instrumentu w ramach danej pozycji w portfelu (np. jeżeli użytkownik kupił akcje spółki w różnych momentach po różnych cenach, średnia ważona cena tych transakcji).
- **Stopa zwrotu** – zmiana wartości inwestycji wyrażona w procentach, zwykle liczona jako (wartość obecna – wartość początkowa) / wartość początkowa × 100%.
- **Finnhub Stock API** – zewnętrzne API dostarczające dane finansowe w czasie rzeczywistym. Darmowy plan tego API pozwala na wykonanie do 60 zapytań na minutę, co jest wykorzystywane w aplikacji do częstego odświeżania notowań.
- **Google Finance** – serwis internetowy firmy Google prezentujący informacje giełdowe, w tym interaktywne wykresy notowań. Aplikacja wykorzystuje wbudowaną przeglądarkę (WebBrowser) do wyświetlania strony Google Finance z wykresem dla wybranego instrumentu.

2. Odniesienia

1. **Raport Studium Wykonalności – Aplikacja „Dycha”** (wersja z 03.05.2025) – dokument analizujący zasadność realizacji projektu, wymagania i potencjalne rozwiązania. Zawiera m.in. opis funkcjonalny aplikacji, analizę ryzyka oraz porównanie możliwych wariantów implementacji.
2. **Dokumentacja API Finnhub** – oficjalna dokumentacja interfejsu Finnhub Stock API (https://finnhub.io/docs/api), zawierająca informacje o sposobie uwierzytelnienia, strukturze danych oraz limitach zapytań (limit 60 zapytań/min dla kont darmowych).
3. **Dokumentacja platformy Firebase** – materiały dla programistów dotyczące korzystania z Firebase (https://firebase.google.com/docs), w szczególności modułów Firebase Authentication (rejestracja/logowanie użytkowników) oraz Cloud Firestore/Realtime Database (przechowywanie danych aplikacji w chmurze).
4. **Wymagania dot. interfejsu Windows Forms** – Microsoft .NET Documentation: Windows Forms Applications (opis możliwości i ograniczeń tworzenia aplikacji okienkowych na platformę Windows).

5. **Google Finance – serwis giełdowy** (https://www.google.com/finance) – źródło wykresów i danych finansowych wykorzystywanych w aplikacji (poprzez osadzony komponent przeglądarki).

Powyższe dokumenty i źródła stanowiły podstawę do opracowania niniejszej specyfikacji oraz realizacji projektu.

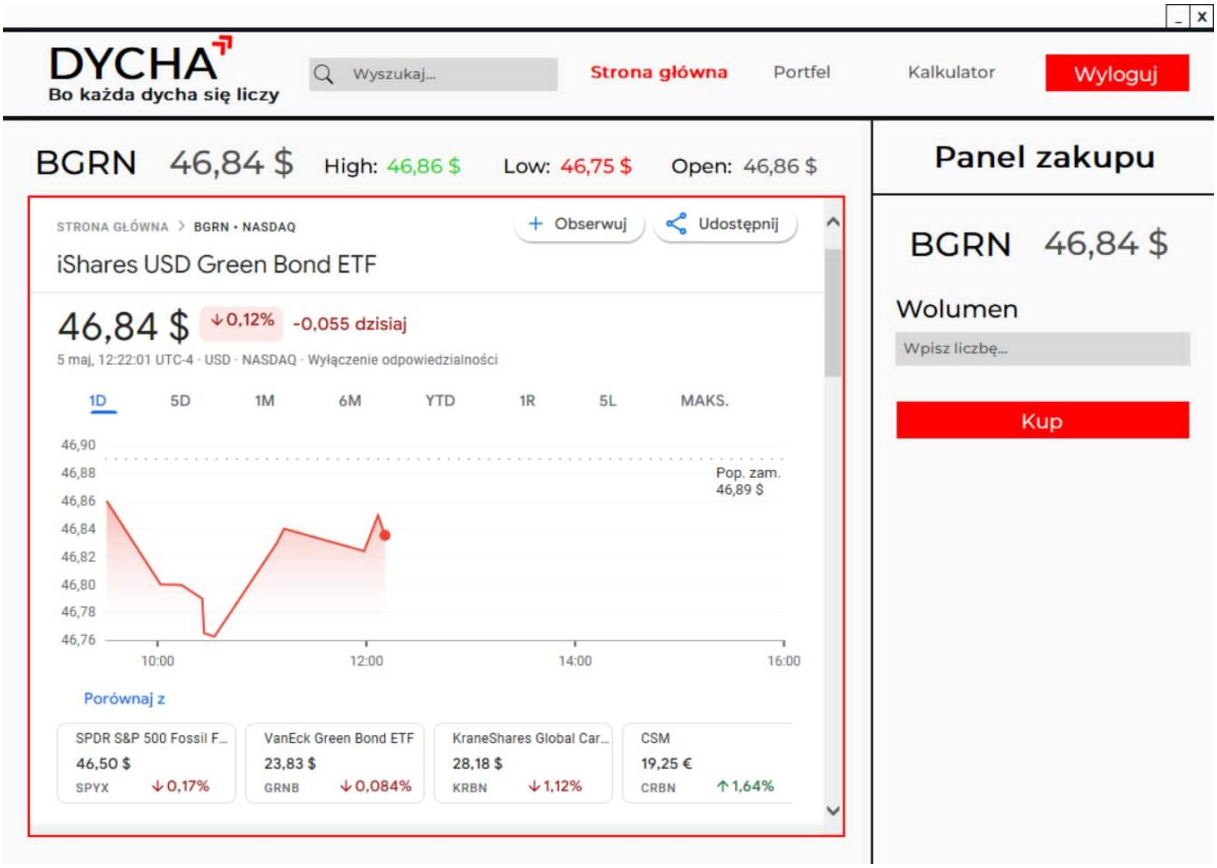
3. Szczegółowe wymagania

3.1. Zewnętrzne interfejsy

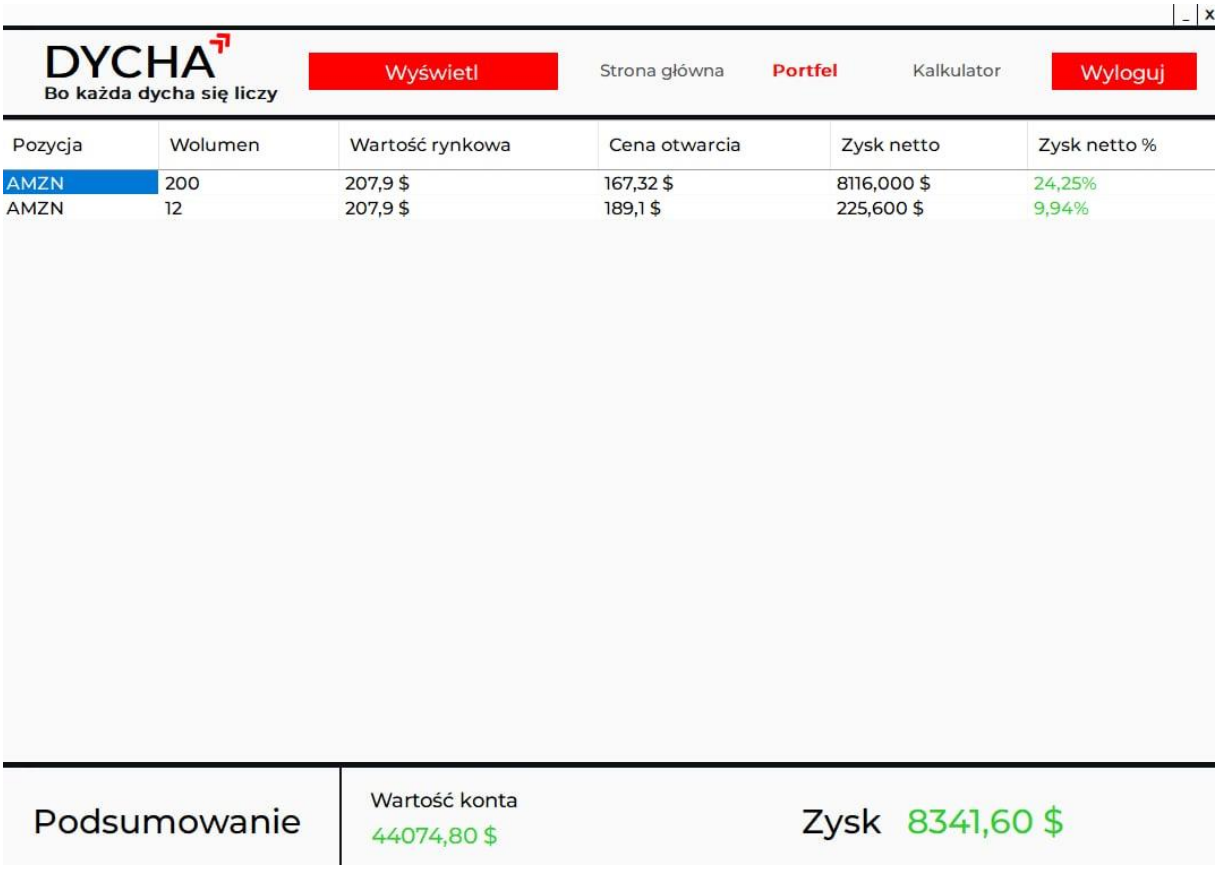
Interfejs użytkownika: Aplikacja **„Dycha”** oferuje graficzny interfejs użytkownika typu desktop (okienkowy). Po uruchomieniu programu użytkownik wita ekran logowania, gdzie może zarejestrować nowe konto lub zalogować się na istniejące. Po pomyślnym zalogowaniu dostępne są trzy główne widoki (sekcje) aplikacji, przełączane za pomocą menu nawigacyjnego w górnej części okna:

- **Strona główna (Dashboard)** – zawiera pasek wyszukiwania instrumentów oraz panel wyświetlania szczegółów wybranego instrumentu. Po wyszukaniu i wybraniu akcji/ETF, użytkownik widzi podstawowe dane rynkowe (bieżący kurs, zmiana dzienna, wartości otwarcia, maksimum/minimum itp.) oraz wykres notowań. Obok wykresu znajduje się **panel zakupu**, który umożliwia wprowadzenie liczby jednostek do kupienia i dodanie tej pozycji do portfela.
- **Portfel** – przedstawia tabelaryczne zestawienie wszystkich pozycji dodanych przez użytkownika. Dla każdej pozycji wyświetlane są informacje takie jak: symbol (ticker), posiadany wolumen, bieżąca wartość rynkowa, cena zakupu (lub średnia cena w przypadku wielu transakcji), aktualny zysk/strata netto oraz procentowa stopa zwrotu. Widok portfela zawiera również podsumowanie całkowitej wartości portfela i łącznego zysku/straty. Użytkownik może zaznaczyć wybraną pozycję z listy – w przyszłości planowane jest umożliwienie dodatkowych akcji (np. usunięcie pozycji lub wyświetlenie wykresu tej spółki).
- **Kalkulator** – oferuje interaktywne narzędzie do prognozowania wyniku inwestycji. W tym widoku użytkownik podaje kwotę planowanej inwestycji (w USD) oraz horyzont czasowy (liczbę dni). Aplikacja prezentuje (po naciśnięciu przycisku „Sprawdź”) **przewidywaną stopę zwrotu** w procentach, **przewidywany zysk/stratę** w dolarach oraz **ocenę ryzyka** dla wybranego instrumentu. Kalkulator korzysta z aktualnie wybranego instrumentu (np. ostatnio wyszukanego na stronie głównej) – jego nazwę i bieżącą cenę widać w nagłówku sekcji.

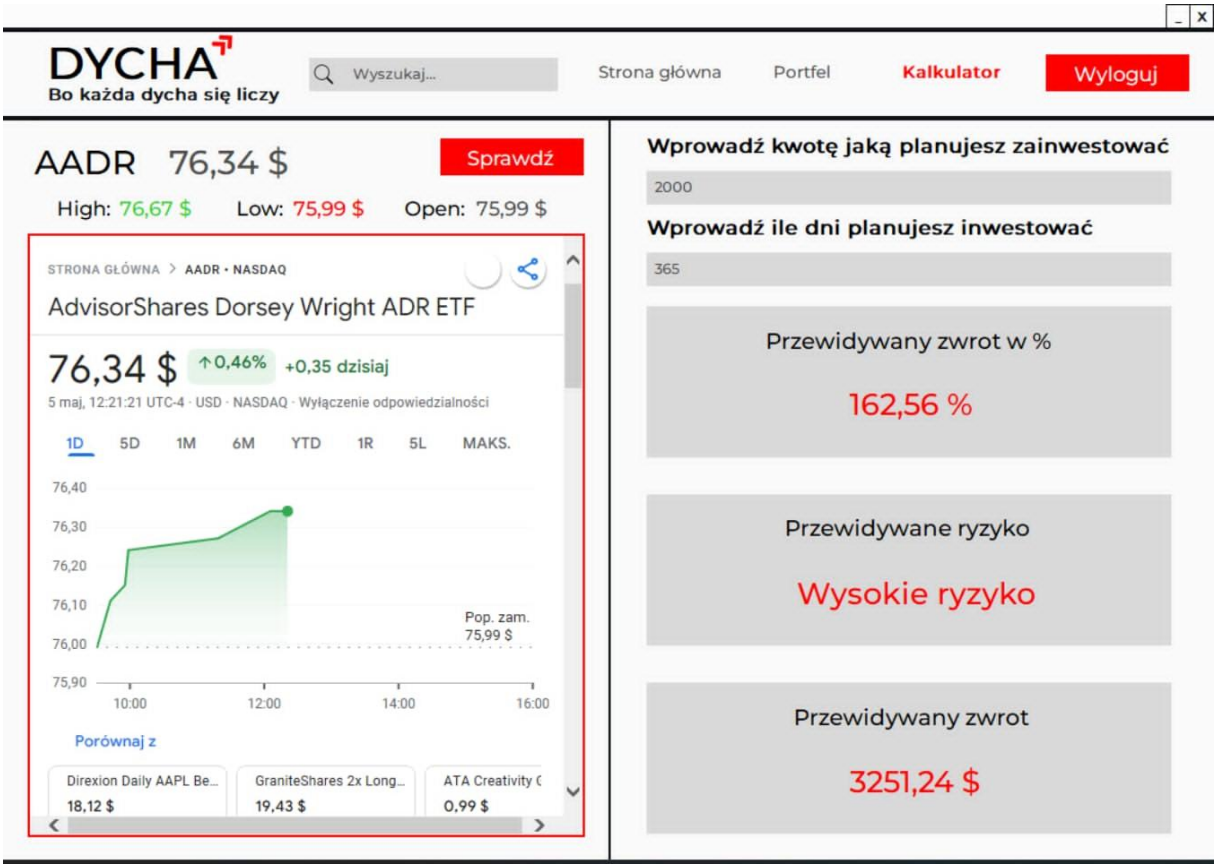
Poniżej przedstawiono przykładowe ekrany interfejsu użytkownika aplikacji „Dycha”:



*Rysunek 1. Główny ekran aplikacji „Dycha” – **Strona główna** z wyświetlonym wykresem wybranego instrumentu (BGRN – iShares USD Green Bond ETF) oraz panelem zakupu po prawej stronie.* Na tym widoku użytkownik może wyszukać instrument w polu **Wyszukaj...** (górny pasek) i po załadowaniu danych zobaczyć aktualny kurs (tu: 46,84 $), zmianę dzienną (−0,12%) oraz wykres notowań intraday. Panel zakupu po prawej umożliwia wpisanie wolumenu i dodanie pozycji do portfela poprzez przycisk **Kup**.



*Rysunek 2. Ekran **Portfel** – lista zakupionych pozycji wraz z ich parametrami.* W przedstawionym przykładzie użytkownik posiada dwie pozycje w akcjach **AMZN** (Amazon). Dla każdej pozycji pokazano wolumen (np. 200 sztuk), wartość rynkową (207,9 $ * 200), cenę otwarcia (cena zakupu; np. 167,32 $), zysk netto (w $) oraz zysk netto % (zielony oznacza zysk, czerwony stratę). W dolnej części okna widoczne jest podsumowanie: **Wartość konta** oraz łączny **Zysk** portfela (tu przykładowo 8341,60 $). Użytkownik może na bieżąco śledzić, jak zmiany cen wpływają na wartość jego inwestycji.



*Rysunek 3. Ekran **Kalkulator** – funkcja prognozowania wyników inwestycji dla wybranego instrumentu (tu: AADR – AdvisorShares Dorsey Wright ADR ETF).* Użytkownik wprowadził przykładowo kwotę 2000 $ oraz horyzont 365 dni, a aplikacja na podstawie danych historycznych obliczyła **przewidywany zwrot** 162,56% oraz **przewidywany zysk** 3251,24 $. Jednocześnie oceniono **przewidywane ryzyko** tej inwestycji jako *wysokie* (na co może wpływać duża zmienność notowań AADR). Wyniki te mają charakter poglądowy – pomagają ocenić potencjał inwestycji i ryzyko, lecz nie stanowią gwarancji osiągnięcia takiego wyniku.

Interfejs użytkownika został zaprojektowany z wykorzystaniem jednolitej kolorystyki oraz czytelnych komponentów (przyciski, pola tabeli, etykiety). Dominującym kolorem akcentującym jest **czerwony** (np. przyciski „Kup", „Wyloguj", etykiety wartości spadków), co spójnie buduje identyfikację wizualną aplikacji. Nawigacja pomiędzy sekcjami (Strona główna, Portfel, Kalkulator) odbywa się poprzez menu u góry okna i jest dostępna na każdym etapie korzystania z aplikacji.

Interfejs sprzętowy: Aplikacja działa na standardowych komputerach osobistych. Minimalne wymagania sprzętowe obejmują: komputer PC klasy **desktop lub laptop** z procesorem min. 2 GHz, **4 GB RAM** oraz ok. 100 MB dostępnej przestrzeni dyskowej (na pliki aplikacji i bufor danych). Do obsługi aplikacji potrzebna jest **klawiatura i mysz** (interakcje przez typowe kontrolki GUI). Aplikacja nie wymaga specjalistycznych urządzeń zewnętrznych.

Interfejs programowy: System integruje się z kilkoma zewnętrznymi usługami/komponentami programowymi:

- **Firebase** – wykorzystany jest moduł uwierzytelniania Firebase Authentication do rejestracji i logowania użytkowników (przy użyciu e-mail i hasła) oraz Cloud Firestore/Realtime Database do przechowywania danych użytkownika (pozycje portfela).

Komunikacja z Firebase odbywa się za pomocą oficjalnych bibliotek SDK dla platformy .NET, co zapewnia bezproblemową integrację z aplikacją desktopową.

- **Finnhub Stock API** – aplikacja korzysta z interfejsu REST API dostawcy danych giełdowych Finnhub. Przy starcie oraz podczas korzystania z aplikacji wysyłane są żądania HTTP (GET) do usług Finnhub w celu pobrania aktualnych informacji o notowaniach oraz danych historycznych wymaganych np. dla kalkulatora. Dostęp do API wymaga klucza API (posiada go aplikacja – konieczne było założenie darmowego konta Finnhub). Obowiązuje limit **60 zapytań na minutę** dla planu darmowego; aplikacja została zaprojektowana tak, aby nie przekraczać tego limitu (zapytania są odsyłane tylko przy rzeczywistej aktywności użytkownika, np. wyszukaniu nowego symbolu czy odświeżeniu danych).
- **Usługa Google Finance (web)** – w celu wyświetlania interaktywnych wykresów kursów, wykorzystano podejście osadzenia przeglądarki internetowej wewnątrz aplikacji. Komponent **WebBrowser** ładuje dedykowaną stronę serwisu Google Finance dla wybranego instrumentu, dzięki czemu użytkownik ma dostęp do profesjonalnie renderowanego wykresu świecowego oraz powiązanych danych (jak na stronie www). Jest to rozwiązanie wykorzystujące istniejące narzędzie webowe zamiast implementowania własnego modułu rysowania wykresów. Wymaga ono dostępu do internetu oraz potencjalnie obsługi elementów dynamicznych strony przez kontrolkę przeglądarki.

Interfejs komunikacyjny: Aplikacja **komunikuje się z zewnętrznymi usługami** poprzez internet, używając standardowych protokołów:

- Połączenie z API Finnhub odbywa się przez **HTTPS** (zapewnia to szyfrowanie transmisji danych finansowych i bezpieczeństwo komunikacji). Żądania zawierają klucz API w nagłówkach, a odpowiedzi zwracane są w formacie JSON, który aplikacja następnie interpretuje.
- Połączenie z Firebase również wykorzystuje protokół HTTPS i/lub gniazda bezpieczne (w przypadku funkcji w czasie rzeczywistym). Cała komunikacja z bazą jest szyfrowana i wymaga uwierzytelnienia (użytkownik musi być zalogowany, by uzyskać dostęp do swoich danych).
- Brak jest bezpośredniej komunikacji z innymi systemami użytkownika (np. nie ma integracji poprzez pliki ani interfejsy z programami typu Excel – użytkownik nie importuje danych lokalnie, a wszystkie dane pobierane są z chmury). Wewnętrznie, moduły aplikacji komunikują się poprzez wywołania metod i zdarzeń w ramach aplikacji (np. portfel użytkownika jest osadzony w bazie danych Firebase).

Podsumowanie interfejsów zewnętrznych: Aplikacja **„Dycha”** jest typowym rozwiązaniem desktopowym działającym w środowisku Windows, w silnym stopniu opierającym się na usługach sieciowych. Interfejs użytkownika kładzie nacisk na czytelność i prostotę obsługi, interfejs sprzętowy wymaga jedynie podstawowego komputera, zaś interfejsy programowe i komunikacyjne integrują rozwiązanie z nowoczesnymi usługami chmurowymi i danymi giełdowymi czasu rzeczywistego.

3.2. Wymagania funkcjonalne

Wymagania funkcjonalne opisują szczegółowo zachowanie systemu **„Dycha”** z perspektywy użytkownika i przetwarzanych danych. Poniżej przedstawiono kluczowe wymagania pogrupowane według obszarów funkcjonalnych:

Rejestracja i logowanie:

- **FR1. Rejestracja nowego użytkownika:** System musi umożliwić stworzenie nowego konta. Użytkownik podaje niezbędne dane rejestracyjne – w wersji 1.0 jest to adres e-mail (będący jednocześnie loginem) oraz hasło. Rejestracja nie wymaga żadnych kodów ani tokenów (brak „tokenu od administratora” – aplikacja jest otwarta dla wszystkich chętnych użytkowników). Po walidacji poprawności danych (np. format e-mail, minimalna długość hasła) konto zostaje utworzone w bazie Firebase, a użytkownik może się zalogować.
- **FR2. Logowanie użytkownika:** System musi pozwolić zarejestrowanemu użytkownikowi na uwierzytelnienie się za pomocą e-maila i hasła. Po wprowadzeniu danych logowania i ich potwierdzeniu (zweryfikowaniu z danymi w Firebase) aplikacja otwiera główny interfejs użytkownika. W przypadku błędnych danych logowania system wyświetla komunikat o nieudanym logowaniu i umożliwia ponowną próbę. Użytkownik musi być zalogowany, aby uzyskać dostęp do innych funkcji aplikacji.
- **FR3. Wylogowanie:** System powinien zapewnić funkcję wylogowania. Użytkownik wybiera opcję **Wyloguj**, po czym aplikacja zamyka bieżącą sesję i powraca do ekranu logowania. Wymaganie to gwarantuje, że po zakończeniu pracy użytkownika jego dane nie będą dostępne bez ponownego podania hasła.

Zarządzanie portfelem i danymi rynkowymi:

- **FR4. Wyszukiwanie instrumentów finansowych:** System musi umożliwić użytkownikowi wyszukiwanie interesujących go instrumentów (akcji, ETF itp.) po nazwie lub symbolu. W tym celu udostępniane jest pole wyszukiwarki. Po wpisaniu zapytania i zatwierdzeniu, aplikacja łączy się z API giełdowym i pobiera informacje o pasujących instrumentach. Co najmniej bieżący kurs i nazwa instrumentu o podanym symbolu powinny zostać wyświetlone. Jeżeli instrument nie zostanie znaleziony, użytkownik otrzymuje stosowny komunikat (np. “Brak danych dla podanego symbolu”).
- **FR5. Wyświetlanie bieżących danych rynkowych:** Dla wybranego przez użytkownika instrumentu system wyświetla aktualne dane: ostatnią cenę, zmianę ceny (absolutną i procentową) w ujęciu dziennym, kurs otwarcia, dzisiejsze minimum i maksimum, poprzednie zamknięcie itp. Te informacje są pobierane z API na bieżąco i odświeżane przy każdym wyborze nowego instrumentu (oraz mogą być okresowo odświeżane np. co minutę dla aktywnie wyświetlanego instrumentu, o ile nie przekroczy to limitów API). Dane prezentowane są w czytelnej formie obok wykresu lub nad nim.
- **FR6. Wyświetlanie wykresu notowań:** System powinien pokazać użytkownikowi wykres historyczny cen wybranego instrumentu. Wymagane jest wsparcie co najmniej dla interwału dziennego (1D) z bieżącej sesji oraz możliwość przełączenia zakresu na 5 dni, 1 miesiąc, 6 miesięcy, rok, 5 lat itp. (zakresy dostępne na Google Finance). Implementacja odbywa się poprzez osadzony komponent przeglądarki, stąd wymaganiem jest poprawne załadowanie strony wykresu i jej wyświetlenie w ramach okna aplikacji. Użytkownik musi

mieć możliwość odczytania wartości z osi czasu i osi cen z wykresu (np. poprzez najechanie kursorem na punkt wykresu – ta funkcjonalność jest zapewniona przez Google Finance).

- **FR7. Dodawanie pozycji do portfela (zakup):** System musi umożliwiać użytkownikowi dodanie aktualnie wyświetlanego instrumentu do swojego portfela inwestycyjnego. Wymaga się, aby użytkownik mógł wprowadzić **wolumen** (liczbę akcji/jednostek, które chce „kupić") w dedykowanym polu. Po zatwierdzeniu (kliknięciu przycisku **Kup**) aplikacja pobiera najnowszą cenę instrumentu i tworzy w bazie danych nową pozycję portfela zawierającą: identyfikator użytkownika, symbol instrumentu, wolumen, cenę zakupu (bieżącą cenę rynkową) oraz datę transakcji. Użytkownik otrzymuje potwierdzenie dodania pozycji (np. komunikat „Dodano do portfela" lub automatyczne pojawienie się nowej pozycji na liście portfela). **Warunek początkowy:** użytkownik musi być zalogowany i mieć wybrany instrument (załadowane dane) na stronie głównej. **Warunki końcowe:** nowa pozycja jest widoczna w Portfelu, a dane są zapisane w Firebase.

- **FR8. Przegląd portfela:** System ma wyświetlać listę wszystkich pozycji dodanych przez zalogowanego użytkownika. Dla każdej pozycji prezentowane są informacje zgodnie z opisem interfejsu (sekcja portfela). Wymagane jest, aby te informacje były obliczane dynamicznie: np. bieżąca wartość rynkowa i zysk/strata powinny być kalkulowane na podstawie aktualnej ceny instrumentu (pobranej z API w momencie wyświetlania portfela lub odświeżania widoku). Użytkownik musi mieć możliwość zaktualizowania danych portfela (np. przyciskiem "Odśwież" lub automatycznie, co pewien interwał czasu) tak, aby odzwierciedlały one zmiany cen na rynku.

- **FR9. Obliczanie średniej ceny zakupu:** Jeżeli użytkownik doda wielokrotnie ten sam instrument (np. dokonał zakupu akcji spółki X w różnych dniach po różnych cenach), system powinien wyświetlać jedną skonsolidowaną pozycję w portfelu dla danej spółki lub odrębne pozycje – w zależności od przyjętego modelu. W wersji 1.0 przyjęto prostszy model: **każda transakcja zakupu jest wyświetlana osobno** (np. jak w Rysunku 2 widać dwie pozycje AMZN dodane oddzielnie). Średnia cena może być jednak obliczana i wyświetlana jako dodatkowa informacja, jeśli pozycje zostaną kiedyś scalone. To wymaganie może zostać dopracowane w przyszłych wersjach (np. sumowanie pozycji po tym samym symbolu).

- **FR10. Usunięcie pozycji z portfela:** *(niezaimplementowane w v1.0)* W docelowym systemie powinna istnieć możliwość usunięcia istniejącej pozycji z portfela (symulacja sprzedaży całości posiadanych udziałów). Ponieważ funkcja ta nie została zaimplementowana w bieżącej wersji, użytkownik musi sam mentalnie "symulować" sprzedaż, ewentualnie korzystając z kalkulatora, a portfel w aplikacji rośnie tylko poprzez dodawanie nowych pozycji. To ograniczenie powinno zostać zaznaczone (co czynimy tutaj) i docelowo usunięte w kolejnych wydaniach.

Analiza i kalkulacje inwestycyjne:

- **FR11. Kalkulator zwrotu i ryzyka:** System musi umożliwić użytkownikowi przeprowadzenie analizy hipotetycznej inwestycji. Główne wejścia to: kwota inwestycji (w USD, wartość całkowita) oraz okres inwestycji (liczba dni). Dodatkowo wykorzystywany jest kontekst wybranego instrumentu (system przyjmuje, że kalkulacja dotyczy instrumentu, który obecnie jest aktywny/wybrany – np. użytkownik wyszukał go

na stronie głównej przed przejściem do kalkulatora). Po uruchomieniu kalkulacji (przycisk **Sprawdź**):

- System pobiera z API historyczne dane cenowe danego instrumentu (np. z okresu ostatnich *N* dni odpowiadających podanemu horyzontowi lub inne istotne statystyki, np. średnią stopę zwrotu roczną).
- Na podstawie tych danych system oblicza *przewidywaną stopę zwrotu* – np. może to być historyczna stopa zwrotu za zadany okres w przeszłości, uśredniona i zaprognozowana na przyszłość (dla uproszczenia). W wyniku kalkulacji otrzymujemy procent, który system wyświetla jako oczekiwany procentowy zwrot z inwestycji.
- System oblicza *przewidywany zysk/stratę* w dolarach jako: (przewidywana stopa zwrotu * kwota inwestycji). Jeśli stopa zwrotu jest pozytywna, mówimy o zysku, jeśli negatywna – o stracie.
- System określa *poziom ryzyka* – na podstawie zmienności historycznej kursu. Przykładowo, jeśli dany instrument cechował się dużymi wahaniami cen (wysoka odchylenie standardowe zmian dziennych), system wyświetli "Wysokie ryzyko"; dla stabilniejszych instrumentów może to być "Niskie" lub "Umiarkowane" ryzyko. (Metodologia klasyfikacji ryzyka może być uproszczona, np. na sztywno zdefiniowane progi zmienności).
- Wyniki (zwrot %, zysk $ i ocena ryzyka) są prezentowane na ekranie kalkulatora w wyróżnionych polach.
- **Warunki początkowe:** użytkownik jest zalogowany i ma wybrany instrument (lub domyślnie – jeśli kalkulator dopuszcza wybór instrumentu niezależnie, to musi istnieć pole do wyboru instrumentu; w wersji v1.0 instrument jest dziedziczony z ekranu głównego). Dane historyczne muszą być dostępne w API (jeśli nie – system poinformuje o braku danych).
- **Rezultat:** użytkownik otrzymuje oszacowanie możliwego wyniku inwestycji oraz ryzyka, co wspomaga go w decyzji, czy warto zainwestować.

- **FR12. Prezentacja wyniku kalkulatora:** System powinien prezentować wyniki kalkulacji w zrozumiały sposób, np.: "Przewidywany zwrot: 150% (wysoki zysk), Przewidywany zysk: 3000 $, Przewidywane ryzyko: Wysokie". Prezentacja musi być czytelna, wartości numeryczne odpowiednio zaokrąglone (np. do dwóch miejsc po przecinku dla procentów i pełnych centów dla dolarów). Kolorystyka może być użyta pomocniczo (np. zielony odcień dla dużego zysku, czerwony dla straty, żółty dla ostrzeżenia o ryzyku).
- **FR13. Ograniczenia kalkulatora:** Jeżeli użytkownik wprowadzi **nieprawidłowe dane** (np. litery zamiast liczb, ujemną kwotę lub 0 dni), system powinien wyświetlić komunikat o błędzie i poprosić o korektę danych wejściowych. Ponadto, jeśli API nie zwróci wymaganych informacji (np. brak danych historycznych za wskazany okres lub przekroczony limit zapytań), kalkulacja powinna zostać przerwana z komunikatem o braku możliwości przeprowadzenia analizy w danym momencie.

Powiadomienia i komunikaty:

- **FR14. Informowanie o błędach sieciowych:** W razie utraty połączenia internetowego lub problemów z dostępem do zewnętrznych usług (API, Firebase), system powinien poinformować użytkownika o braku łączności bądź opóźnieniu w pobieraniu danych. Np. jeśli na ekranie głównym nie uda się pobrać aktualnych notowań, zamiast pustych pól system wyświetli komunikat: "Błąd połączenia – sprawdź internet". Podobnie przy próbie logowania bez internetu – komunikat o niemożności zalogowania.
- **FR15. Potwierdzenia akcji użytkownika:** Kluczowe akcje (jak dodanie pozycji do portfela, rejestracja konta) muszą być potwierdzane, czy to poprzez wyświetlenie komunikatu ("Pozycja dodana") czy poprzez odświeżenie interfejsu, aby użytkownik miał jasność, że operacja się powiodła. W razie niepowodzenia (np. dodanie pozycji nie powiodło się ze względu na brak danych) również należy przedstawić odpowiedni komunikat.

Powyższe wymagania funkcjonalne pokrywają najważniejsze oczekiwane zachowania systemu. Wszystkie funkcje są dostępne dla roli **Użytkownik**, ponieważ – jak wspomniano – aplikacja nie różnicuje uprawnień między różnymi użytkownikami. Każdy użytkownik może więc rejestrować się, logować, dodawać pozycje i korzystać z kalkulatora na równych prawach.

3.3. Wymagania użyteczności

Wymagania dotyczące użyteczności określają, w jaki sposób interfejs i działanie systemu **„Dycha"** zapewniają pozytywne doświadczenie użytkownika:

- **UR1. Przejrzystość interfejsu:** Interfejs użytkownika powinien być **przejrzysty i intuicyjny**, aby użytkownik mógł łatwo poruszać się między ekranami i wykonywać zadania przy minimalnej liczbie kroków. Kluczowe funkcje (jak przełączanie zakładek, przycisk dodawania do portfela, uruchomienie kalkulacji) muszą być wyeksponowane i opisane zrozumiałymi etykietami. Layout aplikacji powinien być spójny – np. nagłówek i menu na wszystkich ekranach jest taki sam, przyciski akcji (Kup, Wyloguj, Sprawdź) wyróżnione kolorem.
- **UR2. Spójność nawigacji:** Aplikacja zachowuje **spójny schemat nawigacji** – menu główne jest zawsze widoczne na górze, powrót do ekranu głównego jest zawsze możliwy kliknięciem logo lub przycisku "Strona główna". Użytkownik nie gubi się w aplikacji; liczba poziomów zagnieżdżenia interfejsu jest minimalna (praktycznie jedna warstwa zakładek).
- **UR3. Responsywność na akcje użytkownika:** System powinien reagować na działania użytkownika w czytelny sposób. Po kliknięciu aktywnego elementu (np. przycisku **Kup** czy **Sprawdź**) użytkownik powinien otrzymać natychmiastowy sygnał, że akcja jest przetwarzana – np. krótka zmiana stanu przycisku, pojawienie się animacji ładowania danych, itp. Jeśli wykonanie operacji trwa dłużej niż np. 1–2 sekundy (np. pobieranie danych z internetu), wskazane jest pokazanie „spinnera" lub komunikatu „Trwa ładowanie...", aby użytkownik wiedział, że aplikacja pracuje.
- **UR4. Walidacja danych wejściowych:** Interfejs powinien zapobiegać typowym błędom użytkownika poprzez walidację wprowadzanych danych. Np. pola do wpisywania liczb (wolumen, kwota, liczba dni) powinny akceptować wyłącznie cyfry i ewentualnie ograniczać zakres (wolumen > 0, liczba dni > 0, kwota > 0). W razie błędu użytkownika

(np. wpisanie tekstu w pole liczby) system od razu sygnalizuje problem (np. czerwone obramowanie pola, komunikat pod polem).

- **UR5. Czytelność informacji:** Wszystkie prezentowane liczby i teksty muszą być czytelne. Warto zadbać o odpowiedni kontrast kolorów (np. czerwony tekst na białym tle jest widoczny, zielony na białym również; nie używać jasnoszarego małego fontu na białym tle itp.). Wykres osadzony powinien mieć czytelne opisy osi – w razie gdyby interfejs wbudowanej strony był nieczytelny, należy rozważyć własne uzupełnienie (np. pod wykresem dodano legendę lub opis). Czcionki w aplikacji powinny mieć rozmiar co najmniej ~11-12pt dla treści głównej i większe dla nagłówków.
- **UR6. Język i terminologia:** Aplikacja **„Dycha”** komunikuje się z użytkownikiem w języku polskim (zgodnie z grupą docelową). Wszystkie etykiety, komunikaty i instrukcje są sformułowane po polsku, z użyciem jednoznacznej terminologii finansowej tam, gdzie to konieczne (np. użyto terminu “Stopa zwrotu” zamiast potocznego “procent zysku”). Unikamy żargonu informatycznego – np. zamiast komunikatu “Connection error: API limit exceeded” pojawi się komunikat po polsku “Przekroczono limit zapytań. Spróbuj ponownie za minutę.”.
- **UR7. Estetyka i profesjonalny wygląd:** Interfejs został zaprojektowany zgodnie z zasadą minimalizmu – **„mniej znaczy więcej”**. Tło jest jednolite, elementy są rozłożone symetrycznie, przewijanie wewnątrz okna ograniczone do sytuacji koniecznych (np. lista portfela może przewijać się, jeśli pozycji jest dużo). Dzięki temu aplikacja sprawia wrażenie profesjonalnej, a jednocześnie przyjaznej dla użytkownika bez zbędnego natłoku informacji.

Wymagania użytecznościowe koncentrują się na tym, by użytkownik mógł skupić się na zadaniach związanych z analizą inwestycji, a nie na obsłudze samej aplikacji. Intuicyjność i przejrzystość interfejsu **„Dycha”** mają kluczowe znaczenie dla jej akceptacji przez docelowych odbiorców.

3.4. Wymagania wydajnościowe

Wymagania dotyczące wydajności określają oczekiwania co do szybkości działania aplikacji **„Dycha”** i jej zachowania pod obciążeniem:

- **WR1. Czas odpowiedzi interfejsu:** Aplikacja powinna działać płynnie na docelowym sprzęcie. Podstawowe akcje użytkownika (nawigacja między ekranami, wpisywanie danych, kliknięcia przycisków) powinny być obsługiwane **natychmiastowo** (w czasie nieodróżnialnym od rzeczywistego – poniżej 0,1 s dla rejestracji kliknięcia czy przełączenia widoku).
- **WR2. Czas ładowania danych z API:** Przy założeniu stabilnego połączenia internetowego, pobranie aktualnych notowań wybranego instrumentu z Finnhub API i wyświetlenie ich w aplikacji powinno zajmować **maksymalnie 2-3 sekundy**. W praktyce większość odpowiedzi API jest otrzymywana w czasie poniżej 1 sekundy; dodatkowy czas jest wymagany na przetworzenie danych i aktualizację interfejsu. Jeśli odpowiedź nie nadejdzie w ciągu 5 sekund, system powinien powiadomić o opóźnieniu (jak wspomniano w wymaganiach funkcjonalnych FR14).
- **WR3. Wydajność aktualizacji portfela:** Gdy użytkownik przełącza się na ekran Portfela, aplikacja może potrzebować pobrać aktualne ceny dla wielu instrumentów (wszystkich

posiadanych pozycji). System powinien wykonywać te zapytania efektywnie, np. równolegle lub w paczkach, aby odświeżenie portfela nastąpiło w rozsądnym czasie. Dla portfela zawierającego do 20 pozycji zaleca się czas odświeżenia nie dłuższy niż **5 sekund** dla wszystkich danych (średnio 0,25 s na pozycję).

- **WR4. Limit zapytań do API:** Aplikacja musi respektować ograniczenie **60 zapytań/minutę** do Finnhub API. W związku z tym zapytania powinny być wysyłane tylko wtedy, kiedy to konieczne (np. unikamy odpytywania API co sekundę bez potrzeby). Ponadto w przypadku intensywnego użytkowania (np. użytkownik szybko przełącza instrumenty, używa kalkulatora wiele razy) aplikacja powinna mieć mechanizm kolejkowania lub ograniczania zapytań, by nie przekroczyć limitu (np. jeżeli użytkownik wywoła kilkanaście akcji w krótkim czasie, system może chwilowo wstrzymać część zapytań i poinformować o potrzebie odczekania).
- **WR5. Wykorzystanie zasobów:** Aplikacja powinna być zoptymalizowana pod kątem zużycia pamięci i CPU. Jako że jest to aplikacja desktopowa, zakłada się dostępność sporych zasobów, ale mimo to: zużycie pamięci RAM przez aplikację **nie powinno przekraczać 200-300 MB** podczas normalnej pracy (dla porównania, to niewielki ułamek dostępnej pamięci w nowoczesnych PC). Wykorzystanie procesora również powinno być niskie – intensywniejsze obliczenia mogą wystąpić np. podczas kalkulacji ryzyka, lecz nawet wtedy CPU może zostać obciążony np. w 50% przez ułamek sekundy. Aplikacja nie powinna powodować zauważalnego spowolnienia całego systemu operacyjnego.
- **WR6. Skalowalność jednostanowiskowa:** Ponieważ jest to aplikacja kliencka działająca po stronie użytkownika, klasycznie rozumiana skalowalność po stronie serwera nie ma tu zastosowania (ciężar obsługi podziału na użytkowników spoczywa na usługach chmurowych: Firebase i Finnhub). Jednakże aplikacja powinna **radzić sobie z większą ilością danych użytkownika** – np. portfelem zawierającym setki pozycji lub zapytaniami o wiele instrumentów. W takich przypadkach interfejs nadal powinien działać poprawnie (być może z nieco wydłużonym czasem ładowania). Baza Firebase jest skalowalna automatycznie w chmurze, a API Finnhub jest w stanie obsłużyć nawet duże wolumeny zapytań (przy wykupieniu wyższego planu), więc w razie rosnących potrzeb można zwiększyć limity bez przebudowy aplikacji.

Podsumowując, aplikacja **„Dycha”** w wersji 1.0 jest **wymagająca** głównie pod względem stałego dostępu do internetu i dotrzymania kroku strumieniowi danych giełdowych, natomiast jej obciążenie dla komputera użytkownika jest stosunkowo niewielkie. Wydajność została uznana za **akceptowalną**, jeśli spełnione są powyższe założenia (krótkie czasy reakcji, sprawne pobieranie danych w granicach limitów, niskie zużycie zasobów lokalnych).

3.5. Wymagania bazodanowe

Aplikacja **„Dycha”** wykorzystuje **bazę danych Firebase** do przechowywania danych. Oznacza to, że w przeciwieństwie do tradycyjnych aplikacji desktopowych, które mogłyby używać lokalnej bazy SQL (np. SQLite) lub własnego serwera, tutaj wszystkie dane użytkownika są **przechowywane zdalnie w chmurze** (NoSQL). Poniżej wyszczególniono wymagania dotyczące modelu danych i ich przechowywania.

3.5.1. Struktura danych

W Firebase dane mogą być przechowywane w formie dokumentów (Cloud Firestore) lub w formacie drzewiastym (Realtime Database). Niezależnie od wybranej opcji (która implementacyjnie została już ustalona w trakcie prac – Realtime Database), logiczna struktura danych prezentuje się następująco:

- **Kolekcja Użytkownicy** – nadrzędna kolekcja dokumentów, gdzie każdy dokument odpowiada jednemu zarejestrowanemu użytkownikowi aplikacji.
    - **Dokument Użytkownik** (ID dokumentu może być np. UID nadany przez Firebase Auth, lub email użytkownika): zawiera podstawowe informacje profilowe, np. adres e-mail (dla referencji, ponieważ autentykacja jest zarządzana przez Firebase Auth), datę rejestracji, ewentualnie nazwę wyświetlaną itp.
    - **Podkolekcja Portfel** – w ramach dokumentu użytkownika istnieje kolekcja "Portfel" zawierająca dokumenty opisujące poszczególne pozycje inwestycyjne dodane przez tego użytkownika.
        - **Dokument Pozycja**: Każdy dokument przechowuje dane o jednej pozycji/zakupie:
            - Pole **symbol** (np. "AMZN"),
            - **nazwa** instrumentu (opcjonalnie, np. "Amazon.com Inc." dla łatwiejszej identyfikacji),
            - **wolumen** (liczba jednostek),
            - **cenaZakupu** (cena za jednostkę w momencie dodania, np. 167.32),
            - **wartoscTransakcji** (wolumen * cenaZakupu, do celów referencyjnych),
            - **dataZakupu** (timestamp dodania do portfela),
            - Ewentualnie **średniaCena** (jeśli transakcje tego samego instrumentu są scalane – w bieżącej wersji każda transakcja to oddzielny dokument, więc średniaCena = cenaZakupu dla pojedynczej transakcji).
        - Kluczem dokumentu pozycji jest wielkość portfela powiększona o jeden.
- **Kolekcja Transakcje** *(opcjonalna)* – można ewentualnie utrzymywać oddzielną kolekcję wszystkich transakcji (zakupów) dla celów logów lub gdyby portfel scalał pozycje. W bieżącej implementacji nie ma osobnej kolekcji transakcji – każda pozycja portfela reprezentuje pojedynczą transakcję.
- **Kolekcja (lub dokument) Ustawienia** – ewentualne dane globalne aplikacji (np. lista obsługiwanych rynków, limity, parametry kalkulatora). Obecnie większość logiki jest zakodowana w aplikacji, więc ta część nie jest wykorzystywana.

Relacje między danymi: Każdy użytkownik posiada swój własny zestaw dokumentów pozycji w portfelu – **dostęp do nich ma tylko on sam** (reguły bezpieczeństwa Firebase powinny to

zapewnić). Nie ma bezpośrednich powiązań między danymi różnych użytkowników (brak relacji między portfelami różnych osób). W ramach danych jednego użytkownika kluczową relacją jest powiązanie dokumentu pozycji z instrumentem giełdowym; ponieważ aplikacja nie przechowuje pełnych informacji o instrumentach (te pobiera dynamicznie z API), dokument pozycji zawiera tylko symbol i podstawowe dane potrzebne do przeliczeń. W razie potrzeby uzyskania aktualnych szczegółów (ceny bieżącej, nazwy spółki, itp.) aplikacja zawsze odpyta Finnhub API zamiast trzymać te dane w bazie (unikamy redundancji i potencjalnej nieaktualności danych lokalnych).

Podsumowanie struktury: model danych jest **stosunkowo prosty** – sprowadza się do skojarzenia użytkownika z listą pozycji w jego portfelu. Dzięki użyciu bazy NoSQL Firebase, nie ma tu skomplikowanych relacji czy joinów; odczyt danych portfela polega na pobraniu dokumentów z kolekcji Portfel konkretnego użytkownika, a zapis dodania pozycji – na wstawieniu nowego dokumentu do tej kolekcji.

3.5.2. Wymagania ogólne dotyczące bazy

- **BR1. Bezpieczeństwo dostępu:** Dostęp do bazy Firebase musi być zabezpieczony regułami autoryzacji. Tylko uwierzytelnieni użytkownicy mogą odczytać/dodać swoje dane. Reguły bezpieczeństwa powinny zapobiegać odczytowi danych cudzych portfeli (każdy użytkownik może czytać i pisać jedynie w ścieżce /Użytkownicy/{swoje userId}/Portfel/...). Dostęp anonimowy lub nieautoryzowany jest zabroniony.
- **BR2. Trwałość danych:** Dane przechowywane w Firebase powinny zachować trwałość i spójność. Oznacza to, że nawet po wyłączeniu aplikacji czy ponownym zalogowaniu użytkownika z innego urządzenia, dane portfela muszą być dostępne i kompletne. Firebase jako usługa chmurowa gwarantuje replikację i trwałość zapisów, więc ryzyko utraty danych jest minimalizowane (należy upewnić się, że operacje zapisu są poprawnie obsłużone po stronie aplikacji – np. potwierdzane sukcesem).
- **BR3. Spójność danych finansowych:** System powinien zapewnić, że dane zapisywane do bazy są poprawne i spójne. Np. nie wolno dopuścić do zapisu pozycji z wolumenem 0 lub ujemnym, cena zakupu powinna być liczbą dodatnią, symbol instrumentu powinien być niepusty i w prawidłowym formacie (litery/tickery). Walidacja tych warunków powinna odbywać się na poziomie aplikacji przed wysłaniem danych do bazy.
- **BR4. Wydajność zapytań bazodanowych:** Ponieważ baza może zawierać wielu użytkowników i wiele pozycji, należy zwrócić uwagę na odpowiednie indeksowanie zapytań. W przypadku Cloud Firestore, proste odczyty kolekcji pod dokumentem użytkownika są indeksowane domyślnie (klucz główny). W razie potrzeby filtrowania (np. wyszukiwanie w portfelu po symbolu, gdyby dodano taką funkcję), trzeba by zadbać o utworzenie ewentualnych dodatkowych indeksów. Obecnie podstawowe operacje (pobranie całej listy pozycji) powinny wykonywać się bardzo szybko ze względu na ograniczony rozmiar danych per użytkownik.
- **BR5. Brak lokalnej bazy cache:** Aplikacja nie utrzymuje lokalnej kopii całej bazy (poza ewentualnym cachingiem mechanizmu Firebase). Nie ma wymagania instalacji żadnego serwera bazy danych na komputerze użytkownika. Wszystkie dane są pozyskiwane zdalnie w momencie potrzebnym. Jedynymi danymi ewentualnie trzymanymi lokalnie mogą być ustawienia konfiguracyjne aplikacji (np. ostatnio zalogowany użytkownik, preferencje UI) – które mogą być zapisane np. w rejestrze Windows lub pliku

konfiguracyjnym. Jednak dane portfela i notowania nie są trwale przechowywane lokalnie.

- **BR6. Skalowalność bazy:** Firebase jako baza w chmurze skaluje się automatycznie – rosnąca liczba użytkowników lub danych może potencjalnie generować większe koszty lub wymagać przejścia na wyższy plan, ale z perspektywy aplikacji nie zmienia to sposobu działania. Dla potrzeb SRS wystarczy stwierdzić, że baza danych **obsłuży zakładaną liczbę użytkowników** (np. do kilkuset lub kilku tysięcy w początkowej fazie) bez utraty wydajności.

3.6. Ograniczenia projektowe

Ta sekcja omawia wszelkie przyjęte z góry założenia projektowe, narzucone technologie i inne ograniczenia, które wpływają na kształt rozwiązania **„Dycha”**:

- **Ograniczenie platformowe:** Aplikacja została zaprojektowana i wykonana jako **aplikacja desktopowa Windows Forms w C#**, co oznacza, że jest uruchamialna tylko na systemach operacyjnych **Microsoft Windows** (Windows 10 / 11). Nie przewidziano wsparcia dla macOS czy Linux – aby skorzystać z aplikacji na tych systemach, użytkownik musiałby użyć narzędzi typu emulator lub warstwy zgodności (co nie jest oficjalnie wspierane). Decyzja o platformie Windows wynikała z dostępności bibliotek finansowych oraz prostoty implementacji interfejsu graficznego dla zespołu programistów.
- **Technologie i narzędzia:** W projekcie postanowiono wykorzystać istniejące **usługi chmurowe** zamiast budować wszystko od podstaw. Stąd wybór padł na:
    - Firebase (zamiast własnego serwera bazy danych) – co ogranicza konieczność zarządzania serwerem, ale wymusza pewien model danych (NoSQL) i zależność od usług Google.
    - Finnhub API (zamiast np. budowania własnego scrapera danych giełdowych lub korzystania z płatnych dostawców) – co narzuca limit zapytań i konieczność posiadania klucza API, ale daje gotowe, aktualne dane giełdowe.
    - Komponent WebBrowser i Google Finance do wykresów (zamiast integracji np. z biblioteką wykresów jak Chart.js czy tworzenia wykresów w .NET) – takie rozwiązanie znacząco uprościło implementację wykresów, ale wiąże się z zależnością od strony trzeciej (Google) i wymaga połączenia internetowego podczas korzystania z wykresu. W razie braku dostępu do Google (np. problemy z internetem lub zmiana strony), wykresy mogą się nie wyświetlać – jest to akceptowalne ograniczenie, przyjęte dla zmniejszenia nakładu pracy.
- **Brak trybu offline:** Aplikacja **nie oferuje trybu offline**. Ze względu na charakter (ciągłe pobieranie aktualnych danych rynkowych) i wykorzystane rozwiązania (chmura), wymagane jest stałe połączenie internetowe podczas korzystania z głównych funkcji. W trybie offline użytkownik mógłby co najwyżej przeglądać ostatnio załadowane dane portfela (gdyby zostały zcache’owane), ale nie zostało to zaimplementowane. Ten świadomy kompromis oznacza, że jednym z ograniczeń jest konieczność posiadania internetu – co w obecnych czasach dla użytkowników nie jest dużym problemem, ale warto to zaznaczyć.

- **Zgodność przeglądarki wbudowanej:** Komponent WebBrowser użyty w aplikacji korzysta z silnika Internet Explorer/Edge (w zależności od wersji .NET i ustawień systemu). To narzuca pewne ograniczenia co do zgodności z nowoczesnymi stronami WWW. Założono, że Google Finance będzie poprawnie działać w tym środowisku (co potwierdzono testami). W przyszłości, jeśli strona zaczęłaby wykorzystywać funkcje nieobsługiwane, konieczne może być przejście na nowszy komponent (np. WebView2 z silnikiem Chromium).
- **Bezpieczeństwo danych:** Aplikacja w wersji 1.0 **nie implementuje własnych mechanizmów szyfrowania danych** na urządzeniu. Polega na zabezpieczeniach zapewnianych przez Firebase (przechowywanie haseł w Firebase Authentication – co wewnętrznie je szyfruje, ale jest to poza aplikacją) oraz na szyfrowaniu komunikacji HTTPS. Nie zastosowano np. lokalnego szyfrowania hasła przed wysłaniem (przesyłane jest wprost przez bezpieczny kanał do Firebase Auth) ani dodatkowego szyfrowania danych w bazie – uznano, że Firebase zapewnia wystarczające bezpieczeństwo (dane na serwerach Google są przechowywane w formie zaszyfrowanej i dostęp do nich wymaga kluczy, które ma tylko autoryzowany system). To uproszczenie było akceptowalne w ramach projektu studenckiego, jednak należy być świadomym, że pewnym ograniczeniem jest tu **zaufanie do dostawcy usług**. Wrażliwe dane (jak hasła) nie są hashowane po stronie klienta – aplikacja nie oferuje takiej funkcji.
- **Brak integracji z plikami lokalnymi:** W analizowanej wersji zrezygnowano z funkcji importu czy eksportu danych do plików takich jak CSV czy Excel. Jest to ograniczenie funkcjonalne – użytkownik nie może np. zaimportować listy transakcji z pliku ani wyeksportować raportu do arkusza. W zamian skupiono się na synchronizacji chmurowej i prezentacji danych w aplikacji. Ewentualna integracja z plikami może być rozważona w przyszłości, ale wymagałaby dodatkowego nakładu pracy (np. parser CSV) oraz mogłaby rodzić kwestie zgodności formatów. Ograniczenie to zostało przyjęte, aby zawęzić zakres projektu i skoncentrować się na kluczowych aspektach (co jest zgodne z założeniem prostoty – nie dodajemy zbędnych opcji).
- **Termin realizacji:** W ramach ograniczeń projektowych należy wspomnieć, że projekt miał określony harmonogram (zgodnie z Raportem Studium Wykonalności). Wymusiło to priorytetyzację funkcji – w pierwszej kolejności zrealizowano rdzeń funkcjonalny (portfel, kalkulator, pobieranie danych), kosztem pominięcia mniej istotnych dodatków (jak wspomniany import/eksport czy rozbudowane role użytkowników). To ograniczenie czasowe zostało uwzględnione w podejmowaniu decyzji projektowych.

Ograniczenia projektowe definiują **ramy**, w których powstała aplikacja. Świadomie wybrano platformę Windows i usługi chmurowe, rezygnując z pewnych uniwersalnych rozwiązań (np. aplikacja webowa czy mobilna). Dzięki temu możliwe było skupienie się na kluczowych funkcjach i dostarczenie działającego produktu w zakładanym terminie.

3.7. Atrybuty systemu

Atrybuty systemu to cechy jakościowe i pozafunkcjonalne, które aplikacja **„Dycha”** powinna spełniać, takie jak niezawodność, bezpieczeństwo, łatwość modyfikacji czy przenośność.

- **Niezawodność:** System powinien być na tyle niezawodny, aby użytkownicy mogli mu zaufać w kontekście przechowywania swoich danych inwestycyjnych i wykonywania analiz. Niezawodność oznacza tu m.in. brak niespodziewanych błędów aplikacji (awarii)

podczas typowej pracy. Aplikacja przeszła testy podstawowych scenariuszy i nie powinna się zawieszać ani zamykać bez kontroli użytkownika. Dodatkowo, integracja z Firebase gwarantuje niezawodność przechowywania danych (wysoka dostępność serwerów w chmurze) – ryzyko utraty danych jest minimalne. Ewentualne błędy w komunikacji (np. brak internetu) są obsługiwane przez system komunikatów, co pozwala na **graceful degradation** zamiast nagłego zakończenia działania.

- **Availability (Dostępność):** Jako aplikacja kliencka, **„Dycha”** jest dostępna dla użytkownika zawsze wtedy, gdy zainstaluje ją na swoim komputerze – nie ma okien serwisowych czy przerw, bo logika działa lokalnie. Jednak dostępność funkcji związanych z danymi zależy od dostępności usług zewnętrznych: Finnhub API i Firebase. Te usługi mają wysoką dostępność, choć nie możemy wykluczyć krótkotrwałych problemów (np. konserwacje serwerów). W praktyce jednak, dla użytkownika, aplikacja jest dostępna do użycia **24/7**, wymagane jest jedynie połączenie sieciowe.
- **Bezpieczeństwo:** Aplikacja została zaprojektowana z uwzględnieniem podstaw bezpieczeństwa:
    - **Uwierzytelnianie** – dostęp do danych wymaga zalogowania (przez Firebase Auth). Każdy użytkownik ma własne dane i nie może odczytać cudzych.
    - **Autoryzacja** – reguły bazy danych zapobiegają nieuprawnionemu dostępowi, co było omówione w sekcji 3.5.2.
    - **Komunikacja szyfrowana** – wszystkie transmisje danych odbywają się przez HTTPS, co chroni przed podsłuchem.
    - **Przechowywanie haseł** – *Uwaga:* Aplikacja sama nie hashuje haseł, lecz polega na Firebase Auth, który zajmuje się bezpieczeństwem danych logowania (w bazie Firebase hasło nie jest przechowywane w postaci czystego tekstu, tylko w formie bezpiecznej, co jest integralną częścią usługi Firebase). Z punktu widzenia specyfikacji, nie implementujemy własnego mechanizmu hashującego, a więc nie wspominamy o nim w wymaganiach – bazujemy na zaufanej usłudze zewnętrznej.
    - **Dane wrażliwe** – Aplikacja operuje na danych finansowych użytkownika (co można uznać za wrażliwe informacje osobiste). W związku z tym zadbano, by nie były one nigdzie publicznie dostępne. Nie są też przechowywane lokalnie na dysku w niezaszyfrowanej postaci – wszystko trzymane jest w chmurze pod ochroną konta. Jedynym potencjalnym słabym punktem jest sam komputer użytkownika (jeśli pozostanie zalogowany i ktoś go użyje, zobaczy dane; lub jeśli ktoś pozna jego hasło). Tych aspektów technicznie aplikacja nie kontroluje – to kwestia odpowiedzialności użytkownika. Niemniej w przyszłości można rozważyć dodanie np. automatycznego wylogowania po okresie bezczynności dla zwiększenia bezpieczeństwa.
- **Przenośność (portability):** Ze względu na przyjęte założenia projektowe, aplikacja nie jest przenośna między różnymi systemami operacyjnymi – działa tylko na Windows (jak wspomniano w ograniczeniach projektowych). Nie jest to system rozproszony w sensie architektury (całość logiki aplikacyjnej jest w kliencie), więc nie ma komponentów, które można by rozdzielić na różne maszyny. Przenoszalność kodu źródłowego na inne

platformy (np. stworzenie wersji na macOS) wymagałaby istotnych zmian (zamiana biblioteki interfejsu GUI, potencjalnie inne mechanizmy bazy). Jednak wykorzystanie języka C# i .NET daje pewną przenośność w ramach platformy Windows (aplikacja działa na różnych wersjach Windows).

- **Łatwość utrzymania (maintainability):** Kod aplikacji został napisany w C# zgodnie z dobrymi praktykami inżynierii oprogramowania (podział na warstwy: interfejs, logika biznesowa, dostęp do danych). Wszelkie stałe (np. klucz API, identyfikatory elementów interfejsu) są trzymane w jednym miejscu, dzięki czemu ewentualne modyfikacje (np. zmiana dostawcy API czy dostosowanie do nowej wersji Firebase SDK) mogą być wykonane stosunkowo szybko. Projekt jest stosunkowo niewielki i zrozumiały dla programistów, co oznacza, że wprowadzanie nowych funkcji lub poprawek nie powinno nastręczać dużych trudności. Dokumentacja (w tym niniejszy SRS i wspomniany raport wykonalności) dodatkowo ułatwia zrozumienie założeń podczas utrzymania.
- **Wydajność i skalowalność:** Omówione w sekcji 3.4 wymogi wydajnościowe wskazują, że aplikacja spełnia swoje zadania wydajnościowo dla zakładanego obciążenia. Jeśli liczba użytkowników wzrośnie znacząco, może pojawić się konieczność np. implementacji mechanizmu cache'ującego dane rynkowe, aby nie przekraczać limitów API, lub optymalizacji pewnych operacji bazodanowych. Ogólnie jednak architektura (single-user client + cloud) jest skalowalna w ten sposób, że dodanie kolejnych użytkowników nie obciąża centralnego serwera aplikacji (bo takiego brak) – ciężar spoczywa na usługach chmurowych, które są skalowalne.
- **Wrażenia użytkownika (UX):** Poza czysto funkcjonalną użytecznością, istotnym atrybutem jest **satysfakcja użytkownika** z korzystania z systemu. Aplikacja ma za zadanie nie tylko działać poprawnie, ale też sprawiać, że użytkownik chętnie będzie z niej korzystał do analiz. Dzięki nastawieniu na prostotę i szybkie działanie, oczekuje się, że użytkownicy docenią brak zbędnych formalności (np. natychmiastowy dostęp do danych po zalogowaniu, brak rozwlekłych konfiguracji).

Podsumowując, atrybuty systemu **„Dycha"** zostały tak dobrane, by stworzyć narzędzie **wiarygodne, bezpieczne i przyjazne** dla użytkownika. Kluczowe cechy to bezpieczeństwo przechowywania danych (dzięki Firebase) i prostota obsługi (intuicyjny interfejs), przy akceptacji ograniczeń takich jak konieczność internetu czy zamknięcie na jedną platformę systemową.

3.8. Informacje dodatkowe (załączniki)

W sekcji tej zawarto dodatkowe informacje uzupełniające specyfikację, takie jak zrzuty ekranu interfejsu (zaprezentowane już powyżej przy opisie interfejsu użytkownika) oraz ewentualne diagramy czy tabele pomocne w zrozumieniu działania systemu.

Załączone zrzuty ekranu: W celu lepszego zilustrowania funkcjonalności aplikacji, do dokumentu dołączono następujące przykładowe ekrany (omówione w sekcji 3.1):

- Rysunek 1: Strona główna z wykresem instrumentu i panelem zakupu (ekran dodawania pozycji).
- Rysunek 2: Ekran Portfela z listą posiadanych pozycji (tabela portfela z podsumowaniem).

- Rysunek 3: Ekran Kalkulatora inwestycyjnego z wprowadzonymi danymi i wyświetlonymi wynikami analizy.
- Rysunek 4: Ekran rejestracji nowego użytkownika.
- Rysunek 5: Ekran logowania istniejącego użytkownika.
- Rysunek 6: Strona główna przed i po logowaniu/rejestracji.

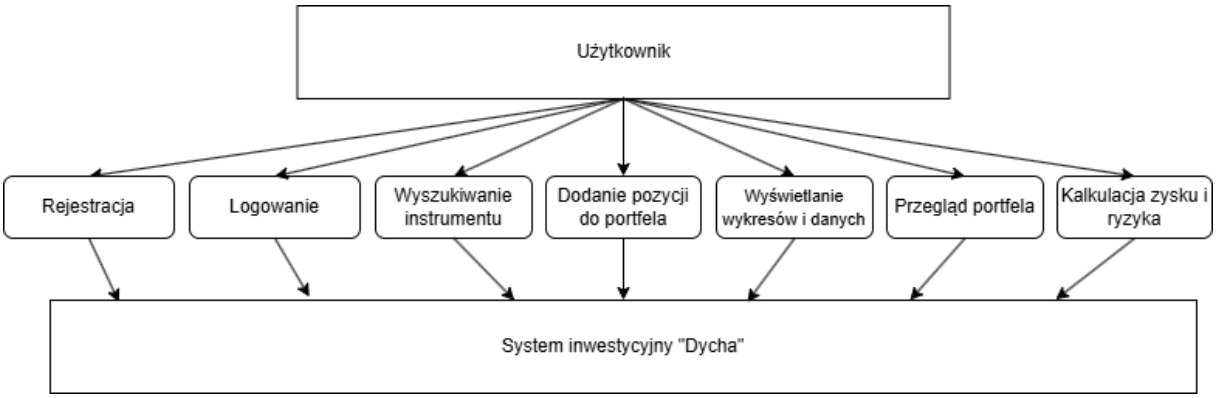
Te ilustracje mają charakter poglądowy i pochodzą z wersji prototypowej aplikacji. Rzeczywisty wygląd może ulec drobnym zmianom w finalnej wersji (np. kosmetyczne poprawki interfejsu), jednak przedstawione elementy funkcjonalne odpowiadają opisanym wymaganiom.

Diagram przypadków użycia: Poniżej przedstawiono opcjonalnie diagram przypadków użycia obrazujący interakcje użytkownika z systemem **„Dycha”**.

(Wyliczenie głównych przypadków użycia):

- Rejestracja użytkownika
- Logowanie
- Wyszukanie instrumentu
- Dodanie pozycji do portfela (zakup)
- Przegląd portfela
- Analiza inwestycji (kalkulator)
- Wylogowanie

Każdy z powyższych przypadków został opisany słownie w specyfikacji wymagań funkcjonalnych (rozdz. 3.2) lub w sekcji 4 poniżej. Diagram służy jako streszczenie tych interakcji, pokazując, że wszystkie akcje wykonuje aktor *Użytkownik*, który wchodzi w interakcje z systemem w ramach wymienionych funkcjonalności.

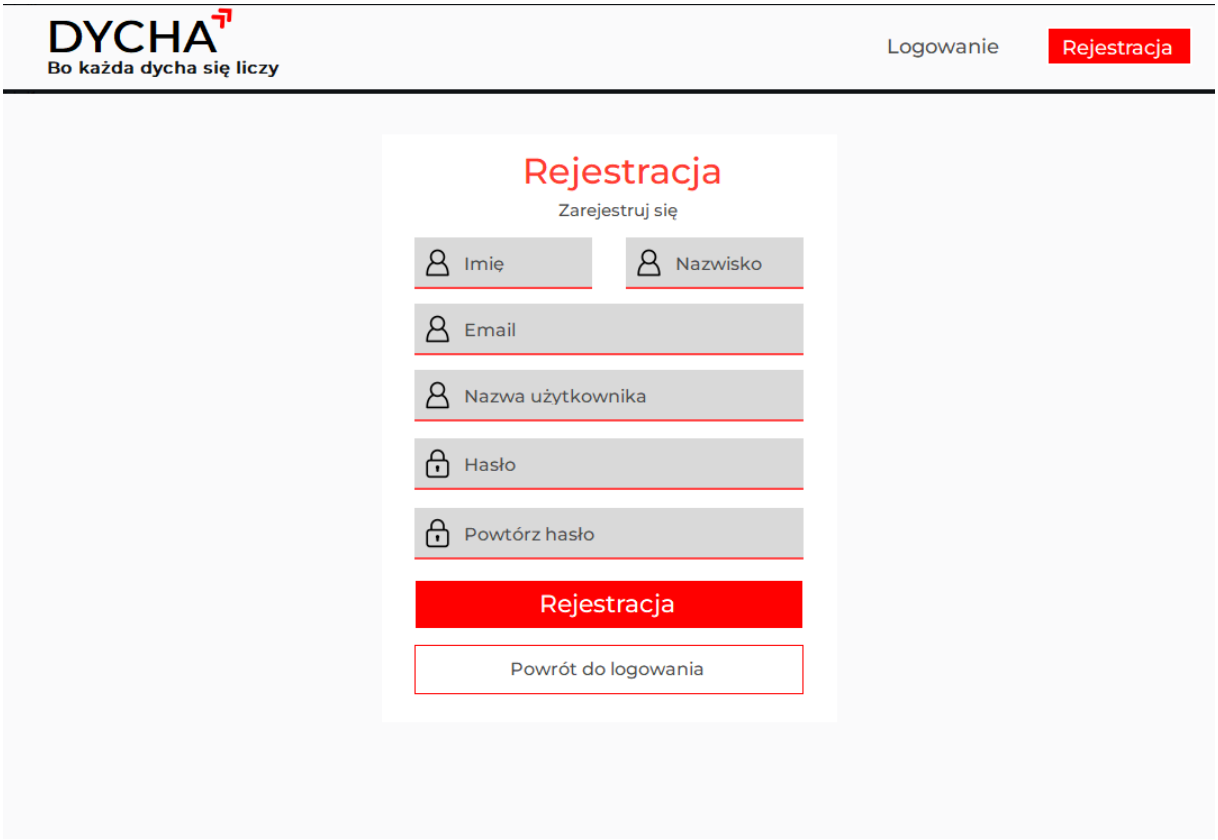


W dalszej części dokumentu znajduje się szczegółowy opis **przypadków użycia** (scenariuszy działania systemu z perspektywy użytkownika), co dopełnia całości specyfikacji, łącząc wymagania z konkretnymi sekwencjami operacji.

4. Przypadki użycia

Poniżej opisano główne przypadki użycia systemu **„Dycha”**, ilustrując krok po kroku, jak użytkownik realizuje poszczególne zadania za pomocą aplikacji. Każdy przypadek użycia zawiera określenie aktorów, scenariusz główny oraz ewentualne alternatywy.

Przypadek użycia 1: Rejestracja nowego użytkownika



*Rysunek 4. Ekran **Rejestracji***

Aktor główny: Użytkownik (niezalogowany)
Opis: Użytkownik tworzy nowe konto, aby uzyskać dostęp do aplikacji.
Warunki początkowe: Aplikacja wyświetla ekran powitalny z opcją rejestracji i logowania. Użytkownik nie posiada jeszcze konta w systemie.
Scenariusz główny:

1. Użytkownik wybiera opcję “Rejestracja” na ekranie startowym.
2. System wyświetla formularz rejestracyjny z polami do wprowadzenia adresu e-mail oraz hasła.
3. Użytkownik wprowadza wymagany adres e-mail i hasło (oraz ewentualnie powtórzenie hasła, jeśli jest wymagane dla walidacji).
4. Użytkownik zatwierdza formularz (np. przyciskiem “Zarejestruj”).
5. System sprawdza poprawność danych:
    - Czy adres e-mail ma prawidłowy format.
    - Czy hasło spełnia minimalne wymagania (np. długość min. 6 znaków).

6. System wysyła dane do usługi Firebase Authentication celem utworzenia konta.
7. Firebase tworzy konto (o unikalnym identyfikatorze UID) lub zwraca błąd jeśli użytkownik o podanym e-mail już istnieje.
8. Jeśli rejestracja się powiodła: **System wyświetla komunikat potwierdzający** (“Konto utworzone pomyślnie”) oraz automatycznie loguje użytkownika lub przekierowuje go do ekranu logowania.
9. Użytkownik (jeśli nie został automatycznie zalogowany) może teraz zalogować się używając właśnie utworzonych danych.

Alternatywny przepływ 1a (Email zajęty): Jeśli na kroku 7 Firebase zwróci informację, że podany email jest już zarejestrowany (konto istnieje):

- 7a. System informuje użytkownika o błędzie: “Konto o podanym adresie e-mail już istnieje.”
- 7b. Użytkownik ma możliwość powrotu do formularza i podania innego adresu lub przejścia do ekranu logowania jeśli przypomniał sobie, że zakładał konto wcześniej.

Alternatywny przepływ 5a (Niepoprawne dane lokalnie): Jeśli na kroku 5 walidacja lokalna wykaże błąd (np. hasło za krótkie):

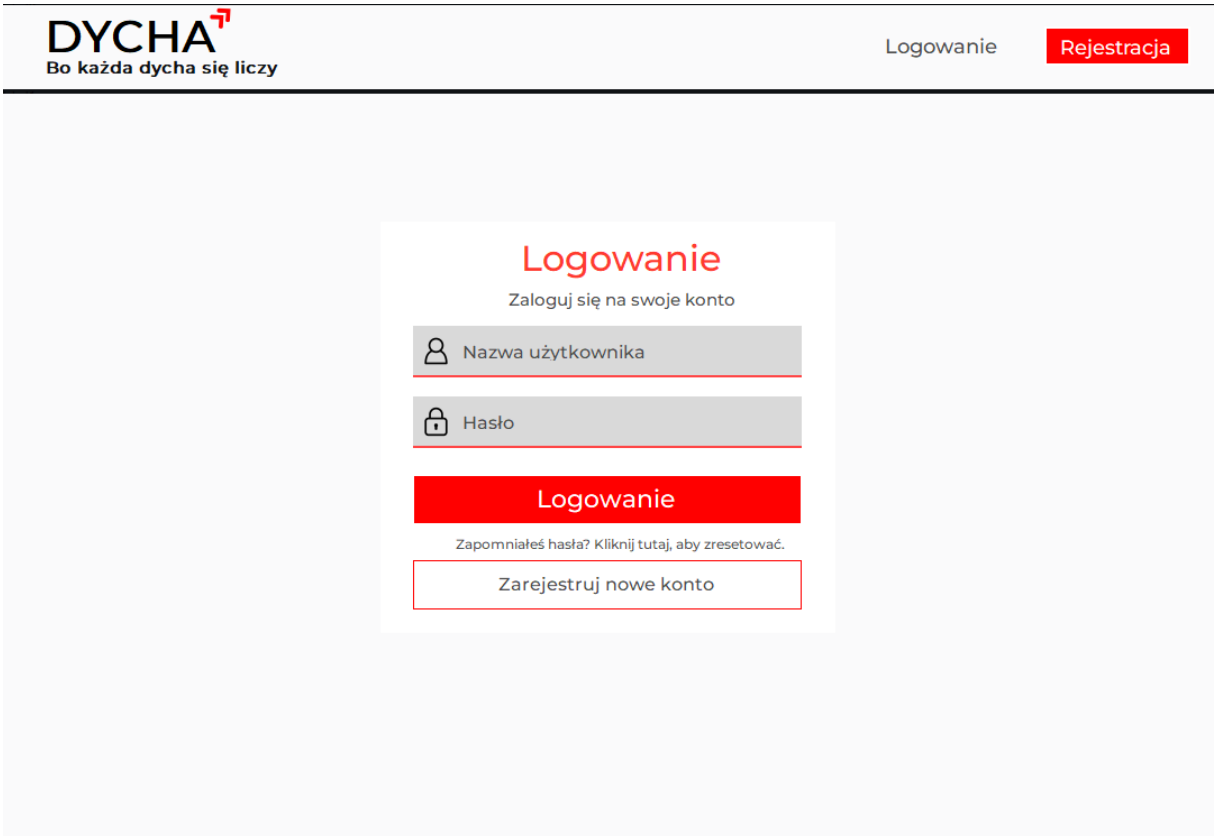
- 5a. System wyświetla odpowiedni komunikat obok błędnego pola (np. pod polem hasła “Hasło musi mieć co najmniej 6 znaków”).
- 5b. Użytkownik poprawia dane i wraca do kroku 4.

Wyjątek łączności: Jeśli podczas próby rejestracji (krok 6-7) wystąpi problem z połączeniem internetowym lub serwerem:

- System wyświetla komunikat “Rejestracja nie powiodła się z powodu problemów technicznych. Sprawdź połączenie z internetem i spróbuj ponownie później.”
- Użytkownik może ponowić próbę później.

Warunki końcowe: Nowe konto użytkownika jest utworzone w systemie (w bazie Firebase). Użytkownik może kontynuować do logowania i dalszej pracy z aplikacją.

Przypadek użycia 2: Logowanie użytkownika



*Rysunek 5. Ekran **Logowania***

Aktor główny: Użytkownik (posiadający konto)

Opis: Użytkownik uwierzytelnia się w systemie, aby uzyskać dostęp do swoich danych i funkcji aplikacji.

Warunki początkowe: Użytkownik znajduje się na ekranie logowania (np. uruchomił aplikację, która domyślnie pokazuje panel logowania; ewentualnie tuż po rejestracji został przekierowany do logowania). Użytkownik ma już utworzone konto (email + hasło).

Scenariusz główny:

1. Użytkownik wprowadza swój adres e-mail oraz hasło w odpowiednie pola logowania.
2. Użytkownik naciska przycisk **Zaloguj**.
3. System przekazuje dane logowania do Firebase Authentication w celu weryfikacji.
4. Firebase sprawdza, czy istnieje konto o podanym e-mail i czy hasło jest prawidłowe.
5. Jeśli dane są poprawne: Firebase zwraca token uwierzytelnienia, a system lokalny uznaje użytkownika za zalogowanego.
6. **System przekierowuje użytkownika do głównego ekranu aplikacji** (Strona główna z menu, początkowo może być pusta bez wybranego instrumentu lub z domyślnym widokiem). Użytkownik teraz ma dostęp do funkcji portfela, kalkulatora itp.
7. System może gdzieś w interfejsie (np. w nagłówku) wskazać, że użytkownik jest zalogowany (np. wyświetlić jego adres e-mail lub nazwę).

Alternatywny przepływ 5a (Błędne dane): Jeśli Firebase zwraca błąd uwierzytelnienia (np. niepoprawne hasło lub konto nie istnieje):

- 5a. System pozostaje na ekranie logowania i wyświetla komunikat “Nieprawidłowy email lub hasło. Spróbuj ponownie.”
- 5b. Użytkownik ma możliwość ponownego wpisania danych (powrót do kroku 1) lub ewentualnie wybrania opcji przypomnienia hasła (jeśli taka istnieje – w v1.0 może nie być zaimplementowane, więc wtedy tylko ponowna próba).

Wyjątek łączności: Jeśli na kroku 3-4 nie można skontaktować się z serwerem (brak internetu):

- System wyświetla komunikat “Brak połączenia z serwerem – nie można się zalogować.”
- Użytkownik może spróbować ponownie później; pozostaje na ekranie logowania.

Warunki końcowe: Użytkownik jest zalogowany do systemu, a interfejs aplikacji przełącza się na tryb pracy (widoczne funkcje aplikacji). Sesja użytkownika pozostaje aktywna do czasu wylogowania lub zamknięcia aplikacji.

Przypadek użycia 3: Wyszukanie i wyświetlenie instrumentu giełdowego

Rysunek 6. ***Strona główna***

Aktor główny: Użytkownik (zalogowany)

Opis: Użytkownik wyszukuje konkretny instrument finansowy (np. akcje spółki lub fundusz ETF) i przegląda jego aktualne notowania oraz wykres.

Warunki początkowe: Użytkownik jest zalogowany i znajduje się na ekranie **Strona główna** aplikacji. Połączenie z internetem jest dostępne.

Scenariusz główny:

1. Użytkownik klika pole wyszukiwania instrumentu (oznaczone np. lupą i tekstem "Wyszukaj..." w górnej części okna).
2. Użytkownik wpisuje nazwę spółki lub symbol (ticker) instrumentu, który go interesuje, np. "AAPL" dla Apple.
3. System (już podczas wpisywania lub po naciśnięciu Enter) może podpowiadać pasujące wyniki – jeśli zaimplementowano funkcję podpowiedzi. Jeśli nie, to:
4. Użytkownik zatwierdza wpisaną frazę (Enter lub przycisk lupy).
5. System nawiązuje połączenie z Finnhub API, wysyłając zapytanie o dane instrumentu o podanej nazwie/symbolu. (Może to być zapytanie typu *quote* dla bieżącej ceny oraz *lookup/symbol* dla rozpoznania nazwy – zależnie od API).
6. API zwraca dane: jeśli instrument został znaleziony, otrzymujemy np. aktualną cenę, zmianę, i ewentualnie identyfikator do pobrania wykresu. Jeśli instrumentu nie znaleziono, API zwróci pusty wynik lub błąd.
7. System sprawdza odpowiedź:
    - Jeżeli instrument znaleziono: **System aktualizuje interfejs** – wyświetla nazwę instrumentu, jego aktualny kurs, zmianę (w % i absolutną), oraz ładuje wykres tego instrumentu w kontrolce WebBrowser. Wykres może potrzebować kolejnego zapytania, np. do Google Finance z parametrem tickeru – aplikacja generuje odpowiedni URL (np. https://www.google.com/finance/quote/AAPL:NASDAQ) i ustawia go w kontrolce przeglądarki. Użytkownik widzi załadowane dane.
    - Jeżeli instrument nie został znaleziony: System wyświetla komunikat "Nie znaleziono instrumentu o podanej nazwie." (być może wskazówkę "sprawdź poprawność symbolu"). Pozostaje stary widok lub pusty wykres.
8. Użytkownik obserwuje wyświetlone informacje. Może np. zauważyć, że **High**, **Low** dzisiejsze są pokazane, poprzednie zamknięcie itd., a wykres intraday jest narysowany.
9. Użytkownik może w tym momencie:
    - zdecydować się na dodanie instrumentu do portfela (przechodzi do przypadku użycia 4),
    - lub wpisać inny instrument (wracając do kroku 2 i powtarzając proces dla nowej frazy).

Alternatywny przepływ (podpowiedzi 3a): Jeśli aplikacja implementuje podpowiedzi przy wyszukiwaniu:

- 3a. System podczas wpisywania tekstu wywołuje API wyszukujące symbole po fragmencie nazwy (np. Finnhub ma endpoint symbol lookup).
- 3b. Użytkownik widzi listę pasujących symboli/nazw i może wybrać odpowiedni z listy zamiast wpisywać całość ręcznie.
- 3c. Po wybraniu z podpowiedzi, system kontynuuje od kroku 5 dla wybranego symbolu.

Wyjątek (limit API lub brak netu): Jeśli zapytanie do API nie powiedzie się (np. brak odpowiedzi, przekroczony limit):

- System wyświetla komunikat “Błąd pobierania danych, spróbuj ponownie za chwilę.” i nie zmienia stanu wyświetlanych danych.
- Użytkownik może poczekać i ponowić próbę.

Warunki końcowe: Dane wybranego instrumentu są wyświetlone użytkownikowi (o ile zostały znalezione). Aplikacja pozostaje na ekranie głównym z załadowanym instrumentem.

Przypadek użycia 4: Dodanie transakcji (zakup instrumentu) do portfela
Aktor główny: Użytkownik (zalogowany)
Opis: Użytkownik dodaje wyświetlony instrument jako pozycję w swoim portfelu inwestycyjnym, określając ile jednostek kupuje.
Warunki początkowe: Użytkownik ma na ekranie głównym wyświetlony instrument finansowy (przebieg przypadku użycia 3 doprowadził do wyświetlenia danych instrumentu). Użytkownik zdecydował, że chce “kupić” (dodać do portfela) ten instrument. Zakładamy, że instrument ma aktualnie pobraną cenę.
Scenariusz główny:

1. Użytkownik znajduje na ekranie panel zakupu (z prawej strony, jak na Rysunku 1).
2. Użytkownik klika w pole **Wolumen** i wpisuje liczbę, odpowiadającą liczbie akcji/jednostek, które chce dodać. Np. wpisuje “10” (co oznacza 10 sztuk).
3. System może w tle sprawdzić poprawność wartości (wolumen powinien być dodatnią liczbą całkowitą). Przyjmijmy, że wpis jest prawidłowy.
4. Użytkownik naciska przycisk **Kup** (dodaj do portfela).
5. System pobiera (być może już posiada z kroku 3 poprzedniego UC) aktualną cenę instrumentu. Może dla pewności wykonać szybkie odświeżenie ceny przez API lub użyć tej wyświetlonej. (W wersji studenckiej zapewne używa właśnie wyświetlanej ceny, aby nie komplikować).
6. System tworzy nową strukturę danych dla pozycji i zapisuje ją w bazie:
    - Przygotowuje obiekt zawierający: symbol instrumentu, wolumen=10, cenaZakupu = np. 150 USD (za sztukę, jeśli tyle wynosi aktualny kurs), datę transakcji = teraz.
    - Wysyła żądanie do Firebase (bazy) dodające dokument w kolekcji portfela bieżącego użytkownika.
7. Firebase zapisuje dane i zwraca potwierdzenie (lub identyfikator dokumentu).
8. System lokalny po potwierdzeniu zapisu:
    - Wyświetla komunikat potwierdzający dodanie (“Dodano do portfela” albo od razu odświeża widok portfela).
    - Ewentualnie automatycznie przełącza się na ekran **Portfel**, aby użytkownik od razu zobaczył efekt (to zależy od projektu UX; jeśli nie, użytkownik sam kliknie “Portfel” – patrz UC5).

9. Jeśli przełączono automatycznie na Portfel: System pobiera aktualny stan portfela z bazy (w tym nowo dodaną pozycję) i wyświetla tabelę. Nowa pozycja powinna być widoczna (np. jako ostatnia na liście).

Alternatywny przepływ 3a (niepoprawny wolumen): Jeśli użytkownik wpisał błędną wartość wolumenu (np. “0” lub “-5” lub “abc”):

- 3a. System uniemożliwia kliknięcie “Kup” (przycisk nieaktywny) lub wyświetla komunikat “Wprowadź poprawną liczbę większą od zera.”
- 3b. Użytkownik poprawia wartość i wraca do kroku 4.

Alternatywny przepływ 5a (brak ceny): Jeśli z jakiegoś powodu system nie ma aktualnej ceny (np. wystąpił błąd przed wyświetleniem):

- 5a. System próbuje ponownie pobrać cenę z API. Jeśli się uda, kontynuuje normalnie.
- 5b. Jeśli się nie uda, informuje użytkownika: “Nie udało się pobrać ceny – operacja przerwana.” (Użytkownik może spróbować ponownie później).

Wyjątek łączności/przechowywania: Jeśli zapis do bazy (krok 6-7) nie powiedzie się (np. utrata internetu po pobraniu ceny, zanim zapisano):

- System wyświetla błąd: “Nie udało się zapisać pozycji. Sprawdź połączenie i spróbuj ponownie.”
- (Opcjonalnie: system może przechować tę pozycję tymczasowo i spróbować zapisać później automatycznie, ale w v1.0 raczej nie ma takiej kolejki – użytkownik musiałby kliknąć ponownie).

Warunki końcowe: Nowa pozycja zostaje dodana do portfela użytkownika (zapisana w bazie). Użytkownik może ją zobaczyć w widoku portfela i od tego momentu będzie ona uwzględniana w obliczeniach (np. sumaryczna wartość, zyski/straty).

Przypadek użycia 5: Przegląd portfela
Aktor główny: Użytkownik (zalogowany)
Opis: Użytkownik przegląda listę wszystkich swoich inwestycji dodanych do portfela i analizuje ich bieżące wyniki.
Warunki początkowe: Użytkownik jest zalogowany i ma już jakieś pozycje w portfelu (dodane wcześniej). Znajduje się w dowolnym miejscu aplikacji (np. na stronie głównej po dodaniu pozycji).
Scenariusz główny:

1. Użytkownik wybiera zakładkę **Portfel** z menu głównego aplikacji.
2. System przełącza widok na ekran Portfela.
3. System pobiera z bazy Firebase listę pozycji portfela należących do zalogowanego użytkownika.
4. Dla każdej pozycji, system może równolegle pobrać z API bieżącą cenę instrumentu (chyba że takie dane są cache’owane z poprzednich operacji). Zakładamy, że trzeba pobrać:

    - Dla każdej pozycji (np. 3 pozycje): wysyła żądanie do API o aktualnościach (kontynuacja przypadku użycia 5)

5. Dla każdej pozycji, system może równolegle pobrać z API bieżącą cenę instrumentu (chyba że dane cenowe są już w pamięci podręcznej z wcześniejszych operacji). Zakładamy, że system wykonuje serię zapytań do Finnhub API – jedno dla każdego unikalnego symbolu w portfelu – aby zaktualizować informacje o tych pozycjach.
6. Po otrzymaniu odpowiedzi (lub w trakcie, jeśli realizowane asynchronicznie) **system wyświetla tabelę portfela**: każdy wiersz odpowiada jednej pozycji inwestycyjnej. Wypełniane są kolumny: *Pozycja (symbol)*, *Wolumen*, *Wartość rynkowa* (wolumen × bieżąca cena), *Cena otwarcia* (cena zakupu), *Zysk netto* (różnica między bieżącą wartością a wartością zakupu) oraz *Zysk netto %* (stosunek zysku netto do łącznego kosztu zakupu, w %). Na dole ekranu system wyświetla **podsumowanie**: łączna wartość konta (suma wartości rynkowych wszystkich pozycji) oraz łączny zysk/strata całego portfela (sumarycznie).
7. Użytkownik analizuje informacje na ekranie. Może przewijać listę, jeśli portfel jest długi. Widzimy np., że jedna pozycja ma zysk +11,76%, inna stratę -1,11% (jak na Rysunku 2).
8. Użytkownik może zdecydować się na przeprowadzenie dalszych akcji z tego poziomu, np. wybrać pozycję i kliknąć **Wyświetl** (jeśli dostępny taki przycisk), co najprawdopodobniej spowoduje przejście do ekranu głównego z wykresem tego instrumentu (to by był dodatkowy, pomniejszy przypadek użycia: *Podgląd szczegółów pozycji*). Jeśli takiej funkcji nie ma, użytkownik po prostu wykorzystuje te informacje do własnej oceny.
9. Użytkownik może pozostać w widoku portfela lub przejść do innej sekcji (np. kalkulatora, by ocenić potencjał dalszych inwestycji).

Alternatywny przepływ (brak pozycji 3a): Jeśli użytkownik nie ma jeszcze żadnych pozycji w portfelu (np. zaraz po założeniu konta, nic nie dodał):

- 3a. System wykrywa pustą listę i wyświetla komunikat typu “Twój portfel jest pusty. Dodaj pierwszą pozycję na stronie głównej.” zamiast tabeli (lub tabelę bez wierszy).
- Użytkownik w takiej sytuacji może tylko przejść do strony głównej i dodać pozycje (przypadek użycia 4).

Alternatywny przepływ 4a (nieudane pobranie cen): Jeśli w kroku 4 któreś zapytanie do API zawiedzie (np. czasowo):

- System może pominąć aktualizację tej jednej pozycji i oznaczyć ją np. specjalnym symbolem lub pozostawić starą wartość.
- Alternatywnie, może wstrzymać odświeżanie całego portfela i wyświetlić ostrzeżenie “Nie udało się zaktualizować cen wszystkich pozycji. Dane mogą być nieaktualne.”.

Warunki końcowe: Użytkownik ma przed sobą aktualny obraz swojego portfela inwestycyjnego. Wszystkie dane pozycji są odświeżone (na tyle, na ile pozwoliło API) i może on ocenić wyniki swoich inwestycji. Portfel nie jest w tym momencie modyfikowany, tylko odczytywany.

Przypadek użycia 6: Analiza inwestycji za pomocą kalkulatora
Aktor główny: Użytkownik (zalogowany)
Opis: Użytkownik korzysta z narzędzia kalkulatora, by oszacować potencjalny zwrot z inwestycji i ryzyko dla wybranego instrumentu.
Warunki początkowe: Użytkownik znajduje się na ekranie **Kalkulator**. Załóżmy, że wcześniej wybrał instrument (np. korzystając z wyszukiwarki na stronie głównej lub poprzez wybranie pozycji z portfela). W nagłówku kalkulatora widoczny jest symbol/nazwa instrumentu, którego dotyczą obliczenia (jak na Rys. 3 – AADR). Jeśli aplikacja wymaga najpierw wskazania instrumentu do kalkulacji, użytkownik powinien to zrobić (np. może być pole wyboru instrumentu na ekranie kalkulatora lub kalkulator zawsze działa na ostatnio oglądanym instrumencie).
Scenariusz główny:

1. Użytkownik wprowadza w polu **Kwota inwestycji** planowaną kwotę, np. 2000 (domyślnie traktowaną jako 2000 USD).
2. Użytkownik wprowadza w polu **Ilość dni inwestycji** okres inwestycji, np. 365 dni (co odpowiada mniej więcej jednemu rokowi).
3. System może weryfikować wpisane dane na bieżąco (np. czy to są liczby dodatnie). Zakładamy, że wpis jest poprawny.
4. Użytkownik naciska przycisk **Sprawdź** (rozpoczynający kalkulację).
5. System pobiera z Finnhub API (lub innego źródła) **dane historyczne** dla danego instrumentu – konkretne zakresy zależą od implementacji. Np. jeśli użytkownik podał 365 dni, system może pobrać serię dziennych cen z ostatniego roku dla tego instrumentu (lub przynajmniej wartość sprzed 365 dni i obecną, by obliczyć zmianę). Alternatywnie mógłby korzystać z rocznej stopy zwrotu już dostępnej gdzieś, ale prawdopodobnie wylicza sam.
6. Po wprowadzeniu danych przez użytkownika, aplikacja wykonuje obliczenia według zaimplementowanego algorytmu:
    - **Szacowany dzienny przyrost ceny** obliczany jest jako średnia różnica pomiędzy wartością maksymalną i minimalną danego instrumentu w okresie (średni zakres dzienny):
      dailyChange = (highPrice - lowPrice) / 2.
    - **Przewidywana przyszła cena instrumentu** obliczana jest poprzez dodanie skumulowanej dziennej zmiany do ceny bieżącej:
      futurePrice = currentPrice + (dailyChange * numberOfDays).
    - **Oczekiwany zysk w dolarach** wyliczany jest na podstawie różnicy między ceną przewidywaną a bieżącą, przeliczoną względem wniesionej kwoty:
      expectedReturn = ((futurePrice - currentPrice) / currentPrice) * investmentAmount.
    - **Procentowy zwrot z inwestycji** obliczany jest według wzoru:
      riskPercentage = expectedReturn / investmentAmount * 100%.
    - **Ocena poziomu ryzyka** dokonywana jest na podstawie wysokości przewidywanego zwrotu procentowego:

- Jeśli zwrot < 10% → oznaczony jako **„niskie ryzyko”** (zielony kolor).
- Jeśli zwrot mieści się w przedziale 10–20% → oznaczony jako **„średnie ryzyko”** (pomarańczowy kolor).
- Jeśli zwrot > 20% → oznaczony jako **„wysokie ryzyko”** (czerwony kolor).

7. System wyświetla wyniki na ekranie kalkulatora:
    - **Przewidywany zwrot w %:** 162,56% (zielony lub inny wyróżnik, bo to wartość dodatnia).
    - **Przewidywany zysk:** 3251,24 $ (też wyróżniony kolorem pozytywnym).
    - **Przewidywane ryzyko:** “Wysokie ryzyko” (najpewniej czerwony lub ostrzegawczy kolor, by zaznaczyć że to niepewna inwestycja).
      Te wartości odpowiadają polom pokazanym na Rysunku 3.
8. Użytkownik analizuje te wyświetlone informacje. Na ich podstawie może podjąć decyzję – np. stwierdzić, że potencjał zysku jest duży, ale ryzyko za wysokie, więc może zmniejszy kwotę inwestycji lub wybierze inny instrument do analizy. Użytkownik ma możliwość zmiany parametrów i ponownego kliknięcia **Sprawdź** (wraca do kroku 1 z nowymi danymi, np. inna kwota albo po zmianie instrumentu).

Alternatywny przepływ 3a (nieprawidłowe dane): Jeśli użytkownik wpisał nieprawidłowe dane wejściowe:

- 3a. System wykrywa błąd (np. pole kwoty puste lub zero, pole dni = 0 lub tekst).
- 3b. System wyświetla komunikat przy polu (np. “Wprowadź dodatnią liczbę”).
- 3c. Użytkownik poprawia dane i ponownie naciska **Sprawdź** (wracamy do kroku 5).

Alternatywny przepływ 5a (brak danych do analizy): Jeśli API nie zwróci wymaganych danych historycznych (np. instrument notowany krócej niż wymagany okres, lub limit zapytań wyczerpany):

- 5a. System wyświetla komunikat “Brak wystarczających danych do przeprowadzenia analizy dla wybranego okresu.” lub “Analiza chwilowo niedostępna, spróbuj ponownie później.” w zależności od przyczyny.
- 5b. Użytkownik może zmniejszyć zakres dni (jeśli problemem był zbyt długi okres) lub po prostu spróbować ponownie później (jeśli problemem był limit API).

Warunki końcowe: Użytkownik otrzymuje na ekranie kalkulatora wyniki symulacji dla wybranego instrumentu. Żadne dane nie są trwale zapisywane w systemie w związku z tym obliczeniem – jest to tylko funkcja analityczna. Użytkownik może wykorzystać wyniki do podjęcia decyzji (np. zdecydować, czy kupić ten instrument – co go może skłonić do realizacji przypadku użycia 4 ponownie).

Przypadek użycia 7: Wylogowanie
Aktor główny: Użytkownik (zalogowany)
Opis: Użytkownik kończy pracę z aplikacją i wylogowuje się, aby zabezpieczyć dostęp do swojego konta.

Warunki początkowe: Użytkownik jest zalogowany i znajduje się w dowolnym miejscu aplikacji (Strona główna/Portfel/Kalkulator).
Scenariusz główny:

1. Użytkownik klika przycisk **Wyloguj** dostępny w interfejsie (np. w prawym górnym rogu okna aplikacji).
2. System natychmiast anuluje/zamyka bieżącą sesję użytkownika:
    - Z punktu widzenia aplikacji desktopowej: usuwa lokalnie zapisane poświadczenia (token Firebase) z pamięci, ewentualnie informuje usługę Firebase o wylogowaniu (choć to głównie kwestia strony klienckiej).
    - Wyczyści wszelkie dane tymczasowe związane z kontem (np. bufor portfela).
3. System przechodzi do ekranu logowania (bądź rejestracji/logowania) – takiego samego, jaki widzi użytkownik przy pierwszym uruchomieniu. Interfejs aplikacji już nie pokazuje danych wrażliwych użytkownika.
4. Użytkownik widzi ekran logowania, co potwierdza, że został wylogowany. Aplikacja pozostaje uruchomiona, umożliwiając zalogowanie się ponownie lub zamknięcie programu.

Alternatywny przepływ: (Brak istotnych alternatyw – wylogowanie to z definicji prosty proces.)

- Można ewentualnie rozważyć: jeśli użytkownik kliknie "X" zamykając okno aplikacji, to jest to inny przypadek – zamknięcie aplikacji. W takiej sytuacji aplikacja również kończy sesję (token po ponownym uruchomieniu nie będzie używany, chyba że zapamiętano użytkownika – takiej funkcji nie definiowaliśmy).

Warunki końcowe: Użytkownik zostaje poprawnie wylogowany. Dostęp do funkcji aplikacji wymaga ponownego zalogowania. Jego dane w chmurze pozostają nienaruszone, a aplikacja wraca do stanu oczekiwania na login. Operacja wylogowania zapewnia, że inni użytkownicy komputera nie uzyskają dostępu do konta inwestora bez podania hasła.

Podsumowanie przypadków użycia: Wszystkie powyższe scenariusze odzwierciedlają funkcjonalność zaimplementowaną w aplikacji **„Dycha"** w wersji 1.0. Specyfikacja wymagań oraz opisy użycia pokazują, że system umożliwia podstawowe operacje potrzebne inwestorowi: od założenia konta, przez zbieranie i przegląd danych inwestycyjnych, po wykonywanie analiz i bezpieczne zakończenie pracy. Ewentualne brakujące funkcjonalności (jak usuwanie pozycji czy import/eksport) zostały świadomie pominięte w tej wersji i mogą stanowić przedmiot przyszłych rozszerzeń. Dokument SRS wraz z raportem studium wykonalności zapewniają pełny obraz systemu, jego ograniczeń i sposobu działania, co umożliwia dalszy rozwój i utrzymanie aplikacji zgodnie z potrzebami użytkowników i interesariuszy.