

Politechnika Opolska

Wydział Elektrotechniki, Automatyki i Informatyki

Katedra Informatyki

PRACA DYPLOMOWA

Studia pierwszego stopnia stacjonarne

Kierunek studiów

Informatyka

**Aplikacja rekomendująca dobór perfum na podstawie preferencji
i składników, wykorzystująca mechanizm analizy danych cosine
similarity**

Promotor:
dr inż. Anna ZATWARNICKA

Pracę wykonała:
Oliwia ZAPART
nr albumu: 99517

Opole, luty 2023

Aplikacja rekomendująca dobór perfum na podstawie preferencji i składników, wykorzystująca mechanizm analizy danych cosine similarity

S t r e s z c z e n i e

W ramach pracy dyplomowej zaprojektowano, zrealizowano oraz przetestowano aplikację na urządzenia mobilne z systemem Android, która rekomenduje dobór perfum na podstawie preferencji użytkownika i składników perfum. System rekomendacji został skonstruowany oraz utworzony przy użyciu mechanizmów analizy danych cosine similarity, natomiast aplikację opracowano z wykorzystaniem języka Python oraz jego frameworka Kivy. Interfejs powstał na bazie heurystyk Nielsena, aby był przejrzysty i przyjazny dla użytkownika. Aplikacja umożliwia stworzenie profilu użytkownika, by wyszukiwać oraz przeglądać dostępne w bazie perfumy. Dodatkowo użytkownicy otrzymują opcję wypełnienia ankiety, która rozpozna ich preferencje zapachowe. Dzięki temu uzyskają dostęp do modułu rekomendacyjnego, oferującego listy perfum stworzone na podstawie wcześniej określonych preferencji.

An application recommending the selection of perfumes based on preferences and ingredients, using the cosine similarity data analysis mechanism

S u m m a r y

As part of the diploma thesis, an application for Android mobile devices was designed, implemented and tested, which recommends the selection of perfumes based on preferences and ingredients. The recommendation system was constructed and created using cosine similarity data analysis mechanisms, while the application was developed using Python and its Kivy framework. The interface was based on heuristics of Nielsen to be transparent and user-friendly. The application allows you to create an account to search and browse the perfumes available in the database. In addition, users are given the option of completing a survey that will recognize their fragrance preferences. Then they will have access to the recommendation module, offering a list of perfumes created on the basis of predetermined preferences.

SPIS TREŚCI

SPIS TREŚCI	3
SPIS RYSUNKÓW	5
SPIS LISTINGÓW	7
SPIS RÓWNAŃ	8
1. WSTĘP	9
1.1. CEL PRACY	9
1.2. ZAKRES REALIZOWANYCH PRAC	10
1.3. TREŚĆ PRACY	10
2. WPROWADZENIE	12
2.1. ANALIZA RYNKU APLIKACJI MOBILNYCH O PERFUMACH	12
2.1.1. <i>Aplikacja Parfumo</i>	14
2.1.2. <i>Aplikacja PERFUMIST</i>	15
2.2. ANALIZA SYSTEMÓW REKOMENDACJI	16
2.2.1. <i>Metoda oparta na treści</i>	17
2.2.2. <i>Metoda grupowego filtrowania</i>	17
2.2.3. <i>Metoda filtrowania hybrydowego</i>	18
2.3. PODSUMOWANIE	18
3. ANALIZA WYMAGAŃ	19
3.1. WYMAGANIA FUNKCYJALNE.....	19
3.2. WYMAGANIA NIEFUNKCYJALNE	21
4. MODEL ANALITYCZNY	22
4.1. USER STORIES	22
4.2. DIAGRAM PRZYPADKÓW UŻYCIA	23
4.3. MODEL BAZY DANYCH	24
5. ETAP PROJEKTOWANIA	28
5.1. NAZWA I LOGO APLIKACJI	28
5.2. ZBIÓR DANYCH	29
5.3. NARZĘDZIA	30
5.3.1. <i>PyCharm</i>	30
5.3.2. <i>Git i GitHub</i>	31
5.3.3. <i>Firebase</i>	32
5.3.4. <i>Adobe Photoshop</i>	32
5.4. TECHNOLOGIE	33
5.4.1. <i>Python</i>	33
5.4.2. <i>Pyrebase</i>	33
5.4.3. <i>Kivy i KivyMD</i>	33
5.4.4. <i>Pozostałe biblioteki</i>	34
5.4.5. <i>Plik JSON</i>	34

6.	SYSTEM REKOMENDACJI	35
6.1.	PODOBIĘSTWO COSINUSOWE – COSINE SIMILARITY	35
6.2.	PSEUDOKOD.....	36
7.	IMPLEMENTACJA	40
7.1.	FRONTEND.....	42
7.1.1.	<i>Ekran startowy</i>	<i>43</i>
7.1.2.	<i>Ekran rejestracji.....</i>	<i>44</i>
7.1.3.	<i>Ekran logowania.....</i>	<i>46</i>
7.1.4.	<i>Ekran główny.....</i>	<i>47</i>
7.1.5.	<i>Ekran pozycji perfumy</i>	<i>48</i>
7.1.6.	<i>Moduł ankiety</i>	<i>50</i>
7.1.7.	<i>Moduł rekomendacyjny.....</i>	<i>53</i>
7.1.8.	<i>Moduł wyszukiwania</i>	<i>54</i>
7.2.	BACKEND	56
7.2.1.	<i>Logowanie i rejestracja</i>	<i>56</i>
7.2.2.	<i>Moduł ankiety preferencji</i>	<i>58</i>
7.2.3.	<i>Moduł rekomendacyjny.....</i>	<i>60</i>
7.2.4.	<i>Moduł wyszukiwania</i>	<i>63</i>
7.2.5.	<i>Tłumaczenie informacji ze zbioru danych.....</i>	<i>64</i>
8.	TESTOWANIE.....	66
8.1.	PROBLEM Z DANymi	66
8.2.	PROBLEMY IMPLEMENTACYJNE	67
8.3.	PROBLEMY WYDAJNOŚCIOWE	67
8.4.	TESTY AKCEPTACJI UŻYTKOWNIKÓW	68
9.	BADANIE SYSTEMU REKOMENDACJI	69
9.1.	ANALIZA ZESTAWU DANYCH.....	69
9.2.	BADANIE KONKRETNYCH POZYCJI ZAPACHOWYCH.....	69
9.2.1.	<i>Badanie perfumy dla kobiet: Romance</i>	<i>69</i>
9.2.2.	<i>Badanie perfumy dla mężczyzn: 24 Pure</i>	<i>72</i>
9.2.3.	<i>Badanie perfumy unisex: Wisal</i>	<i>75</i>
9.3.	PODSUMOWANIE	78
10.	PODSUMOWANIE	79
	BIBLIOGRAFIA	80

SPIS RYSUNKÓW

<i>Rysunek 2.1 Aplikacja Sklep Play a) wyszukiwanie frazy: perfumy z warunkiem b) wyszukiwanie frazy: perfumy bez warunku</i>	13
<i>Rysunek 2.2 Interfejs aplikacji Parfumo a) pytanie o zapach w stosunku do płci b) pytanie o okazję c) pytanie o ulubiony zapach</i>	15
<i>Rysunek 2.3 Interfejs aplikacji PERFUMIST a) ekran główny b) ekran produktu</i>	16
<i>Rysunek 4.1 Diagram przypadków użycia wykonany w Creately</i>	23
<i>Rysunek 4.2 Przykładowe dane z obiektu users</i>	24
<i>Rysunek 4.3 Przykładowe dane z obiektu perfumes</i>	25
<i>Rysunek 4.4 Przykładowe dane z obiektu surveys</i>	26
<i>Rysunek 4.5 Reguły bazy danych</i>	26
<i>Rysunek 5.1 Wygląd logotypu aplikacji</i>	28
<i>Rysunek 5.2 Wygląd sygnetu aplikacji</i>	29
<i>Rysunek 5.3 Widok modelu zbioru danych z Kaggle</i>	29
<i>Rysunek 5.4 Interfejs środowiska programistycznego PyCharm</i>	31
<i>Rysunek 5.5 Prywatne repozytorium na platformie GitHub</i>	32
<i>Rysunek 6.1 Cosine similarity</i>	35
<i>Rysunek 7.1 Struktura projektu aplikacji</i>	40
<i>Rysunek 7.2 Plik README.md</i>	42
<i>Rysunek 7.3 Ekran startowy aplikacji</i>	43
<i>Rysunek 7.4 Ekran rejestracji aplikacji</i>	44
<i>Rysunek 7.5 Ekran rejestracji – błąd formatu email</i>	45
<i>Rysunek 7.6 Ekran rejestracji – błąd siły hasła</i>	45
<i>Rysunek 7.7 Ekran rejestracji – błąd zgodności haseł</i>	46
<i>Rysunek 7.8 Ekran rejestracji – błąd zgodności emaila</i>	46
<i>Rysunek 7.9 Ekran logowania aplikacji</i>	47
<i>Rysunek 7.10 Ekran rejestracji – komunikat błędu</i>	47
<i>Rysunek 7.11 Ekran główny aplikacji</i>	48
<i>Rysunek 7.12 Ekran pozycji perfumy aplikacji</i>	49
<i>Rysunek 7.13 Ikona, reprezentująca perfumy dla kobiet</i>	49
<i>Rysunek 7.14 Ikona, reprezentująca perfumy dla mężczyzn</i>	49
<i>Rysunek 7.15 Ikona, reprezentująca perfumy unisex</i>	49
<i>Rysunek 7.16 Wyświetlanie zapachu perfumy</i>	50
<i>Rysunek 7.17 Wyświetlanie nut bazy perfumy</i>	50
<i>Rysunek 7.18 Ekran modułu ankiety</i>	51
<i>Rysunek 7.19 Komunikat błędu dla pierwszej pozycji zapachowej</i>	52
<i>Rysunek 7.20 Komunikat błędu dla pierwszej i drugiej pozycji zapachowej</i>	52
<i>Rysunek 7.21 Ekran modułu rekomendacyjnego</i>	53
<i>Rysunek 7.22 Komunikat o błędzie w module rekomendacyjnym</i>	54
<i>Rysunek 7.23 Ekran modułu wyszukiwania</i>	55
<i>Rysunek 8.1 Model V</i>	66
<i>Rysunek 9.1 Pozycja o indeksie 18 – Romance</i>	70
<i>Rysunek 9.2 Debugger – posortowana lista dla perfumy Romance</i>	70
<i>Rysunek 9.3 Pozycja o indeksie 182 - Hawaii London</i>	71
<i>Rysunek 9.4 Lista rekomendacji dla perfumy Romance</i>	72
<i>Rysunek 9.5 Pozycja o indeksie 17 – 24 Pure</i>	73
<i>Rysunek 9.6 Debugger – posortowana lista dla perfumy 24 Pure</i>	73
<i>Rysunek 9.7 Pozycja o indeksie 338 – Lanvin</i>	74

<i>Rysunek 9.8 Lista rekomendacji dla perfumy 24 Pure</i>	<i>75</i>
<i>Rysunek 9.9 Pozycja o indeksie 575 – Wisal.....</i>	<i>76</i>
<i>Rysunek 9.10 Debugger – posortowana lista dla perfumy Wisal.....</i>	<i>76</i>
<i>Rysunek 9.11 Pozycja o indeksie 609 – Oxygene.....</i>	<i>77</i>
<i>Rysunek 9.12 Lista rekomendacji dla perfumy Wisal</i>	<i>78</i>

SPIS LISTINGÓW

<i>Listing 6.1 Pseudokod metody rekomendacyjnej rec_func</i>	<i>39</i>
<i>Listing 7.1 Konfiguracja połączenia z bazą.....</i>	<i>56</i>
<i>Listing 7.2 Metoda registration.....</i>	<i>57</i>
<i>Listing 7.3 Metoda login.....</i>	<i>58</i>
<i>Listing 7.4 Metoda logout</i>	<i>58</i>
<i>Listing 7.5 Klasa SurveyWindow.....</i>	<i>59</i>
<i>Listing 7.6 Metoda get_survey.....</i>	<i>59</i>
<i>Listing 7.7 Metoda add_survey</i>	<i>60</i>
<i>Listing 7.8 Metoda rec_func.....</i>	<i>61</i>
<i>Listing 7.9 Klasa RecommendationWindow</i>	<i>63</i>
<i>Listing 7.10 Klasa DiscoverMain.....</i>	<i>64</i>
<i>Listing 7.11 Metoda get_perfumes_titles</i>	<i>64</i>
<i>Listing 7.12 Metoda set_items_base</i>	<i>65</i>

SPIS RÓWNAŃ

<i>Równanie 1</i>	36
<i>Równanie 2</i>	36

1. Wstęp

Internet, powstały w latach sześćdziesiątych [Historia Internetu, 2022], od początku swojego istnienia rozwijał się w bardzo szybkim tempie. Ogólnosiwiatowa sieć komputerowa stworzona została na podstawie eksperymentu, jaki miał na celu skonstruowanie takiej infrastruktury łączności, która nie posiada centrali oraz umożliwia automatyczne wyszukiwanie połączeń między komputerami. ARPA wsparła koncept projektu i w ramach sieci ARPANET zostało połączone ze sobą kilka uniwersytetów w Stanach Zjednoczonych. Uważa się do dzisiaj ARPANET za bezpośredniego przodka Internetu. W ciągu następnych lat sieć sprawnie rozbudowywano, dzięki zaangażowaniu coraz większej liczby naukowców.

W roku 1989 z inicjatywy Tima Bernersa-Lee [Berners-Lee, 2022] został przedstawiony projekt nazwany World Wide Web (WWW). Miał on za zadanie umożliwić komunikację naukowcom poprzez wykorzystanie dokumentów hipertekstowych. Rok później stworzona została pierwsza strona internetowa, a dwa lata później pierwsza graficzna przeglądarka WWW – Mosaic.

Ilość danych udostępnianych przez sieć jest nieporównywalnie większa niż zaledwie dziesięć lat temu. Użytkownicy Internetu przyzwyczajeni zostali do bogactwa oraz wygody, jakie oferuje globalna sieć, toteż wpływ cyfryzacji i automatyzacji na codzienne życie nie mógł zostać zignorowany. Statystyki prezentują, iż w Polsce oraz Niemczech przez Internet kupuje aż 77% obywateli [Branża e-commerce w Polsce, 2023], natomiast we Francji już 67%. Dodatkowo pandemia COVID-19 tylko przyspieszyła rozwój handlu elektronicznego – e-commerce umożliwił klientom dostęp do olbrzymiej bazy produktów, mimo obowiązujących restrykcji [E-commerce i COVID-19, 2023].

Wraz ze wzrostem handlu internetowego, powiększyła się cała branża kosmetyczna, a co za tym idzie również trend perfumeryjny. Polacy przykładają coraz większą uwagę do wyboru pozycji zapachowych – decydują się na większe pojemności produktów oraz skuteczniejsze formuły [Rośnie rynek kosmetyków, 2022].

Co ciekawe, mimo gwałtownego wzrostu zainteresowania perfumami, na rynku nie ma przystępnych, niekomercyjnych narzędzi do wyboru pozycji zapachowych. Większość dostępnych aplikacji to produkty wielkich korporacji skupionych na masowej sprzedaży perfum. Istnieje niezagospodarowana nisza w obrębie aplikacji niekomercyjnych, które w prosty sposób pomogą użytkownikowi, niezaznajomionemu z perfumiarstwem, dobrać odpowiednie pozycje.

1.1. Cel pracy

Celem pracy jest opracowanie oraz realizacja aplikacji rekomendującej dobór perfum na podstawie preferencji i składników, przy wykorzystaniu mechanizmu analizy danych cosine similarity.

Omawiana aplikacja ma za zadanie umożliwiać dostęp do bazy perfum oraz oferować funkcjonalność rekomendacji. Korzystanie z aplikacji jest dopuszczalne tylko po zakończeniu procesu rejestracji sukcesem – użytkownik tworzy konto, podając swój adres email, a także

wybierając hasło. Zarejestrowane osoby, poprzez późniejsze logowanie, mają mieć możliwość wyszukiwania perfum po nazwie, a także oglądania ich pełnej specyfikacji. Aplikacja powinna udostępniać ankietę preferencji, którą zalogowany użytkownik będzie mógł wypełnić oraz edytować, by otrzymać dostęp do systemu, rekomendującego mu konkretne perfumy w postaci listy. Przeznaczeniem pracy jest również zapoznanie się z mechanizmem analizy danych cosine similarity oraz zastosowaniem go do budowy systemu rekomendacyjnego.

1.2. Zakres realizowanych prac

Przed rozpoczęciem realizacji niniejszej pracy należało pozyskać odpowiednią wiedzę, a także zaplanować zadania, których utworzenie umożliwiło podzielenie całości projektu na mniejsze problemy w celu uzyskania efektywniejszej pracy. Przygotowano poniższą listę zadań:

- odnalezienie informacji na temat systemów rekomendacyjnych,
- zapoznanie się z mechanizmem analizy danych cosine similarity,
- określenie wymagań funkcjonalnych i нефункциональных,
- opracowanie konceptu ankiety, dotyczącej zbierania preferencji zapachowych,
- znalezienie i zweryfikowanie wydajności zbioru danych na temat perfum,
- wybranie narzędzi do opracowania oraz utworzenia bazy danych ze wcześniej znalezionego zestawu danych,
- powiększenie wiedzy z zakresu posługiwania się językiem Python [Python, 2022],
- zapoznanie z technologią Kivy [Kivy, 2022],
- analiza technologii, które można użyć do utworzenia systemu rekomendacyjnego,
- implementacja aplikacji,
- implementacja systemu rekomendacyjnego,
- testowanie aplikacji,
- badanie systemu rekomendacyjnego,
- opracowanie wniosków z działania systemu rekomendacyjnego.

1.3. Treść pracy

Niniejsza praca została podzielona na dziesięć rozdziałów. Każdy z nich prezentuje odrębne zagadnienie, dotyczące tworzenia aplikacji.

W ramach pierwszego rozdziału określone zostały podstawy takie jak: wstęp, cel pracy, a także zakres realizowanych prac. Dodatkowo opisano w nim krótko całą treść pracy.

Drugi rozdział skupia się na analizie rynku aplikacji mobilnych o perfumach oraz systemów rekomendacyjnych, w celu rozeznania się w tym, jaką metodę najlepiej wybrać, aby zaprojektować omawianą aplikację.

W trzecim rozdziale opisane zostały wymagania funkcjonalne oraz нефункциональные, potrzebne do utworzenia aplikacji.

Czwarty rozdział obejmuje prezentację user stories, a także diagramu przypadków użycia. Ponadto omówiono w nim również model bazy danych.

W rozdziale piątym wyjaśniono narzędzia oraz technologie użyte do zaprojektowania i utworzenia aplikacji oraz systemu rekomendacji. Przedstawiono szczegółowy opis zbioru danych, a także zawarto w nim objaśnienie nazwy i logo aplikacji,

Rozdział szósty zawiera charakterystykę projektu systemu rekomendacji, a także sposób jego działania w postaci pseudokodu.

W ramach rozdziału siódmego zaprezentowano implementację projektu. W pierwszej części rozdziału ukazano interfejs aplikacji i sposób interakcji użytkownika z nim. W drugiej przedstawiono kod oprogramowania razem z jego wyjaśnieniem.

W rozdziale ósmym autorka przedstawiła testowanie aplikacji oraz wyłoniła wszystkie problemy, które otrzymała, dzięki testom. Zaprezentowano problemy: z danymi, implementacyjne, wydajnościowe oraz testy akceptacji użytkowników.

Rozdział dziewiąty obejmował analizę zestawu danych w perspektywie działającego systemu rekomendacyjnego, a także badanie konkretnych pozycji zapachowych oraz podsumowanie otrzymanych wyników.

Rozdział dziesiąty zawiera podsumowanie niniejszej pracy. Pokazuje również ewentualny, dalszy plan na rozwój aplikacji.

Przypisy bibliograficzne zostały zapisane zgodnie z notacją harwardzką [Harvard 2022].

2. Wprowadzenie

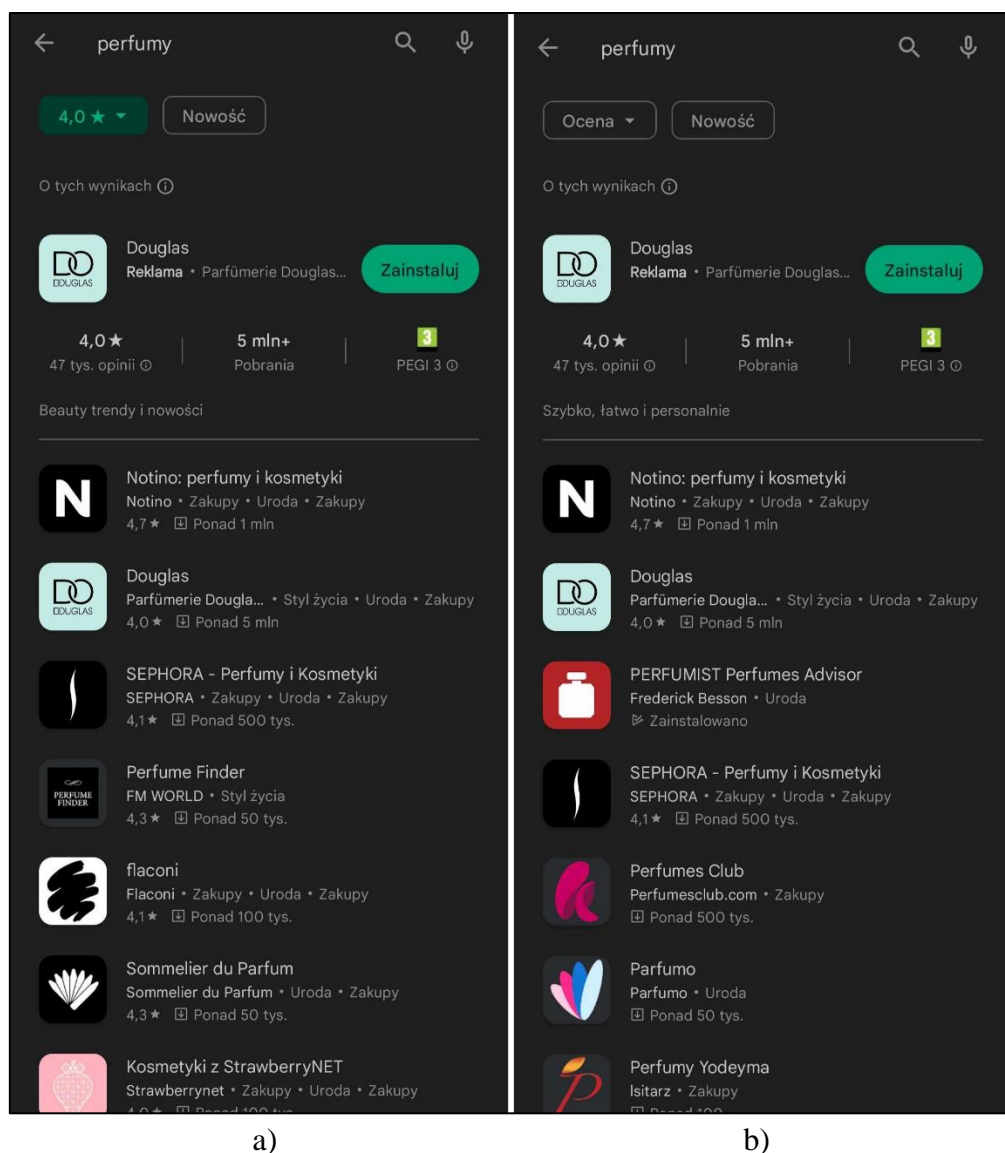
Masowa globalizacja miała ogromny wpływ na spopularyzowanie branży kosmetycznej. Import oraz eksport produktów higieny osobistej przyczynił się do gwałtownego rozwoju ekonomicznego, co spowodowało, iż obecnie rynek kosmetyczny w Polsce wycenia się na wartość około 22 mld złotych [Rozwój branży kosmetycznej, 2022]. Warto wspomnieć, że rosnąca świadomość konsumentów i powiększający się dochód rozporządzalny, w ostatnich latach wzmocniły znaczenie omawianego sektora gospodarki. PMR prognozuje, iż w 2022 roku nastąpi wzrost rynku kosmetycznego o 6,1% do około 27,7 miliardów złotych [PMR, 2022].

Naturalnie na znaczeniu zyskały również perfumy – trend perfumeryjny urósł w siłę wraz z całą branżą. Badania pokazują, iż Polacy coraz częściej sięgają po nowe, luksusowe produkty zapachowe. Sprzedaż perfum kobiecych w pierwszej połowie 2022 roku wzrosła aż o 137%, natomiast męskich podwoiła się w porównaniu z tym samym przedziałem czasowym z poprzedniego roku [Rośnie rynek kosmetyków, 2022]. Co ciekawe, mimo zakwalifikowania perfum do dominującego trendu kosmetycznego, znalezienie odpowiedniego produktu nie jest proste. Osoby poszukujące nowych pozycji zapachowych często wpadają w pułapkę nadmiaru informacji oraz sprzecznych opinii ekspertów, co sprzyja osłabieniu zainteresowania tematyką perfum. Mocno indywidualne potrzeby konsumentów są obecnie zdeorganizowane przez propagandę marketingową, którą ukrywa się pod bogatymi opisami perfum na stronach internetowych, kolorowymi grafikami czy estetycznymi reklamami telewizyjnymi. Dodatkowo kwestia automatyzacji omawianego sektora również pozostała niejednoznaczna, co wynika bezpośrednio z braku technologii, które umożliwiłyby ocenę specyfikacji zapachu poprzez ekran komputera czy telefonu.

2.1. Analiza rynku aplikacji mobilnych o perfumach

Przed rozpoczęciem tworzenia systemu, warto przeprowadzić analizę rynku, aby określić znaczenie oprogramowania w porównaniu do zbioru już istniejących rozwiązań o podobnej tematyce. Twórca powinien zdawać sobie sprawę z jego istotności oraz przydatności na tle konkurencji jeszcze przed zainicjowaniem procesu opracowywania wymagań, żeby mieć możliwość wytworzenia niepowtarzalnego produktu.

Do przeanalizowania rynku użyty został internetowy sklep Google Play. Sklep Play jest fabrycznie zainstalowany na urządzeniach z Androidem, toteż każdy użytkownik telefonu z tym systemem operacyjnym ma do niego dostęp [Sklep Google Play, 2022]. Obecność płatnych oraz darmowych aplikacji sprawia, iż Google Play jest odpowiednim miejscem na przegląd dostępnych systemów oraz ich eksplorację.



Rysunek 2.1 Aplikacja Sklep Play
 a) wyszukiwanie frazy: perfumy z warunkiem
 b) wyszukiwanie frazy: perfumy bez warunku
 [źródło: Sklep Google Play 2022]

Na rysunku 2.1 można zauważyć dwa zrzuty ekranu ze sklepu internetowego Google Play, które przedstawiają rezultat wyszukiwania po wpisaniu frazy: perfumy. Po lewej stronie znajduje się wynik dostępnych aplikacji z dodatkowym warunkiem, jakim jest przefiltrowanie po wszystkich produktach z oceną większą niż 4 gwiazdki, w celu zawężenia poszukiwań do najlepszych produktów na rynku. Po prawej wyświetlona jest lista bez dodatkowego warunku.

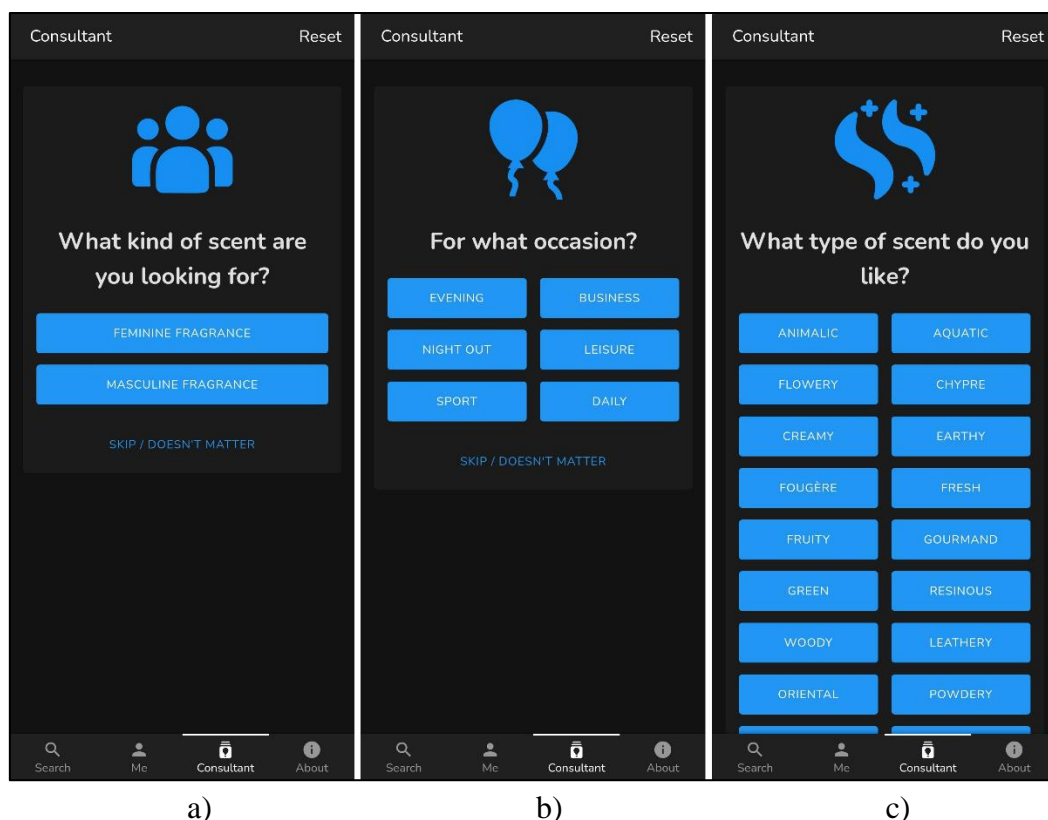
W przypadku wyszukiwania z wymaganiem wysokiej oceny, pokazują się same aplikacje, które należą do firm nastawionych na sprzedaż swoich produktów – nie zależy im na pomocy użytkownikowi. Na starcie tracą wiarygodność, ograniczając siebie oraz konsumentów własną polityką sprzedażową oraz bazą danych – każda z nich posiada jedynie perfumy, jakie sklep ma na stanie, przez co rekomendacja może mieć charakter bardzo niewiarygodny. Dodatkowo w większości nie są to aplikacje tylko skoncentrowane na produktach zapachowych, ponieważ w swojej ofercie posiadają towary z pozostałych sektorów branży

kosmetycznej; to oznacza, że dla każdej osoby, chcącej otrzymać rekomendację w sposób szybki i prosty, takie rozwiązanie będzie niewydajne oraz niewygodne.

Podczas wyszukiwania bez dodatkowego warunku, niektóre rezultaty się pokryły – w wyniku wciąż pozostały produkty największych korporacji kosmetycznych. Jednak pojawiły się również pozycje oficjalnie niepowiązane, niewspółpracujące z markami perfum, czyli Parfumo i PERFUMIST. Obie aplikacje posiadają duże, różnorodne bazy danych zapachów, a także możliwość otrzymania rekomendowanych pozycji, co sprawia, iż stają się interesujące dla użytkownika, który chciałby odkrywać nowe perfumy.

2.1.1. Aplikacja Parfumo

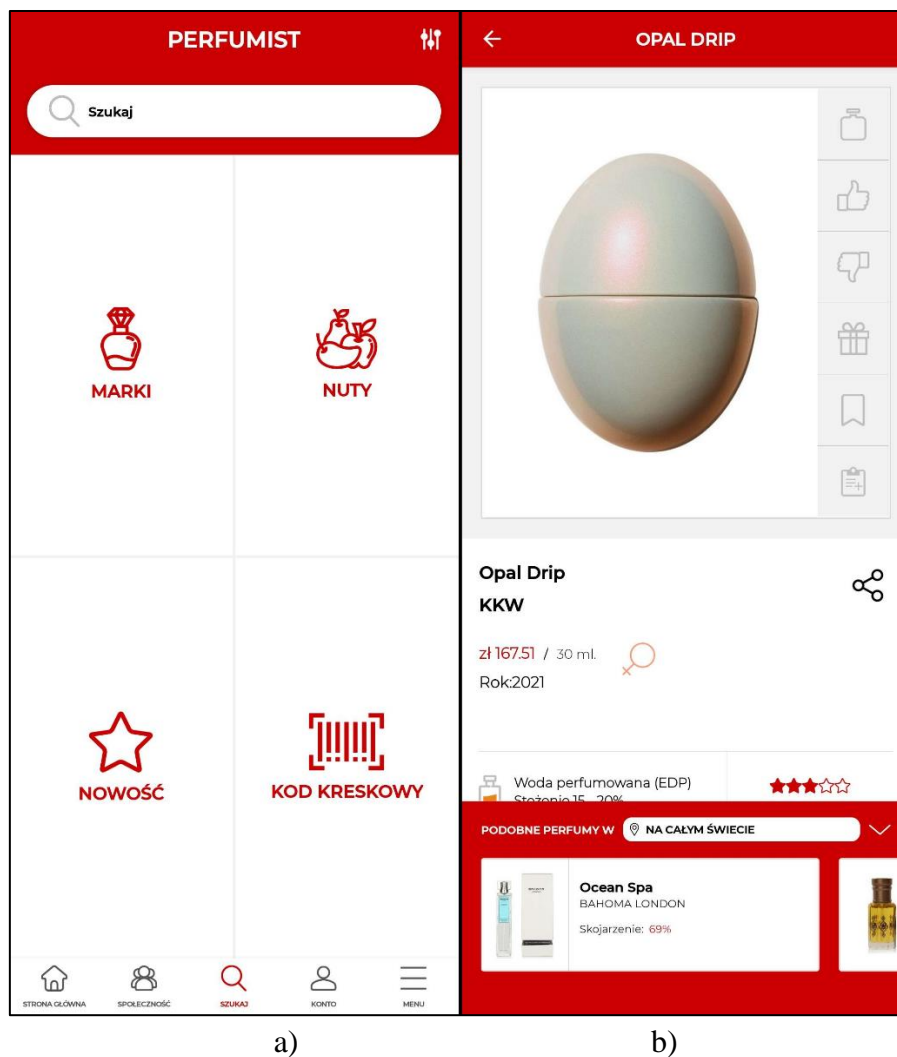
W przypadku aplikacji Parfumo, której interfejs widzimy na rysunku 2.2, aby otrzymać polecane produkty, należy wypełnić ankietę (konsultację) złożoną z kilku pytań. Tylko jedno z nich rzeczywiście dotyczy składu, ale nie pozostawia się użytkownikowi możliwości wyboru więcej niż jednego, zaproponowanego zapachu. Pojawia się więc wątpliwość dotycząca zbyt ogólnego podejścia do struktur kompozycji. Wybierając tylko jedno określenie zapachu, rezygnujemy ze szczegółowości i precyzji polecenia na rzecz ilości rekomendowanych pozycji. Mimo że zbiór perfum zostaje zmniejszony, początkujący konsument otrzymuje listę pozycji, która nie informuje go, w jakim stopniu jest zgodna z tym, co wcześniej wypełnił w ankiecie. Ponadto aplikacja jest tylko w jednej wersji językowej, po angielsku, toteż niektóre, nieznane nazwy składników mogą wzbudzić irytację lub wymusić zamknięcie aplikacji, aby odnaleźć tłumaczenie. Warto wspomnieć, że chęć zmiany elementu zapachu wiąże ze sobą reset całej konsultacji i ponowne przechodzenie przez paroetapową ankietę, co w perspektywie dłuższego użytkowania może męczyć.



Rysunek 2.2 Interfejs aplikacji Parfumo
a) pytanie o zapach w stosunku do płci
b) pytanie o okazję
c) pytanie o ulubiony zapach
[źródło: Aplikacja Parfumo 2022]

2.1.2. Aplikacja PERFUMIST

W przypadku PERFUMIST (na rysunku 2.3) otrzymanie polecenia produktu zapachowego nie jest tak oczywiste jak w Parfumo. Użytkownik nie wypełnia ankiety ani też nie odpowiada na pytania. Ponadto strona główna nie sugeruje, iż aplikacja umożliwia jakąkolwiek rekomendację. Dostać się do modułu polecenia trzeba poprzez kliknięcie w konkretną pozycję zapachową. Wtedy dopiero wyświetlone są odpowiednie produkty razem z procentem skojarzenia. Liczba rekomendowanych w ten sposób perfum nie przekracza 15 pozycji. Należy wspomnieć o tym, że aplikacja dostępna jest w różnych wersjach językowych, także po polsku. Posiada również wiele innych funkcjonalności, które niestety mogą zostać przez użytkownika przeoczone, niewykorzystane (głównie przez nieintuicyjne miejsce ich osadzenia).



Rysunek 2.3 Interfejs aplikacji PERFUMIST
a) ekran główny
b) ekran produktu
[źródło: aplikacja PERFUMIST 2022]

2.2. Analiza systemów rekomendacji

Systemy rekomendacyjne są używane przez największe korporacje i pionierów w różnorodnych branżach, aby polepszyć sprzedaż, zwiększyć współczynnik CTR (ang. Click-through rate, czyli „współczynnik klikalności”, który jest procentowym określeniem stosunku liczby kliknięć do liczby wyświetleń) [CTR, 2023] czy też umocnić pozycję na rynku poprzez pomyślne zaspokajanie potrzeb klientów. W tym podrozdziale zostaną krótko scharakteryzowane trzy główne podejścia do metod filtrowania treści oraz miejsca, w których systemy na nich bazujące zostały wykorzystane.

2.2.1. Metoda oparta na treści

System oparty na treści wykorzystuje metodę filtrowania, która bazuje na zbieraniu, a także przeprowadzaniu analizy treści (np. preferencji użytkowników) w celu wytworzenia listy rekomendowanych pozycji. Ogólne założenie filtrowania opartego na treści jest następujące: jeżeli użytkownikowi spodoba się określona pozycja, istnieje bardzo duże prawdopodobieństwo, iż przypadnie mu do gustu produkt o zbliżonej specyfikacji. Metoda używa specyfikacji danej pozycji, aby utworzyć zespół cech, na których zostaną przeprowadzone wszystkie analizy – każdy zbiór cech danego produktu będzie osobno porównywany z zestawami cech innych produktów, szukając największej zgodności [Krysik, A., 2022]. Zaletą tej metody jest brak problemu ze startem systemu, ponieważ system potrzebuje danych towaru, a nie relacji użytkowników z produktami. Dodatkowo jest adaptacyjny i pozycje polecane nie zależą od innych użytkowników – liczą się wyłącznie preferencje osoby, chcącej uzyskać polecenie, co skutkuje tym, iż polecane są nawet te niepopularne produkty. Jednakże wadę stanowi wymaganie dostarczenia wszystkich niezbędnych informacji o produkcie, wszelkie braki będą przeszkadzać systemowi rekomendującemu w wyszukaniu najtrafniejszych pozycji [Holewa, K., 2022].

Przykład zastosowania metody filtrowania opartej na treści prezentuje serwis YouTube [YouTube, 2022], który umożliwia swoim użytkownikom wstawianie i odtwarzanie filmów oraz muzyki. System analizuje treść filmów, oglądanych przez użytkownika, a następnie poleca mu pozycje skupione wokół tego samego zagadnienia, tematu [Krysik, A., 2022].

2.2.2. Metoda grupowego filtrowania

Metoda grupowego filtrowania w systemach rekomendacyjnych polega na analizie danych pozyskanych od użytkowników, którzy weszli w interakcję z podobnymi pozycjami (np. kupili je lub polubili), by następnie połączyć te informacje, w celu utworzenia rekomendacji dla kolejnego użytkownika, wchodzącego w kontakt z przeanalizowanymi produktami. Głównym założeniem grupowego filtrowania jest koncept, iż osoby, dokonujące wyboru, będą preferować polecenia oparte na decyzjach, jakie podjęli inni użytkownicy w przeszłości [Krysik, A., 2022]. Zaletą tej metody jest fakt, iż specyfikacja produktu nie musi być „rozumiana” przez system. Ponadto gwarantuje ona adaptacyjność. Niestety jednocześnie istnieje duża wada, jaką stanowi brak możliwości otrzymania rekomendacji w przypadku nowych użytkowników, którzy nie mieli żadnej interakcji w obrębie systemu. To samo dotyczy pozycji, jaka nigdy nie została wcześniej wybrana – nie zostanie nikomu polecona [Holewa, K., 2022].

Przykład użycia metody grupowego filtrowania występuje na cyfrowej platformie muzycznej Spotify [Spotify, 2022], która służy do odtwarzania m.in. muzyki oraz podcastów. Wykorzystując zachowanie, czyli interakcję użytkowników z pozycjami, z przeszłości, tworzy dla nich rekomendacje w postaci np. nowych list utworów do przesłuchania [Kumar, A., 2022].

2.2.3. Metoda filtrowania hybrydowego

Hybrydowy system rekomendacyjny składa się z połączenia metody grupowego filtrowania, a także metody filtrowania opartej na treści. To rozwiązanie często sprawdza się bardziej niż korzystanie z obu metod oddzielnie, ponieważ twórcy systemu są w stanie zaprojektować strukturę, która będzie najbardziej funkcjonalna dla ich konkretnych potrzeb. Nic dziwnego, że taka taktyka szybko zdobyła popularność. Obecnie nawet gigantyczne korporacje, posiadające funkcje oparte na poleceniu produktów, korzystają z metody hybrydowego filtrowania. Największa amerykańska spółka akcyjna, która zajmuje się handlem elektronicznym, Amazon [Amazon, 2022] opiera się na połączeniu grupowego filtrowania razem z tym opartym na treści. Dzięki temu jest w stanie wykorzystać zachowanie klientów z przeszłości, aby polecić im nowe, specjalnie dopasowane, produkty. Co ciekawe, Amazon posiada jeden z najbardziej skomplikowanych oraz trafnych systemów rekomendacyjnych na świecie [Kumar, A., 2022].

2.3. Podsumowanie

Dzięki przeprowadzonej analizie dostępnych aplikacji, dotyczących perfum, określono wymagania funkcjonalne oraz niefunkcjonalne, potrzebne do wytworzenia prostej aplikacji, polecającej pozycje zapachowe użytkownikom nieznającym się na perfumiarstwie. Z przeglądu aplikacji oraz metod wynika, że jedna grupa aplikacji to głównie produkty sklepów kosmetycznych, które mają ułatwić dokonanie w nich zakupów. Sektor, dotyczący perfum, nie jest tam na tyle rozwinięty, aby użytkownik, nieposiadający specjalistycznej wiedzy, mógł swobodnie dokonywać wyboru, licząc na uzyskanie zadowalającego zapachu. Druga wyodrębniona grupa to aplikacje niekomercyjne. Niestety większość z nich nie posiada niskiego progu wejścia, oczekując od użytkowników chociaż podstawowej wiedzy na temat kompozycji perfum. Występują w nich skomplikowane interfejsy, a także niekonkretne lub trudno dostępne funkcjonalności. Dlatego też podczas projektowania interfejsu oraz logiki systemu, należy wziąć pod uwagę wszystkie wnioski, aby stworzyć prosty oraz praktyczny produkt.

Natomiast analiza systemów rekomendacyjnych umożliwiła podjęcie decyzji o tym, która metoda będzie najbardziej odpowiednia do zaprojektowania funkcjonalności polecenia perfum. Pracując z dużą ilością danych, jakimi są kompozycje zapachowe, zdecydowano o zastosowaniu metody filtrowania opartej na treści. Pozwoli ona na skuteczne sformułowanie procentu zgodności między pozyskanymi w ankiecie preferencjami użytkownika a pozycjami, jakie znajdują się w bazie danych. To umożliwi otrzymanie rekomendacji perfum bez konieczności posiadania specjalistycznej wiedzy o kompozycjach zapachowych, a także zbudowanej sieci relacji między użytkownikami, tak jak w przypadku metody filtrowania grupowego.

3. Analiza wymagań

Kluczowym etapem realizowania projektu jest przygotowanie specyfikacji wymagań odnośnie funkcjonowania aplikacji. Wszystkie wymogi użytkownika powinny zostać ustrukturalizowane oraz prosto opisane, adekwatnie wykorzystując język naturalny lub różnorodne diagramy stanów czy aktywności [Specyfikacja wymagań na system, 2022]. Opracowany zapis najlepiej skonstruować tak, żeby jego interpretacja nie sprawiała trudności, formując potrzeby w krótkie, jednoznaczne punkty lub przejrzyste schematy. Analiza skupia się wokół określenia wymagań użytkownika odnośnie aplikacji. Wymagania dzielimy na funkcjonalne oraz нефункционалне [Rodzaje wymagań, 2022].

3.1. Wymagania funkcjonalne

Wymagania funkcjonalne są opisem funkcjonalności, jakie powinna oferować aplikacja. Muszą pozostać unikatowe, ponieważ zależą od rodzaju oprogramowania oraz potrzeb potencjalnych użytkowników [Wymagania funkcjonalne i нефункционалне, 2022].

Aplikacja przed zalogowaniem ma za zadanie umożliwiać dostęp do:

- ekranu startowego,
- ekranu rejestracji,
- ekranu logowania.

Aplikacja po zalogowaniu powinna umożliwiać dostęp do:

- ekranu głównego,
- ekranu pozycji perfumy,
- modułu ankiety preferencji,
- modułu rekomendacyjnego,
- modułu wyszukiwania.

Ekran startowy musi pozwolić niezalogowanemu użytkownikowi:

- przejść do ekranu rejestracji,
- przejść do ekranu logowania.

Ekran rejestracji musi umożliwić niezalogowanemu użytkownikowi:

- rejestrację nowego konta,
- sprawdzenie poprawności wpisanych danych,
- po rejestracji zakończonej sukcesem przejście do ekranu głównego,
- powrót do ekranu startowego.

Ekran logowania musi pozwolić niezalogowanemu użytkownikowi na:

- zalogowanie do wcześniej utworzonego konta,

- przejście do ekranu rejestracji,
- przejście do ekranu głównego po zalogowaniu,
- powrót do ekranu startowego.

Ekran główny musi umożliwić zalogowanemu użytkownikowi:

- przejście do modułu ankiety preferencji,
- przejście do modułu rekomendacyjnego,
- przejście do modułu wyszukiwania,
- wylogowanie, a co za tym idzie – powrót do ekranu startowego.

Ekran pozycji perfumy musi pozwolić zalogowanemu użytkownikowi na:

- zapoznanie się ze specyfikacją konkretnej pozycji,
- zapoznanie się z zapachem pozycji,
- zapoznanie się z listą nut bazy perfumy,
- zapoznanie się z listą nut serca perfumy,
- powrót do wcześniejszego modułu.

Moduł ankiety preferencji musi umożliwić zalogowanemu użytkownikowi:

- powrót do ekranu startowego,
- wypełnienie ankiety preferencji,
- sprawdzenie dostępności wybranych pozycji,
- zapisanie ankiety preferencji,
- edycję ankiety preferencji,
- przejście do modułu rekomendacji.

Moduł rekomendacyjny musi pozwolić zalogowanemu użytkownikowi na:

- zapoznanie się dostępnymi rekomendacjami perfum oraz ich procentem zgodności,
- przejście do ekranu pozycji konkretnej, wybranej perfumy,
- powrót do ekranu startowego.

Moduł wyszukiwania musi umożliwić zalogowanemu użytkownikowi:

- powrót do ekranu startowego,
- wyszukiwanie perfum po ich nazwach,
- przejście do ekranu pozycji konkretnej, wybranej perfumy.

3.2. Wymagania niefunkcjonalne

Wymagania niefunkcjonalne są opisem właściwości, a także ograniczeń aplikacji. Często wymaganiom niefunkcjonalnym przypada ważniejsza rola niż funkcjonalnym, ponieważ brak ich spełnienia może się przyczynić do zaburzenia użyteczności systemu [Wymagania funkcjonalne i niefunkcjonalne, 2022].

Omawiana aplikacja powinna:

- być intuicyjna oraz prosta w użyciu – laik komputerowy nie powinien mieć problemu podczas nauki obsługi programu,
- mieć przejrzysty interfejs, skonstruowany w zgodzie z heurystykami Nielsena,
- stosować się do podstaw zasad uniwersalnego projektowania aplikacji, czyli posiadać czytelną czcionkę, rozsądne odstępy między komponentami, kolory przyjazne dla użytkownika czy też niski poziom wysiłku fizycznego, podczas korzystania z systemu [Projektowanie uniwersalne, 2022],
- zapewnić działanie systemu rekomendacyjnego bez zakłóceń oraz z jak największą zgodnością,
- posiadać responsywność na rozdzielczość o współczynniku proporcji 16:9, który jest standardem dla smartfonów i innych urządzeń od 2010 roku [Smartphone “Aspect Ratio”, 2022] ,
- być przystępna dla osoby, która nie jest zaznajomiona z zapachami i kompozycjami perfum; umożliwiać wypełnienie ankiety preferencji w sposób niezależny od znajomości konkretnych składników np. poprzez odwoływanie się bezpośrednio do konkretnych, istniejących produktów, z którymi użytkownik mógł mieć wcześniej styczność,
- być maksymalnie bezawaryjna.

4. Model analityczny

Rozdział został podzielony na trzy podrozdziały. W pierwszym przedstawione będą historie użytkowników, utworzone, aby zgromadzić i poznać wszystkie ich wymagania. W drugim zostanie ukazany diagram przypadków użycia, jaki w graficzny sposób obrazuje wszystkie funkcjonalności. W trzecim, ostatnim podrozdziale, zaprezentowana będzie baza danych specjalnie zaprojektowana dla poprawnego działania aplikacji.

4.1. User stories

User story to historyjka użytkownika, którą stanowi prosty oraz krótki opis danej funkcjonalności z perspektywy osoby, korzystającej z systemu [User story, 2022]. Opis, zazwyczaj składający się z jednego zdania, zawiera najważniejsze informacje o danej funkcji. Historyjki trafiają do dokumentacji, żeby ułatwić programistom zrozumienie zdefiniowanych funkcjonalności, a co za tym idzie poprawne ich oprogramowanie. Dodatkowo formułowane są według konkretnego wzorca, w którym najważniejsze jest to, aby była informacja o roli użytkownika, zadaniu, jakie ma wykonać oraz skutku wykonanej czynności. Poniższy szablon zostanie użyty do opisan historyjek użytkownika w dalszej części podrozdziału:

JAKO <kto?> CHCIAŁBYM <co?> PO TO, ABY <czemu?>

User stories niezalogowanego użytkownika:

1. **JAKO** niezalogowany użytkownik **CHCIAŁBYM** móc się zarejestrować **PO TO, ABY** założyć nowe konto w aplikacji.
2. **JAKO** niezalogowany użytkownik **CHCIAŁBYM** móc się zalogować **PO TO, ABY** uzyskać dostęp do wcześniej założonego konta.

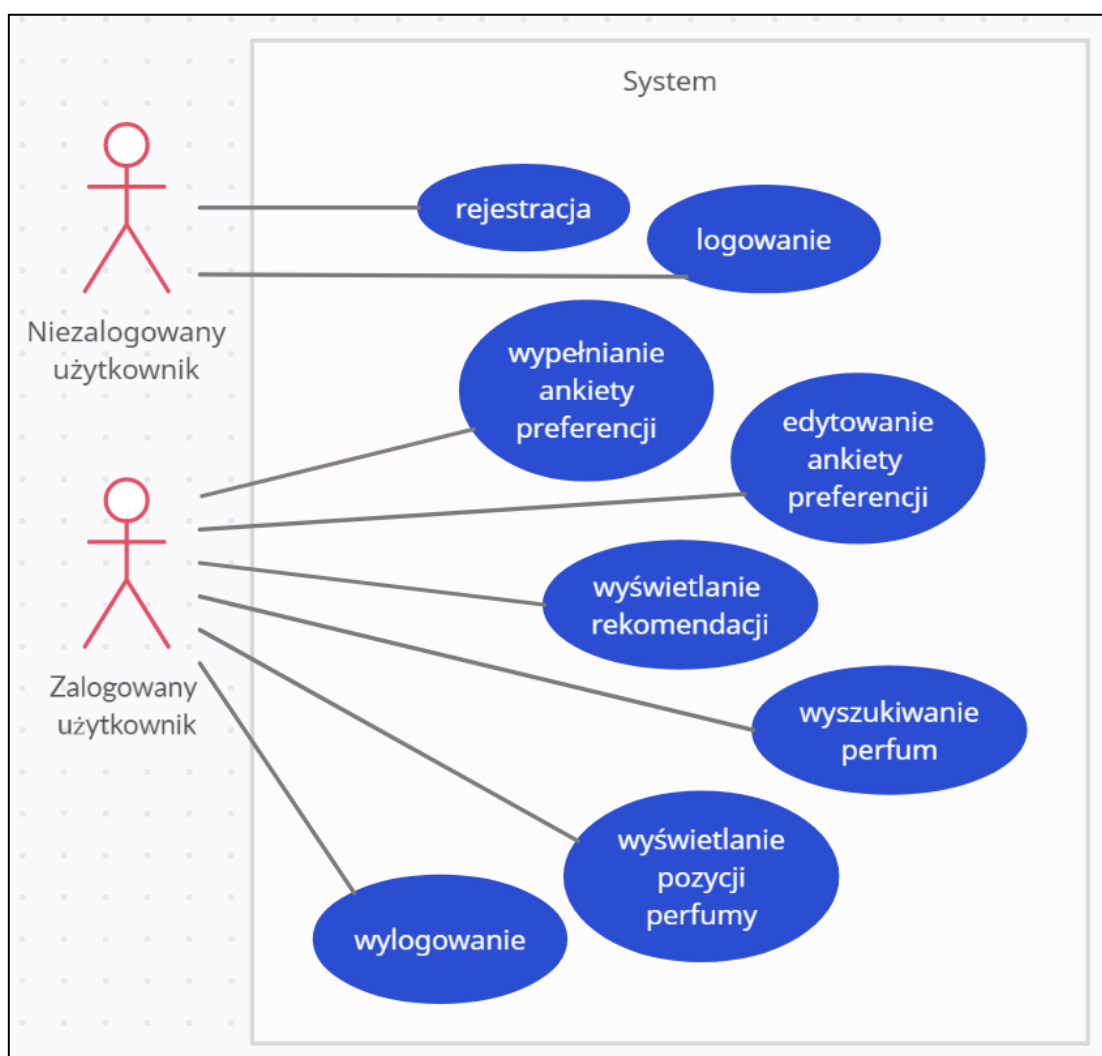
User stories zalogowanego użytkownika:

1. **JAKO** zalogowany użytkownik **CHCIAŁBYM** mieć dostęp do ekranu startowego **PO TO, ABY** otrzymać akces do wszystkich modułów aplikacji.
2. **JAKO** zalogowany użytkownik **CHCIAŁBYM** mieć możliwość wypełnienia ankiety **PO TO, ABY** zdefiniować swoje preferencje zapachowe.
3. **JAKO** zalogowany użytkownik **CHCIAŁBYM** móc edytować ankietę **PO TO, ABY** zmienić wcześniej określone preferencje zapachowe.
4. **JAKO** zalogowany użytkownik **CHCIAŁBYM** mieć opcję wyświetlenia zawartości modułu rekomendacji **PO TO, ABY** odczytać listę pozycji perfum rekomendowanych.
5. **JAKO** zalogowany użytkownik **CHCIAŁBYM** mieć możliwość wyszukiwania perfum po nazwie **PO TO, ABY** przeglądać pozycje dostępnych perfum w bazie.
6. **JAKO** zalogowany użytkownik **CHCIAŁBYM** móc wyświetlić stronę wybranej perfumy **PO TO, ABY** przeanalizować jej specyfikację.

7. **JAKO** zalogowany użytkownik **CHCIAŁBYM** mieć opcję wylogowania się **PO TO, ABY** wyjść z wcześniej utworzonego konta.

4.2. Diagram przypadków użycia

Diagram przypadków użycia (z ang. use case diagram) jest schematem, który ukazuje funkcjonalność systemu oraz jego otoczenie. Tak zaprojektowane rozwiązanie umożliwia przejrzyste modelowanie aplikacji i wymagań, a także łatwe zobrazowanie możliwości systemu z perspektywy użytkownika [Diagram przypadków użycia, 2022]. Na rysunku 4.1, przedstawiono diagram przypadków użycia dla omawianej aplikacji.



Rysunek 4.1 Diagram przypadków użycia wykonany w Creately
[źródło: Creately 2022]

4.3. Model bazy danych

Zawartość tego podrozdziału została poświęcona przedstawieniu projektu bazy danych, który ma zastosowanie w omawianej aplikacji. Głównym zadaniem zaprojektowanej bazy danych jest przechowywanie informacji na temat użytkowników, ich ankiet preferencji, a także zbioru ponad tysiąca perfum.

Do wykonania aplikacji została użyta Firebase Realtime Database, czyli baza danych NoSQL czasu rzeczywistego hostowana w chmurze [Firebase Realtime Database, 2022]. Wszystkie dane w tej bazie przechowywane są hierarchicznie jako obiekty JSON [JSON, 2022], przez co można ją sobie prosto zwizualizować jako drzewo JSON. Zasady działania oraz jej przeznaczenie różnią się od relacyjnych baz danych – struktura NoSQL nie posiada żadnych tabel, rekordów ani relacji. Dodatkowo każdą ze zbudowanych gałęzi można dowolnie modyfikować, a wolność od schematów typowych dla baz SQL, pozwala na przechowywanie informacji w dowolnym formacie. Umożliwia również synchronizację między użytkownikami w czasie rzeczywistym.



Rysunek 4.2 Przykładowe dane z obiektu *users*
[źródło na podstawie: *Firebase Realtime Database 2022*]

Utworzona baza danych, dostosowana do potrzeb omawianej aplikacji, posiada trzy obiekty – *users*, *perfumes*, *surveys*. Na rysunku 4.2 widać wygląd przykładowych danych, dotyczących obiektu o kluczu *users*. Następnie dla każdego użytkownika w obiekcie *users* istnieje jego niepowtarzalny identyfikator (otrzymywany podczas rejestracji) jako kolejny klucz obiektu, który przechowuje bardziej szczegółowe, unikalne dane (email, aktywność i datę ostatniego logowania). Analogicznie wygląda struktura pozostałych obiektów. Koncept ich

budowy skupia się wokół prostoty i płytkości, co powoduje łatwy dostęp do danych oraz zwiększenie wydajności – starano się uniknąć sytuacji, w której może nastąpić pobór większej ilości danych niż jest to potrzebne.

Obiekt `perfumes` został utworzony przy pomocy zbioru danych, jaki wcześniej pobrano z Kaggle. Niemniej jednak, aby wprowadzić go do bazy, korzystając z funkcji Import JSON, którą oferuje Realtime Database, wpierw należało przekonwertować go na JSON z domyślnego formatu CSV [CSV, 2022] (czyli pliku tekstowego z danymi uschematyzowanymi poprzez oddzielenie informacji przecinkami). Na rysunku 4.3 zobrazowany jest wygląd jednego obiektu z `perfumes`, a także cała jego specyfikacja m.in. nuty zapachowe, stężenie czy też marka przedstawiona w formie klucz-wartość. Ponadto kluczem każdego obiektu z kolekcji obiektów `perfumes` jest unikatowy numer indeksu pozycji, umożliwiający formułowanie przejrzystych zapytań do bazy danych.



*Rysunek 4.3 Przykładowe dane z obiektu `perfumes`
[źródło na podstawie: Firebase Realtime Database 2022]*

Ostatnim obiektem jest `surveys`, którego strukturę zobrazował rysunek 4.4. Warto wspomnieć, że od razu po rejestracji zostaje stworzony obiekt w `surveys` o kluczu, który odpowiada unikatowej wartości przypisanej do każdego użytkownika po utworzeniu konta. Klucze, znajdujące się w pojedynczym obiekcie, są cyframi, ponieważ odpowiadają za numer pytania w ankiecie, jaką użytkownik wypełnia podczas korzystania z aplikacji. Ich wartości to kolejno: typ logiczny (odpowiadający płci), nazwa pierwszej perfumy i nazwa drugiej perfumy.



Rysunek 4.4 Przykładowe dane z obiektu *surveys*
[źródło na podstawie: *Firebase Realtime Database 2022*]

Co więcej, w Firebase Realtime Database są reguły bezpieczeństwa, które można dowolnie edytować w każdej chwili korzystania z bazy danych czasu rzeczywistego. Służą one do określenia, kto ma możliwość odczytu, zapisu danych w bazie, a także umożliwia zdefiniowanie odpowiednich indeksów. Reguły działają bez przerwy na serwerach Firebase i domyślnie, od razu po utworzeniu bazy, ustalone są w taki sposób, iż nie pozwalają nikomu na dostęp, chroniąc bazę przed nadużyciami. Na rysunku 4.5 przedstawione są zmiany reguł zabezpieczeń, które zostały specjalnie określone dla omawianej aplikacji. Widoczny jest ten sam warunek dotyczący czytania i wpisywania danych dla wszystkich istniejących obiektów – tylko osoba, która została uwierzytelniona ma prawo do interakcji z informacjami w bazie. Dodatkowo w regułach można indeksować pola, zwiększając wydajność zapytań.

```
{
  "rules": {
    "users": {
      ".read": "auth != null",
      ".write": "auth != null"
    },
    "surveys": {
      ".read": "auth != null",
      ".write": "auth != null"
    },
    "perfumes": {
      ".read": "auth != null",
      ".write": "auth != null",
      ".indexOn": ["department", "new_price", "brand", "name", "scents", "base_note", "middle_note", "concentration"]
    },
  },
}
```

Rysunek 4.5 Reguły bazy danych
[źródło na podstawie: *Firebase Realtime Database 2022*]

Rysunek 4.5 przedstawia również definicję indeksów w obiekcie `perfumes` na polach: `department`, `new_price`, `brand`, `name`, `scents`, `base_note`, `middle_note`, `concentration`.

5. Etap projektowania

Etap projektowania to część pracy, w której przybliża się decyzje oraz rozwiązana podjęte, aby utworzyć w pełni funkcjonalną aplikację. Ten rozdział umożliwi opisanie nazwy oraz logo produktu, a także zbioru danych, jaki zostanie wykorzystany do zbudowania bazy danych. Ponadto zaprezentuje budowę wraz z działaniem systemu rekomendacji i wszystkie narzędzia oraz technologie, które zostały użyte podczas pracy nad aplikacją.

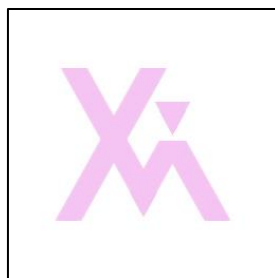
5.1. Nazwa i logo aplikacji

Nazwa aplikacji YasMine została stworzona poprzez przekształcenie angielskiego słowa jasmine (oznaczającego jaśmin) w połączenie dwóch innych słów: Yas – slangowego określenia ekscytacji, które sprowadza się do wykrzyknienia: „yes!” (po polsku: „tak!”); Mine – oznaczającego przynależność (po polsku: „moje”). Jest to gra słów, bazująca na podobnym brzmieniu wyrazów oraz wyróżniająca jaśmin – jeden z najpopularniejszych i najbardziej charakterystycznych zapachów.

Logo aplikacji składa się z logotypu i sygnetu. Logotyp jest graficznym przedstawieniem nazwy, zaś sygnet stanowi symbol graficzny, który ma się kojarzyć z marką [Logo, 2022]. Obie grafiki są autorskie oraz wykonane przy pomocy programu graficznego Adobe Photoshop [Adobe Photoshop, 2022]. Logotyp (na rysunku 5.1) utworzono przy użyciu czytelnej czcionki, a także delikatnych, pastelowych kolorów. W przypadku omawianej aplikacji sygnet (na rysunku 5.2) został zaprojektowany jako połączenie wszystkich liter nazwy aplikacji.



*Rysunek 5.1 Wygląd logotypu aplikacji
[źródło: opracowanie własne]*



Rysunek 5.2 Wygląd sygnetu aplikacji
[źródło: opracowanie własne]

5.2. Zbiór danych

Zbiór danych o perfumach, który wykorzystano do utworzenia bazy danych, został pobrany z platformy Kaggle [Bin Taleb, M., 2022]. Serwis kaggle.com jest największą na świecie społecznością, która gromadzi specjalistów, odpowiadających za analizę danych oraz statystykę, a także inżynierów, jacy poświęcają się pracy nad uczeniem maszynowym. Kaggle posiada otwarte zestawy danych z różnych dziedzin [Kaggle, 2022]. Znajdują się również na nim zbiory, dotyczące perfum. Mimo ograniczonej ilości zestawów o tej tematyce, udało się znaleźć zbiór zbliżony do wymagań omawianej aplikacji.

noon_perfumes_dataset.csv (154.05 kB)								
<div> Detail Compact Column </div> <div>10 of 15 columns</div>								
#	brand	name	old_price	new_price	ml	concentra...	department	
0	PACO RABANNE	1 Million Lucky	395	244.55	100	EDT	Men	
1	Roberto Cavalli	Paradiso Assoluto	415	107.95	50	EDP	Women	
2	S.T.Dupont	Royal Amber	265	186.9	100	EDP	Unisex	
3	GUESS	Seductive Blue	290	103.2	100	EDT	Men	
4	Roberto Cavalli	Uomo	260	94.95	50	EDP	Women	

Rysunek 5.3 Widok modelu zbioru danych z Kaggle
[źródło: Bin Taleb, M. 2022]

Zebrane dane, jak opisuje ich autor, pochodzą ze strony internetowej noon.com, która posiada ogromną ofertę towarów i działa w obrębie regionu Arabii Saudyjskiej. Oczywiście do utworzenia zbioru, wyodrębniono sekcję perfum, a następnie twórca zestawu przerobił je na odpowiedni model (zaprezentowany na rysunku 5.3). Łącznie zebrano 1002 pozycji do późniejszego przekształcenia ich na plik w formacie JSON i zaimportowania do bazy danych [JSON, 2022].

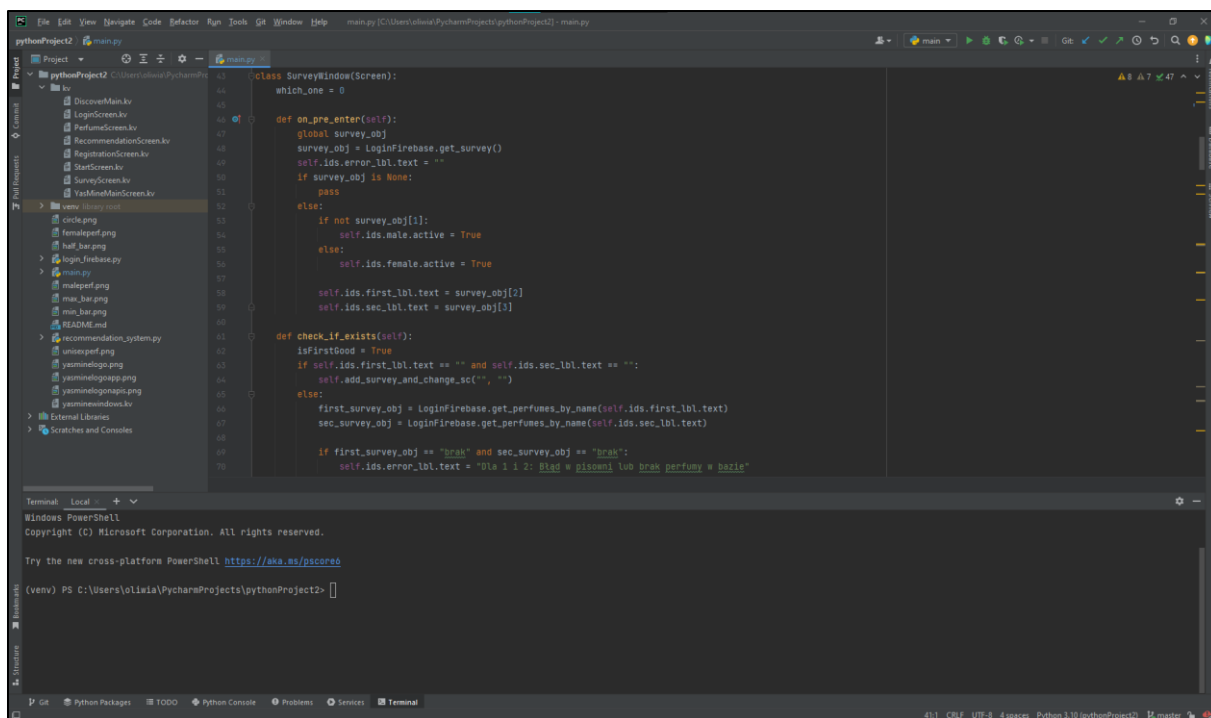
Jednakże trzeba wspomnieć o znaczącej wadzie zestawu, którą jest brak pełnej specyfikacji zaprezentowanych pozycji. Perfumy posiadają unikatową kompozycję – składa się ona z nuty górnej (głowy), środkowej (serca) i dolnej (bazy) [Zasady tworzenia kompozycji zapachowych, 2022]. Każda z nut odparowuje, uwalniając zapach w różnym czasie, przez co ich dominacja w ogólnym odczuciu zapachu jest inna. Niestety znaleziony zestaw danych nie posiada kolumny z nutami górnymi (głowy), co uniemożliwia stworzenie pełnej charakterystyki danego zapachu. Jednocześnie wybrany zbiór autorka niniejszej pracy ocenia jako najlepszy w porównaniu do innych zestawień oferowanych przez Kaggle na ten sam temat.

5.3. Narzędzia

Wykonanie omawianej w pracy dyplomowej aplikacji nie byłoby możliwe bez opisanych poniżej narzędzi.

5.3.1. PyCharm

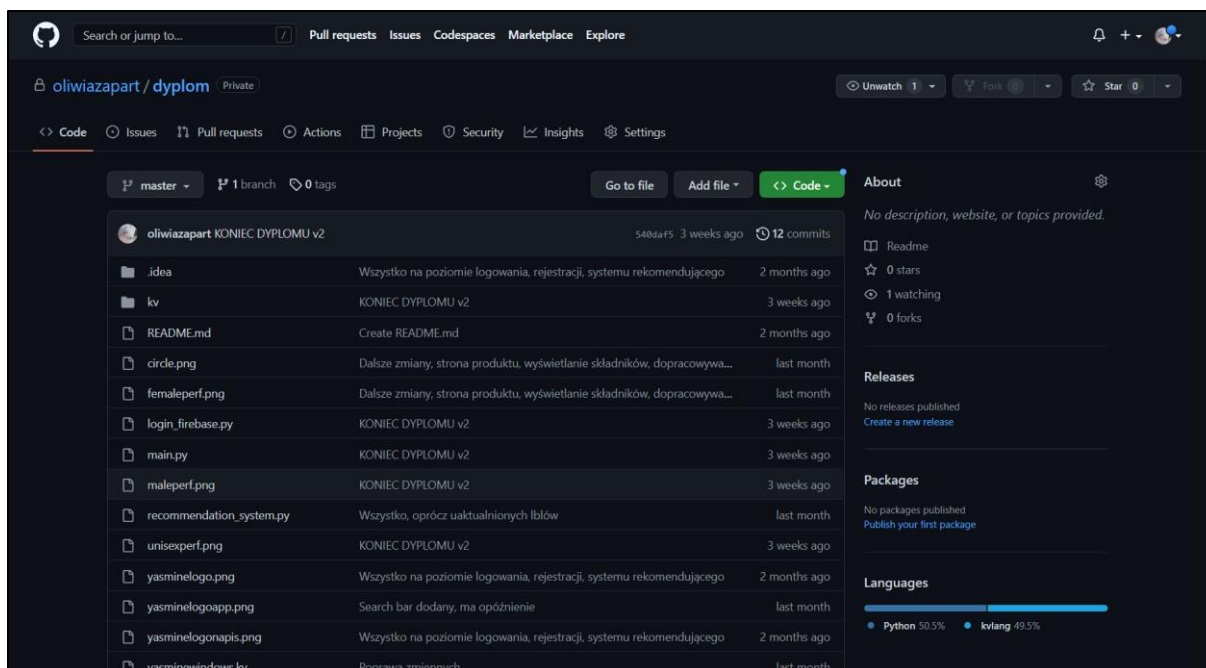
Utworzenie aplikacji było możliwe, dzięki wykorzystaniu zintegrowanego środowiska programistycznego PyCharm [PyCharm, 2022] firmy JetBrains. Umożliwia ono edytowanie kodu, a także jego debuggowanie, czyli sprawdzanie poprawności z wyróżnieniem miejsc, w których zostały popełnione błędy. Dodatkowo środowisko ma wbudowaną kontrolę wersji, automatyczne formatowanie, kolorowanie składni czy też porządkowanie kodu, przez co praca w programie staje się przyjemna, a kod przejrzysty. PyCharm stanowi idealne narzędzie dla początkujących programistów w języku Python, ponieważ posiada prosty i intuicyjny interfejs (widoczny na rysunku 5.4) oraz jego wersja Community jest dostępna za darmo.



*Rysunek 5.4 Interfejs środowiska programistycznego PyCharm
[źródło: PyCharm 2022]*

5.3.2. Git i GitHub

Do zarządzania kodem, podczas procesu tworzenia aplikacji, został użyty system kontroli wersji Git [Git, 2022]. Umożliwił on zapisywanie zmian, które na przestrzeni okresu wykonywania programu, zostały wprowadzone, pozwalając dodać do nich stosowny komentarz, aby twórca wiedział, czemu dokładnie służyła dana modyfikacja. Praca z wykorzystaniem systemu kontroli wersji jest bardzo bezpieczna oraz wygodna, poprzez możliwość cofania zmian i podglądu kodu programu sprzed wielu tygodni. Na rysunku 5.5 jest ukazany wygląd prywatnego repozytorium na gałęzi głównej utworzonego przy pomocy serwisu GitHub [GitHub, 2022], który zawiera w sobie projekt omawianej w pracy aplikacji. GitHub, wykorzystując system kontroli wersji Git, pozwala na przechowanie kodu źródłowego w repozytoriach zdalnych (prywatnych oraz publicznych), dzięki czemu autor posiada dostęp do kodu z dowolnego urządzenia, co zdecydowanie ułatwia i uelastycznia charakter pracy nad programem.



*Rysunek 5.5 Prywatne repozytorium na platformie GitHub
[źródło: GitHub 2022]*

5.3.3. Firebase

Do utworzenia aplikacji wykorzystano Firebase [Firebase, 2022] – jest to kompleksowy zestaw narzędzi do projektowania oraz wdrażania aplikacji, bazujący na usługach chmurowych. Na Firebase składa się łącznie aż 18 produktów, jednakże do implementacji projektu zostały wykorzystane tak naprawdę tylko te, należące do modułu Development (czyli narzędzi do budowy aplikacji). Przede wszystkim platforma umożliwiła stworzenie bazy danych NoSQL czasu rzeczywistego (ang. Realtime Database), która natychmiast synchronizuje wszelkie modyfikacje – za każdym razem, kiedy dane się zmieniają, wszystkie połączone urządzenia otrzymują aktualizację w ciągu milisekund. Nie ma potrzeby stosowania serwera, ponieważ Realtime Database jest dostępna bezpośrednio z przeglądarki lub telefonu. Dodatkowo w ramach Firebase skorzystano z modułu Authentication, jaki umożliwia dostęp do rozbudowanego mechanizmu uwierzytelniania i zarządzania kontami użytkowników, a także kontrolowania dostępu do danych. Różnorodne funkcjonalności Firebase sprawiają, iż jest on najczęściej używany oraz ceniony przez zespoły, tworzące aplikacje (od startupów aż po globalne przedsiębiorstwa) [Kalam, N., 2022].

5.3.4. Adobe Photoshop

Do utworzenia oraz modyfikacji wszelkich grafik, wykorzystanych w aplikacji, użyto programu graficznego Adobe Photoshop [Adobe Photoshop, 2022] firmy Adobe Systems. Photoshop pozwolił na utworzenie logo aplikacji – w tym logotypu i sygnetu – oraz przeróbki ikon perfum, które swoim kolorem mają charakteryzować przynależność płciową. Dzięki dużej

popularności narzędzia nie było problemu z utworzeniem grafik, ponieważ posiada on bardzo rozwiniętą społeczność, a co za tym idzie wiele darmowych poradników.

5.4. Technologie

Do implementacji w pełni funkcjonalnej aplikacji wykorzystano technologie, które zostaną opisane w tym podrozdziale.

5.4.1. Python

Cała aplikacja, jak i zawarty w niej moduł rekomendacyjny, powstała z wykorzystaniem Pythona [Python, 2022] – języka programowania wysokiego poziomu. Na wybór języka Python wpłynął fakt, iż posiada jedną z największych społeczności, a także stale rosnącą ilość bibliotek, co sprawia, że cechuje go duża wszechstronność. Jest to najczęściej wykorzystywany język do analizy danych oraz uczenia maszynowego, dzięki swojej prostej składni, uniwersalności (może działać na dowolnej platformie m.in. Windows, MacOS, Linux) oraz stabilnym bibliotekom i platformom programistycznym (ang. framework) z doskonałym wsparciem.

5.4.2. Pyrebase

Do komunikacji Pythona z zestawem narzędzi Firebase został użyty Pyrebase, [Pyrebase, 2022] Jest to biblioteka w języku Python dla Firebase API. Została napisana dla Pythona 3 i pozwala na manipulowanie stworzoną za pomocą Firebase’a bazą danych czasu rzeczywistego z poziomu kodu. Pyrebase umożliwia proste przeprowadzenie konfiguracji bazy oraz jej zainicjalizowanie, a także dostęp do wielu innych usług – m.in. uwierzytelniania użytkownika, dostępu i modyfikacji (operacje CRUD) bazy danych poprzez proste oraz złożone zapytania czy też możliwości korzystania z opcji przesyłania grafik do Firebase.

5.4.3. Kivy i KivyMD

Do utworzenia interfejsu użytkownika został wykorzystany otwarty framework języka programowania Python: Kivy [Kivy, 2022]. Kivy oferuje niestandardowy zestaw narzędzi, dzięki któremu można opracować przejrzysty, oryginalny, a także intuicyjny wygląd aplikacji. W dodatku ten framework jest wieloplatformowy, przez co kod może być użyty zarówno w aplikacjach na Androida, jak i na iOS, co w perspektywie rozwoju aplikacji zmniejsza w przyszłości ewentualne koszty oraz czas na przepisywanie programu. Ponadto, aby uatrakcyjnić interfejs, w aplikacji zostało użyte także rozszerzenie Kivy, czyli KivyMD [KivyMD, 2022]. KivyMD to zbiór tzw. widżetów Material Design, czyli prostego, czytelnego oraz łatwego do dostosowania stylu graficznego stworzonego przez Google, który łączy ze sobą innowacyjność i zasady dobrego projektowania [Tulibacka, A., 2022].

5.4.4. Pozostałe biblioteki

Pandas [Pandas, 2022] to biblioteka napisana dla języka Python, która najczęściej jest używana do uczenia maszynowego, a także zadań związanych z analizą danych. Cechuje ją bardzo dobra współpraca z wieloma innymi bibliotekami, które również zajmują się nauką o danych. W omawianej aplikacji została użyta podczas budowy modułu rekomendacyjnego, gdzie występowała duża ilość informacji pobieranych z bazy danych, umożliwiając przekonwertowanie danych w formacie JSON na obiekt.

Scikit-learn [Scikit-learn, 2022], znana również z nazwy sklearn, jest biblioteką, która zawiera klasyczne algorytmy klasyfikacji, regresji i klastrowania, a także wiele innych narzędzi uczenia maszynowego. Ona również została wykorzystana do budowy systemu rekomendacji, pozwalając np. przekształcić dane kompozycji perfum, czyli ciąg tekstowy, na wektor.

Googletrans [Googletrans, 2022] jest bezpłatną, a także nieograniczoną biblioteką Pythona, która zaimplementowała Google Translate API. Dzięki temu oferuje m.in. automatyczne wykrywanie języka, tłumaczenia zbiorcze czy też szybkie i niezawodne działanie poprzez korzystanie z tych samych serwerów, co popularny translate.google.com. Googletrans został użyty do przetłumaczenia specyfikacji (składu) perfum w omawianej aplikacji.

Requests [Requests, 2022] to prosta, lecz efektywna biblioteka, która służy do obsługi protokołu HTTP (wysyłania żądań, odbierania odpowiedzi itp.). W aplikacji ta biblioteka wykorzystywana jest do łapania wyjątków oraz obsługi błędów w module komunikacji programu z bazą danych Firebase.

5.4.5. Plik JSON

Podczas budowy aplikacji niejednemu raz pojawiają się dane przechowywane w plikach JSON [JSON, 2022]. Format JSON (ang. JavaScript Object Notation) jest otwartym formatem zapisu struktur danych. Często używa się go do szeregowania danych strukturalnych oraz przesyłania ich za pośrednictwem sieci. JSON wymaga mniej formatowania i jest popularną alternatywą dla XML [XML, 2023]. Dane w tym formacie są zapisywane w parach klucz-wartość.

6. System rekomendacji

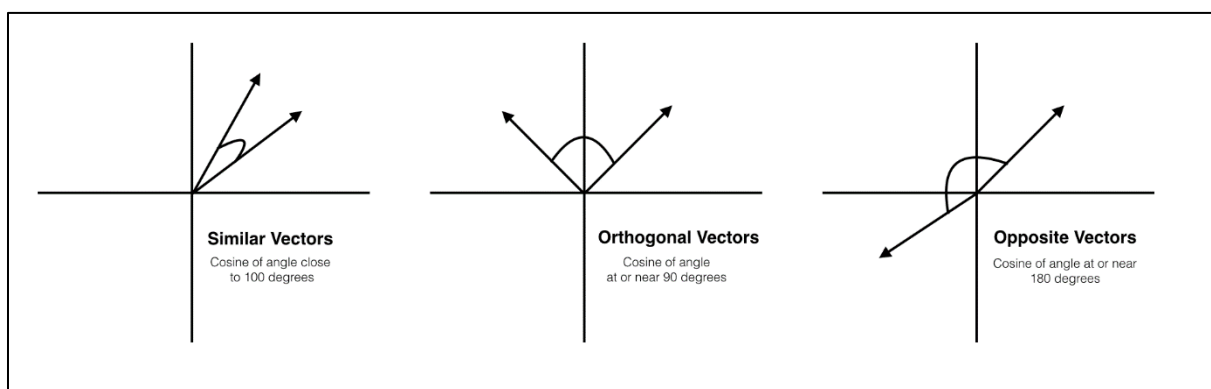
Rekomendacja perfum to główna funkcjonalność omawianej aplikacji. System rekomendacji został zaprojektowany, aby jak najlepiej dobierać pozycje zapachowe zgodnie z preferencjami użytkowników. W tym rozdziale zostanie opisany mechanizm cosine similarity, nazywany w pracy także podobieństwem cosinusowym, na którym bazuje system, realizowany w ramach niniejszej pracy. Autorka przedstawi również pseudokod, który w prosty sposób zaprezentuje działanie metody rekomendacyjnej.

6.1. Podobieństwo cosinusowe – cosine similarity

System rekomendacji zaprojektowano, używając metody opartej na treści, której sposób działania, a także zalety i wady, zostały dokładnie opisane w podrozdziale 2.2.1. – autorka doszła do wniosku, iż takie podejście będzie najrozsądniejsze, ponieważ znaleziony zestaw danych posiadał dostatecznie dobrze rozwiniętą specyfikację pozycji, aby móc łatwo manipulować jego zawartością.

Metoda oparta na treści może zostać stworzona poprzez użycie różnorodnych technik obliczeniowych, skupionych wokół porównywania danych. Jednakże wybrano podobieństwo cosinusowe, ponieważ jest niezwykle popularnym, optymalnym i chwalonym podejściem [Mohammad, S., 2022].

Podobieństwo cosinusowe to miara podobieństwa dwóch wektorów, wyliczona dzięki zmierzeniu kąta cosinus między tymi wektorami. Tę technikę wyróżnia uniwersalność i możliwość obsługi danych o zmiennej długości (w przeciwieństwie do odległości Hamminga [Odległość Hamminga, 2023]), dlatego też często jest wykorzystywana w różnych algorytmach uczenia maszynowego (np. w KNN, aby określić odległość między sąsiadami) [Darshan, M., 2023].



Rysunek 6.1 Cosine similarity
[źródło: Cosine Similarity 2023]

Na rysunku 6.1 przedstawiono graficzną reprezentację działania cosine similarity wraz z opisaniem trzech przypadków relacji między dwoma porównywanymi wektorami.

Podobieństwo rozpoczyna się od znalezienia cosinusa dwóch niezerowych wektorów. Obliczenia są możliwe, dzięki przekształceniu euklidesowego wzoru na iloczyn skalarny (widocznego na równaniu 1) do wzoru widocznego na równaniu 2. Zazwyczaj dane wyjściowe są generowane w przestrzeni dodatniej między granicami 0 i 1, gdzie 0 oznacza absolutny brak podobieństwa, a 1 całkowite podobieństwo.

$$A \cdot B = \|A\| \|B\| \cos \theta \quad (1)$$

$$\cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (2)$$

gdzie:

A i B to wektory, które są porównywane.

6.2. Pseudokod

W tym podrozdziale zostanie zaprezentowany pseudokod, przedstawiający metodę, która odpowiada za zaimplementowanie systemu rekomendacyjnego aplikacji.

Pseudokod jest wygodną formą przedstawienia fragmentów kodu oraz algorytmów w postaci tekstu informacyjnego. Nie posiada on składni, jaką narzucają języki programowania, przez co jest bezużyteczny podczas komplikacji, jednakże ma ogromne znaczenie dla ludzi, którzy w prosty sposób będą chcieli zrozumieć przedstawiony kawałek oprogramowania [Pseudokod, 2023].

metoda `rec_func` przyjmuje trzy argumenty:

- logiczny typ danych, odpowiadający za zdefiniowaną w preferencjach płeć
- nazwę perfumy zdefiniowanej w ankiecie preferencji
- listę, zawierającą wszystkie perfumy dostępne w bazie

początek metody `rec_func`

lista perfum zostaje przekonwertowana na JSON string

JSON string jest przekształcany na dwuwymiarową tabelaryczną strukturę danych (Pandas Dataframe)

jeżeli płeć równa się fałsz

zmiennej gender przypisujemy wartość: "Men"

w przeciwnym wypadku

zmiennej gender przypisujemy wartość: "Women"

zdefiniowanie listy nut zapachowych, do której przypisujemy odpowiednie nazwy kolumn

dla każdego elementu w zdefiniowanej liście nut zapachowych

wybierz z tabelarycznej struktury danych kolumnę o nazwie równej elementowi i zastąp wszystkie wiersze o wartości NULL pustym łańcuchem

początek metody combine_features

metoda przyjmuje wiersz pozycji perfumy, a potem łączy wszystkie pola z kolumn nut zapachowych

zwraca ciąg tekstowy, czyli jeden wiersz o wartości, zawierającej zawartość pól nut zapachowych

koniec metody combine_features

dodanie nowej kolumny (combined_features) do tabelarycznej struktury danych poprzez wywołanie metody combine_features dla każdego wiersza

utworzenie macierzy z kolumny combined_features

przypisanie do zmiennej cosine_sim macierzy z obliczonym podobieństwem cosinusowym między każdą pozycją

początek metody get_id_from_name

metoda zwraca indeks perfumy, używając dostarczonej nazwy pozycji

koniec metody get_id_from_name

przypisanie zmiennej perfume_id, za pomocą metody get_id_from_name, indeksu perfumy wybranej w ankiecie preferencji

wygenerowanie listy rekomendowanych pozycji do perfumy o indeksie zawartym w zmiennej `perfume_id`

posortowanie malejąco listy rekomendowanych pozycji po współczynniku zgodności

`początek metody get_title_from_index`

metoda przyjmuje indeks perfumy

jeżeli klasyfikacja płciowa perfumy jest równa zmiennej `gender`:

zwracana jest jej nazwa

jeżeli perfuma tym indeksem jest oznaczona `unisex`:

zwracana jest jej nazwa

w innym wypadku

zwracany jest wartość pusta

`koniec metody get_title_from_index`

inicjalizacja pustej listy `title_list`

dla każdej perfumy w posortowanej liście rekomendacyjnej

przypisanie do zmiennej `title` jej nazwy poprzez użycie metody `get_title_from_index`

przypisanie do zmiennej `perf_factor` współczynnika podobieństwa

jeżeli tytuł nie równa się wartości pustej

zmienna `title` i `perf_factor` dodawane są do `title_list`

inicjalizujemy obiekt klasy `Set`: `seen`, w którym przechowane zostaną raz pozycje listy

inicjalizujemy pustą listę: `result`, w której umieszczone zostaną niezduplikowane pozycje

dla każdej pozycji z listy `title_list`

jeżeli pierwszego elementu pozycji nie znaleziono w obiekcie `seen`:

pierwszy element pozycji dodany zostanie do obiektu `seen`

cała pozycja zostanie dodana do listy `result`

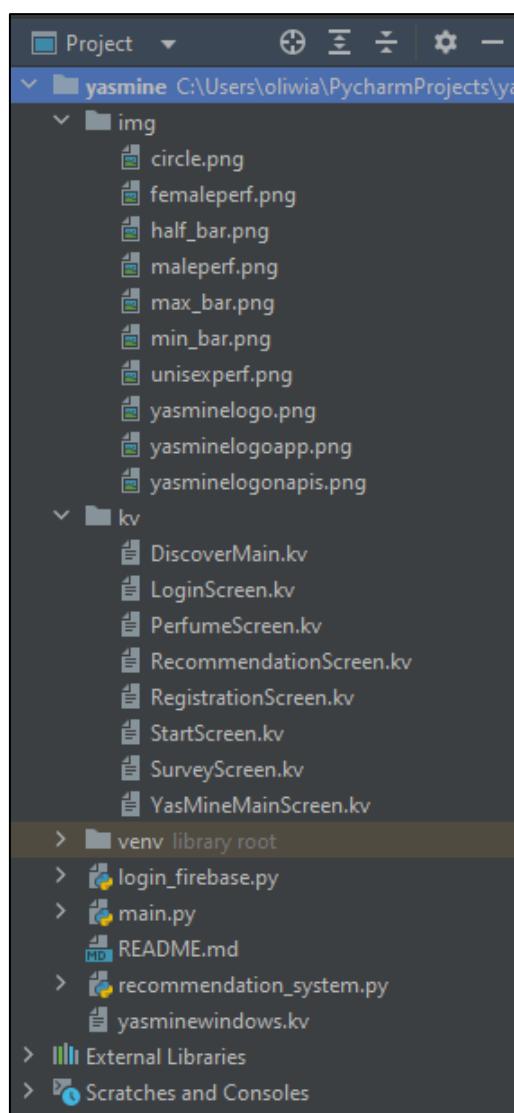
```
metoda rec_func zwraca listę result, zawierającą posortowaną listę  
pozycji rekomendowanych  
koniec metody rec_func
```

Listing 6.1 Pseudokod metody rekomendacyjnej rec_func

Na listingu 6.1 został zaprezentowany pseudokod metody `rec_func`, która zawiera w sobie całą logikę systemu rekomendacyjnego aplikacji.

7. Implementacja

W tym rozdziale ukazano implementację opracowywanej aplikacji, czyli omówiono jej działanie oraz przedstawiono kod. Rozdział podzielono na dwa podrozdziały – w pierwszym omawiany jest frontend aplikacji, czyli interfejs i komunikacja z nim. W drugim zaprezentowany zostanie backend, którym jest logika oprogramowania, wraz z najważniejszymi segmentami kodu aplikacji [Peszka, D., 2022].



*Rysunek 7.1 Struktura projektu aplikacji
[źródło na podstawie: : PyCharm 2022]*

Na rysunku 7.1 ukazano widok struktury projektu aplikacji YasMine w środowisku PyCharm. W przedstawionym katalogu widzimy trzy główne foldery oraz pozostałe pliki, które wchodzi w skład budowy aplikacji.

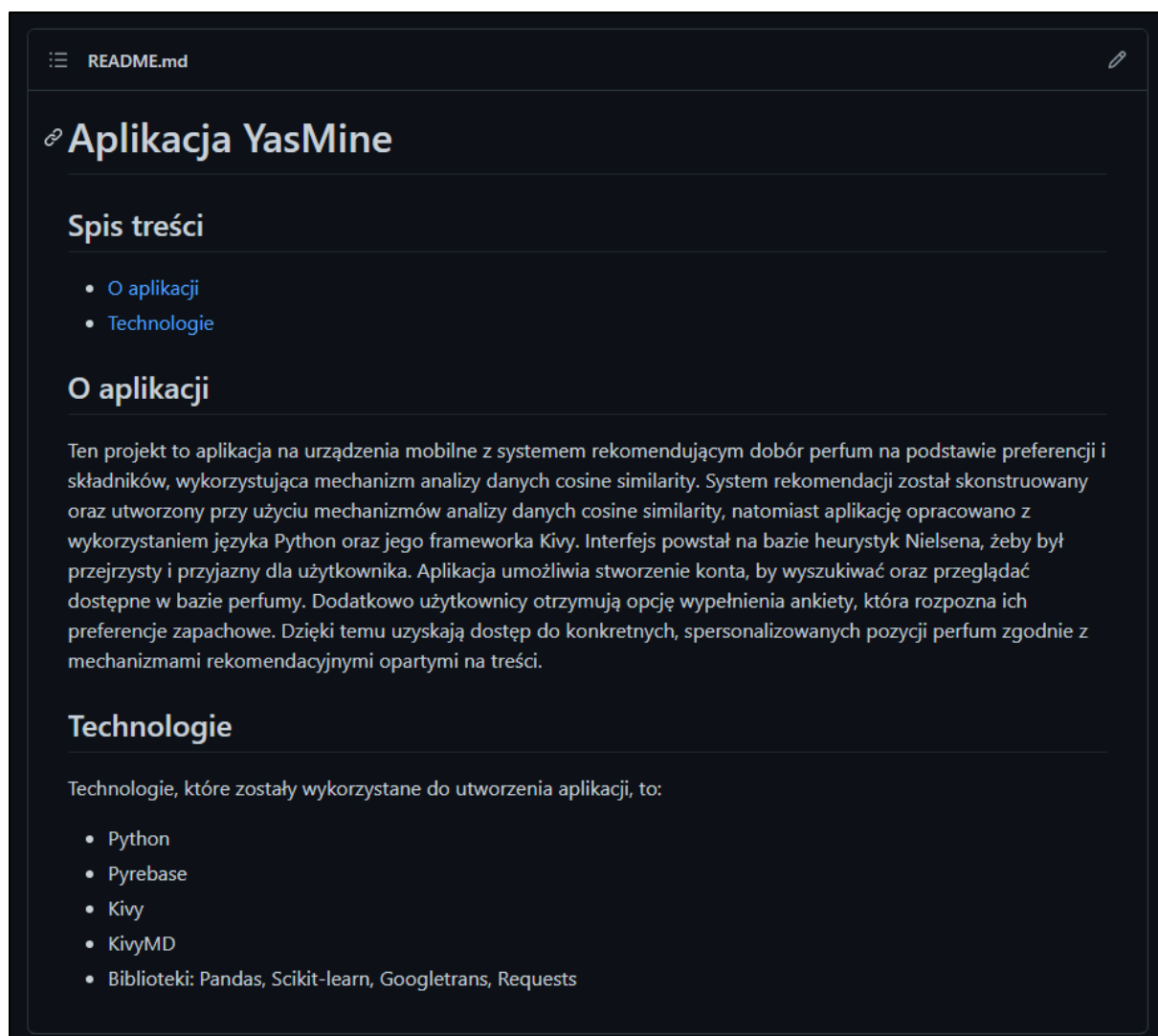
Pierwszy folder `img` składa się ze wszystkich grafik, jakie zostały wykorzystane w programie, czyli m.in. ikon perfum, sygnetu i logo aplikacji.

Drugi folder kv jest wypełniony wszystkimi plikami z rozszerzeniem .kv, które odpowiadają za przechowywanie zdefiniowanych w nich widżetów. Język Kivy, umożliwiając tworzenie plików właśnie o formacie KV, ułatwia oddzielenie logiki aplikacji od interfejsu użytkownika, przez co projekt pozostaje spójnie ustrukturyzowany. Każdy z plików odnosi się do osobnego ekranu np. `LoginScreen.kv` to interfejs ekranu logowania.

Ostatni folder to `venv`, czyli Python Virtual Environment [Python Virtual Environment, 2022], który jest odseparowanym oraz całkowicie niezależnym od głównej instalacji Pythona środowiskiem. Każdy tworzony projekt powinien posiadać własne środowisko, ponieważ dzięki temu będzie się składać z unikatowego zestawu pakietów specjalnie dostosowanego do danej aplikacji.

Pliki, które również znajdują się w katalogu, to:

- Plik `main.py` jest rdzeniem (głównym elementem projektu), ponieważ to w nim zostaje zainicjowana oraz uruchomiona aplikacja. Dodatkowo również tam zostały utworzone wszystkie klasy oraz metody, które obsługują widżety stworzone w plikach z folderu kv.
- Plik `login_firebase.py` zawiera w sobie konfigurację połączenia z bazą, a także metody, jakie odpowiadają za operacje związane z komunikacją z bazą danych, czyli np. logowanie, rejestracja czy podstawowe zapytania.
- Plik `recommendation_system.py` odpowiada za przechowywanie klasy, która posiada statyczną metodę do polecania pozycji perfum na podstawie preferencji użytkownika – w nim mieści się logika modułu rekomendacyjnego całej aplikacji.
- Plik `yasminewindows.kv` zawiera w sobie wszystkie pliki .kv z folderu kv, umożliwiając późniejsze załadowanie ich w `main.py` z pojedynczego pliku, sprawiając, iż kod jest czytelny oraz uporządkowany.
- Plik `README.md` odpowiada za krótki opis aplikacji YasMine, a jego zawartość pokazana jest na rysunku 7.2. Znajduje się on na platformie GitHub, aby osoby, które będą chciały poznać projekt, miały możliwość, by się z nim zapoznać (oczywiście po zmianie repozytorium z prywatnego na publiczne).



*Rysunek 7.2 Plik README.md
[źródło na podstawie: GitHub 2022]*

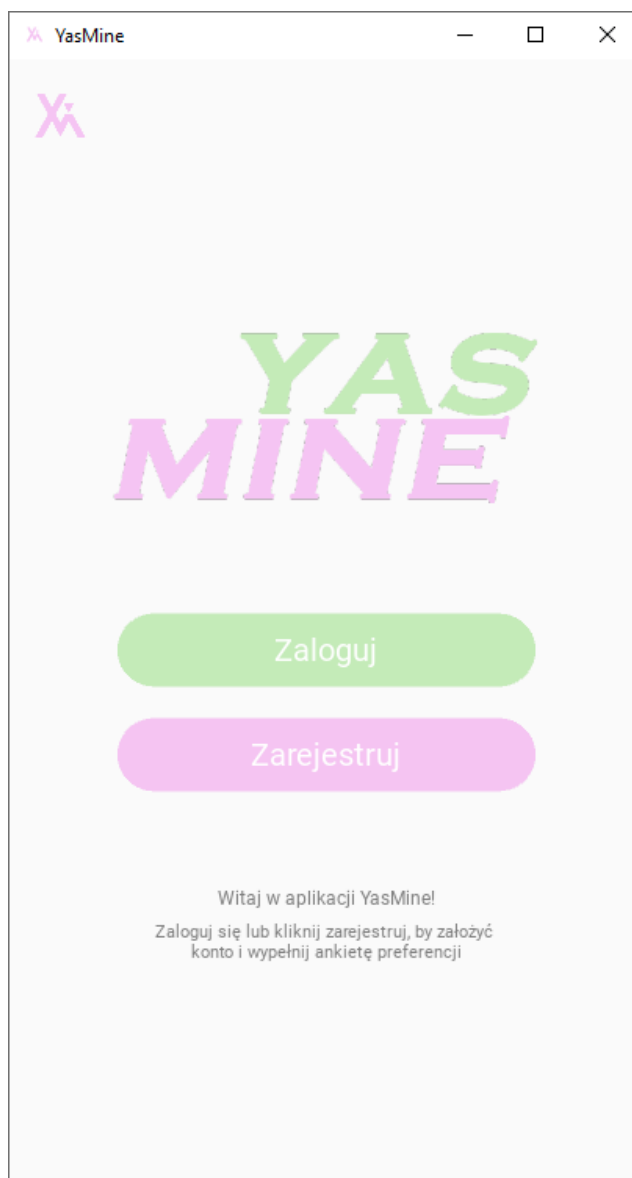
7.1. Frontend

W tym podrozdziale został omówiony interfejs aplikacji, czyli wygląd oraz sposób korzystania z niej. Do utworzenia interfejsu wykorzystano framework Kivy i KivyMD. Jego forma została utworzona poprzez wspomaganie się zasadami uniwersalnego projektowania aplikacji.

Kolory, które zostały użyte w aplikacji, to pastelowy róż, zieleń oraz niebieski – traktowane są jako trzy kolory odpowiadające za akcenty na stronie. Zostały w ten sposób dobrane, ponieważ delikatne barwy, czyli te o mniejszym nasyceniu, działają na użytkowników uspokajająco. Zauważono również pozytywny wpływ pastelowych kolorów na poprawę nastroju użytkowników [Colour Psychology, 2022]. Pozostałe barwy to głównie biały, który użyty został do zaznaczenia wolnej przestrzeni w aplikacji oraz jako kolor czcionki i szary, jaki odpowiada wyłącznie za kolor tekstu.

7.1.1. Ekran startowy

Ekran startowy (na rys. 7.3) jest pierwszym ekranem, który niezalogowany użytkownik widzi po włączeniu aplikacji. Dodatkowo już podczas pierwszej interakcji ukazuje on kolory, które będą dominować w wyglądzie systemu. Posiada pięć ważnych elementów, oddzielonych znacznymi odstępami, aby uniknąć chaosu w odbiorze treści.

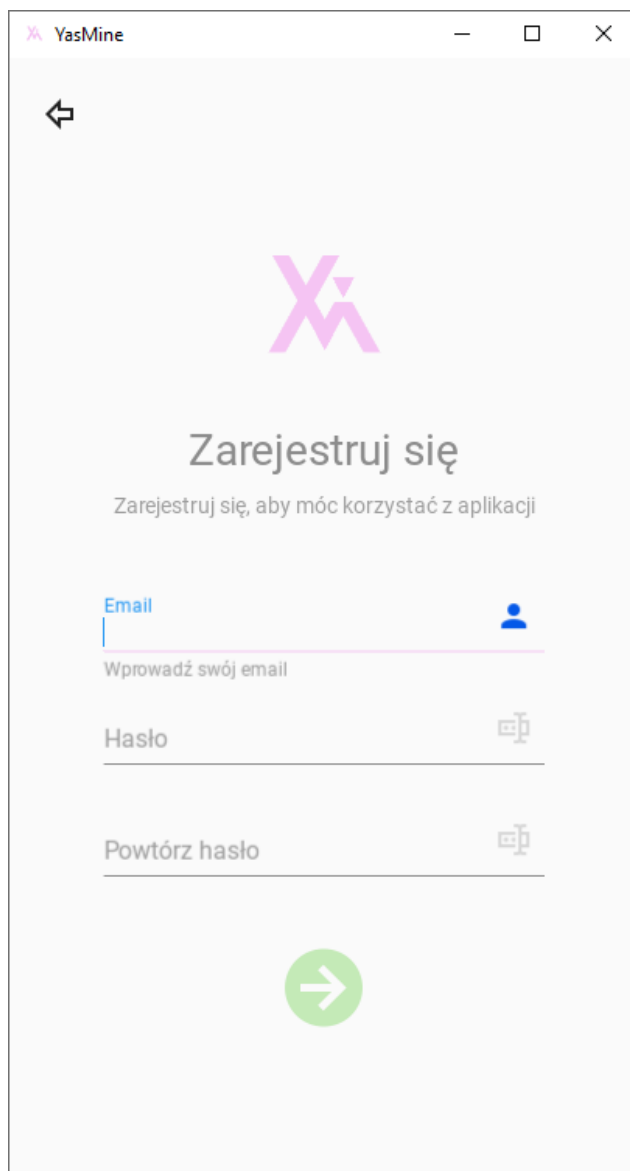


Rysunek 7.3 Ekran startowy aplikacji
[źródło: opracowanie własne]

W lewym górnym rogu umieszczony jest sygnet, a pod nim wyśrodkowany logotyp, przez co użytkownik już na starcie zostaje poinformowany jak prezentuje się logo aplikacji. Niżej istnieją dwa przyciski: zaloguj oraz zarejestruj – stanowią one odnośniki do innych ekranów, na których będzie można wykonać operacje związane z kontem użytkownika. Dodatkowo na samym dole jest krótka informacja dla osób korzystających pierwszy raz z aplikacji ze stosownymi podpowiedziami, co należy zrobić, aby uzyskać upragniony efekt.

7.1.2. Ekran rejestracji

Ekran rejestracji (na rys. 7.4), do którego można się dostać z ekranu startowego, a także ekranu logowania, jest miejscem, w jakim niezalogowany użytkownik ma możliwość założenia konta.



*Rysunek 7.4 Ekran rejestracji aplikacji
[źródło: opracowanie własne]*

Pod sygnetem aplikacji znajduje się napis „Zarejestruj się” oraz krótka informacja dla korzystającej z rejestracji osoby, jaka oznajmia przeznaczenie ekranu. Pod nimi są trzy pola, które trzeba wypełnić, aby utworzyć konto. Każde z nich ma odpowiednią podpowiedź, co należy wpisać wewnątrz, żeby zakończyć proces sukcesem. Każde pole podświetla się na niebiesko, gdy z niego korzystamy. Dodatkowo z ekranu rejestracji można znów przenieść się do ekranu startowego, korzystając ze strzałki w lewym górnym rogu.

Ograniczone zaufanie do użytkownika niosło za sobą potrzebę wprowadzenia odpowiednich zabezpieczeń. Są one wyświetlane jako czerwony tekst pod przyciskiem strzałki, który odpowiada za rejestrację.

Pierwsze z nich, zaprezentowane na rys. 7.5, dotyczy formy, jaką jest email. Osoba go wprowadzająca, nie może wpisać tam tekstu, który nie przypomina formatem emaila. Drugie zabezpieczenie odnosi się do samego hasła, jakie musi mieć co najmniej sześć znaków – przykład komunikatu ukazano na rysunku 7.6.

The screenshot shows the registration screen of the YasMine application. At the top, there is a back arrow icon and the YasMine logo. Below the logo, the text "Zarejestruj się" is displayed, followed by the subtitle "Zarejestruj się, aby móc korzystać z aplikacji". There are three input fields: "Email" with the value "xyz", "Hasło" with the value "*****", and "Powtórz hasło" with the value "*****". Each field has a placeholder text "Wprowadź swój email" or "Wprowadź swoje hasło (min. 6 znaków)". Below the input fields, there is a green circular button with a white right-pointing arrow. At the bottom, a red error message "Nieprawidłowy email!" is displayed.

Rysunek 7.5 Ekran rejestracji – błąd formatu email
[źródło: opracowanie własne]

The screenshot shows the registration screen of the YasMine application. At the top, there is a back arrow icon and the YasMine logo. Below the logo, the text "Zarejestruj się" is displayed, followed by the subtitle "Zarejestruj się, aby móc korzystać z aplikacji". There are three input fields: "Email" with the value "oliwia@gmail.com", "Hasło" with the value "****", and "Powtórz hasło" with the value "****". Each field has a placeholder text "Wprowadź swój email" or "Wprowadź swoje hasło (min. 6 znaków)". Below the input fields, there is a green circular button with a white right-pointing arrow. At the bottom, a red error message "Słabe hasło!" is displayed.

Rysunek 7.6 Ekran rejestracji – błąd siły hasła
[źródło: opracowanie własne]

Trzecie zabezpieczenie dotyczy dwóch ostatnich pól, w których hasła muszą być zapisane jednakowo, aby założyć konto (na rys. 7.7). Natomiast czwarte, przedstawione na rysunku 7.8, odnosi się do sprawdzenia czy w systemie nie istnieje już użytkownik o tym samym emailu.

The screenshot shows the registration screen of the YasMine application. At the top, there is a back arrow icon. Below it is the YasMine logo. The title "Zarejestruj się" is centered, followed by the subtitle "Zarejestruj się, aby móc korzystać z aplikacji". The form contains three input fields: "Email" with the value "oliwia@gmail.com", "Hasło" with "*****", and "Powtórz hasło" with "*****". Below the password fields is a green arrow button. At the bottom, a red error message states: "Wpisane hasła nie są identyczne!".

*Rysunek 7.7 Ekran rejestracji – błąd zgodności haseł
[źródło: opracowanie własne]*

The screenshot shows the registration screen of the YasMine application, identical in layout to the previous one. However, the red error message at the bottom states: "Użytkownik z takim email'em już istnieje!".

*Rysunek 7.8 Ekran rejestracji – błąd zgodności emaila
[źródło: opracowanie własne]*

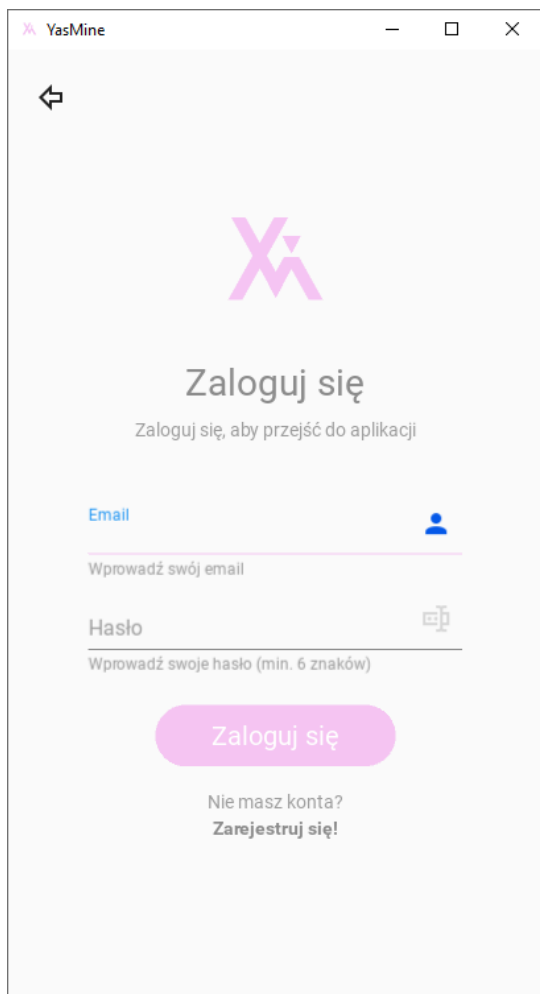
7.1.3. Ekran logowania

Ekran logowania (na rys. 7.9), do którego można się dostać poprzez ekran startowy, jest miejscem, w którym niezalogowany użytkownik ma możliwość zalogowania się do wcześniej utworzonego konta.

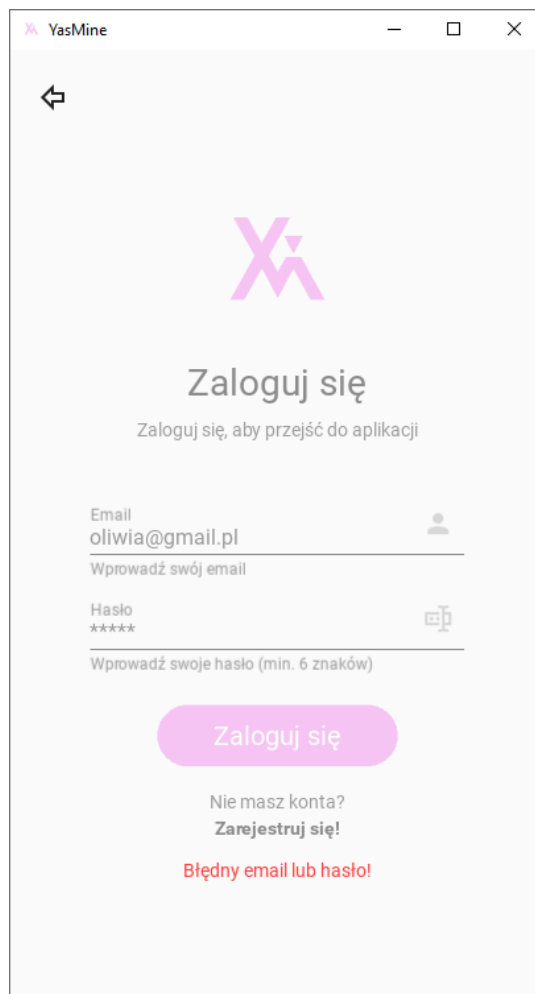
Wygląda analogicznie do ekranu rejestracji, przez jaki każda osoba posiadająca konto musiała przejść, aby móc się zalogować, co od razu ułatwia poruszanie się po nim – nawet sygnet oraz napis „Zaloguj się” wraz z krótką informacją jest podobnie ułożony. Ekran posiada dwa pola, jakie wymagają emailu oraz hasła, aby się zalogować, a pod nimi znajduje się przycisk, odpowiadający za logowanie. Następnie widoczna jest odpowiedź, dotycząca sytuacji, w której użytkownik nie ma konta, a chciałby się dostać do aplikacji. Znajduje się tam pogrubiony tekst, dotyczący rejestracji, jaki od razu po naciśnięciu przekierowuje osobę do ekranu rejestracji, pozwalając założyć konto.

Ten ekran również posiada zabezpieczenia, zaprezentowane na rysunku 7.10, w kolorze czerwonym, wyświetlane na samym dole strony w momencie niepoprawnej próby zalogowania

się. Po wpisaniu złego adresu email lub hasła, użytkownik dostanie informację zwrotną, iż popełnił błąd, przez co będzie zmuszony na nowo uzupełnić dane oraz ponowić logowanie.



*Rysunek 7.9 Ekran logowania aplikacji
[źródło: opracowanie własne]*

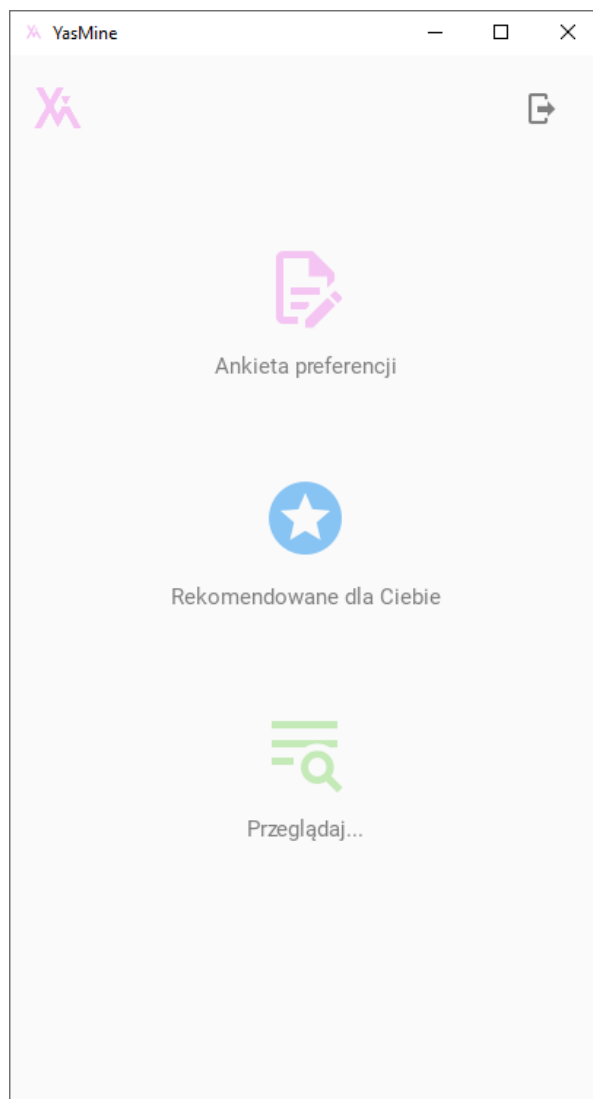


*Rysunek 7.10 Ekran rejestracji – komunikat błędu
[źródło: opracowanie własne]*

7.1.4. Ekran główny

Ekran główny (na rys. 7.11) to centrum aplikacji. Pojawia się od razu po zakończonej rejestracji oraz pomyślnym logowaniu i przedstawia wszystkie moduły, do jakich zalogowany użytkownik ma dostęp.

Na górze ekranu po lewej stronie umieszczony jest sygnet aplikacji, a po prawej znajduje się przycisk, którego naciśnięcie umożliwia wylogowanie z konta i ponowne przejście na ekran startowy. Resztę przestrzeni pokrywają przyciski oraz tekst, jaki je odpowiednio opisuje. Każdy z nich posiada unikatowy kolor, dobrany ze wcześniej opisanej palety barw. Po naciśnięciu różowej ikony podpisanej „Ankieta preferencji”, użytkownik zostaje przekierowany do modułu ankiety. Kliknięcie w niebieską ikonę „Rekomendowane dla Ciebie” prowadzi do modułu rekomendacyjnego. Natomiast naciśnięcie zielonej ikonki „Przeglądaj” kieruje do modułu wyszukiwania.

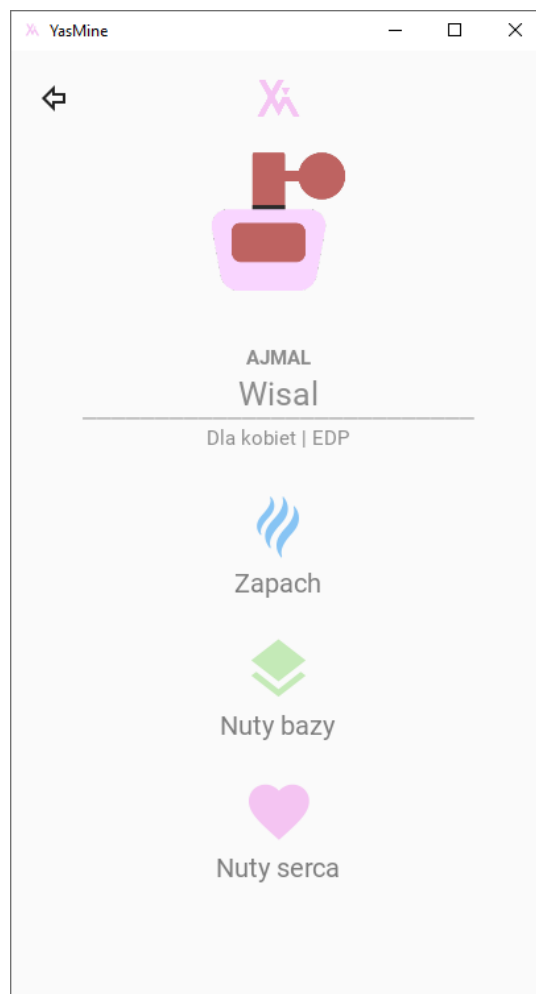


*Rysunek 7.11 Ekran główny aplikacji
[źródło: opracowanie własne]*

7.1.5. Ekran pozycji perfumy

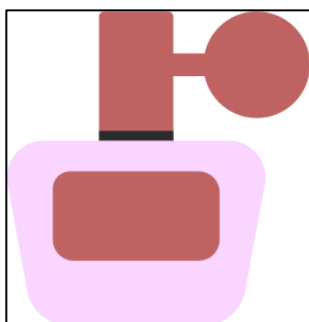
Ekran pozycji perfumy (na rys. 7.12) to miejsce, w którym użytkownik jest w stanie odczytać specyfikację wybranego produktu. Do tego ekranu można się dostać poprzez moduł rekomendacyjny, jeżeli kliknie się w polecane perfumy, a także przez moduł wyszukiwania jak wybierze się daną pozycję z listy.

Na górze ekranu, po lewej stronie, znajduje się przycisk strzałki i odpowiada on za wycofywanie się z pozycji, przekierowując do poprzedniego modułu. Na tej samej wysokości istnieje sygnet aplikacji, a pod nim ikonka perfumy, która jest unikalna dla trzech klasyfikacji płciowych.

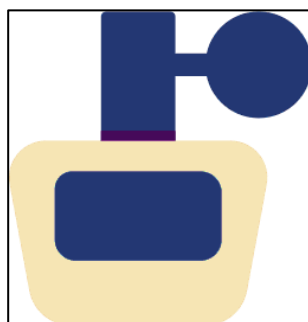


*Rysunek 7.12 Ekran pozycji perfumy aplikacji
[źródło: opracowanie własne]*

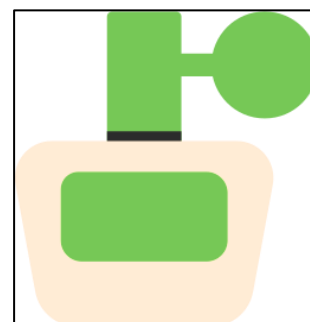
Rysunki 7.13, 7.14 i 7.15 prezentują wszystkie dostępne ikony perfum, które wyświetlają się na samej górze ekranu, symbolizując przeznaczenie płciowej danej pozycji.



*Rysunek 7.13 Ikona, reprezentująca
perfumy dla kobiet
[źródło: opracowanie własne]*



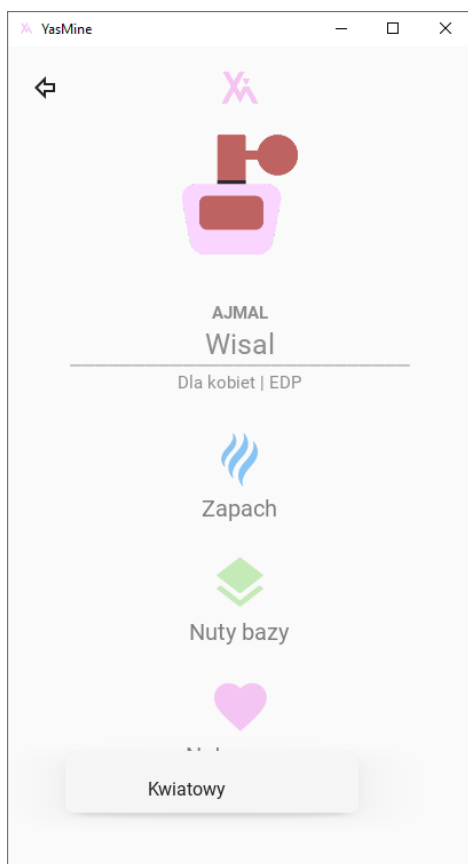
*Rysunek 7.14 Ikona, reprezentująca
perfumy dla mężczyzn
[źródło: opracowanie własne]*



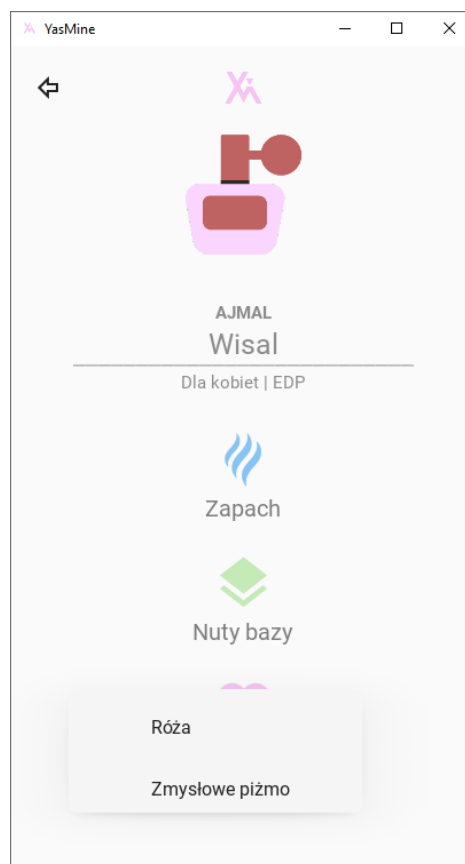
*Rysunek 7.15 Ikona, reprezentująca
perfumy unisex
[źródło: opracowanie własne]*

Pod ikoną jest część specyfikacji w formie opisowej. To w tym miejscu znajdują się informacje na temat marki, nazwy perfumy, dla kogo została stworzona oraz jej stężenia. Jest to niezbędna dla użytkownika treść, która pozwoli mu na dokładniejsze zapoznanie się z wybraną pozycją.

Następnie ukazane są trzy przyciski w wybranych podczas projektowania aplikacji, pastelowych kolorach. Po naciśnięciu każdego z nich, na ekranie pojawi się okienko z informacjami, powiązanymi z każdą z opcji. Na rysunku 7.16 ukazany jest interfejs po naciśnięciu pola „Zapach”. Natomiast na rysunku 7.17 przedstawiony został widok aplikacji po naciśnięciu „Nuty bazy”. Analogicznie działa przycisk „Nuty serca”. Dzięki skondensowaniu tych wszystkich danych w jedno kliknięcie, użytkownik uzyskał przejrzysty interfejs o prostej obsłudze.



Rysunek 7.16 Wyświetlanie zapachu perfumy
[źródło: opracowanie własne]



Rysunek 7.17 Wyświetlanie nut bazy perfumy
[źródło: opracowanie własne]

7.1.6. Moduł ankiety

Moduł ankiety składa się z ekranu zatytułowanego „Ankieta preferencji” (na rys. 7.18), w którym użytkownik dostaje możliwość wypełnienia ankiety, dzięki czemu moduł rekomendacji będzie mógł polecać mu perfumy na podstawie dokonanych wyborów.

Ankieta zawiera tylko dwa pytania. Pierwsze z nich to wybór płci, który w przyszłości ograniczy prezentowane perfumy tylko do wybranej opcji. Warto wspomnieć, że nieważne, co

wyberzemy, perfumy typu unisex zawsze będą dołączane do listy polecanych pozycji, ponieważ, tak jak ich nazwa wskazuje – są dla każdego.

Na drugie pytanie składają się dwa pola tekstowe. Należy do nich wpisać perfumy, na podstawie których system rekomendacyjny poleci użytkownikowi nowe produkty. Pola można pozostawić puste, jednakże będzie się to wiązało z ograniczeniem funkcjonalności rekomendacji.

Rysunek 7.18 Ekran modułu ankiety
[źródło: opracowanie własne]

Forma ankiety (zwłaszcza drugiego pytania), jak już wspomniano w wymaganiach niefunkcjonalnych, specjalnie została utworzona z myślą o użytkownikach, którzy nie znają się na składnikach kompozycji – nie wiedzą jak pachną. Trudność w wyobrażeniu zapachu, skłoniła autorkę pracy do znalezienia optymalnego sposobu na zdefiniowanie preferencji. Tym właśnie rozwiązaniem stało się pytanie, wyłaniające dwie znane użytkownikowi pozycje, na bazie jakich system będzie mógł przeanalizować konkretne nuty zapachowe, a co za tym idzie polecić odpowiednie perfumy.

Naciskając na niebieski przycisk, użytkownik zapisuje wszystkie odpowiedzi ankiety jako swoje preferencje i zostaje automatycznie przekierowany do modułu rekomendacyjnego, gdzie będzie mógł poznać polecane perfumy.

Moduł ankiety również został zabezpieczony przed ewentualnymi błędami, związanymi z brakiem wpisanej pozycji w bazie danych lub niepoprawną pisownią. Zostaje wtedy wyświetlona stosowna informacja w kolorze czerwonym, która uniemożliwia zapisanie źle wypełnionej ankiety. Na rysunku 7.19 i 7.20 zostały zaprezentowane komunikaty, występujące podczas niepoprawnego uzupełnienia ankiety.

YasMine

←

Xi

Ankieta preferencji

Wybierz płć


☒ Kobieta ☐ Mężczyzna

Wybierz perfumy

1. Pozycja zapachowa
XYZ
Wpisz pierwszą pozycję

2. Pozycja zapachowa
Red
Wpisz drugą pozycję

Dla 1: Błąd w pisowni lub brak perfumy w bazie


Zapisz i odkryj
nowe zapachy

Rysunek 7.19 Komunikat błędu dla pierwszej pozycji zapachowej
[źródło: opracowanie własne]

YasMine

←

Xi

Ankieta preferencji

Wybierz płć


☒ Kobieta ☐ Mężczyzna

Wybierz perfumy

1. Pozycja zapachowa
XYZ
Wpisz pierwszą pozycję

2. Pozycja zapachowa
ZYX
Wpisz drugą pozycję

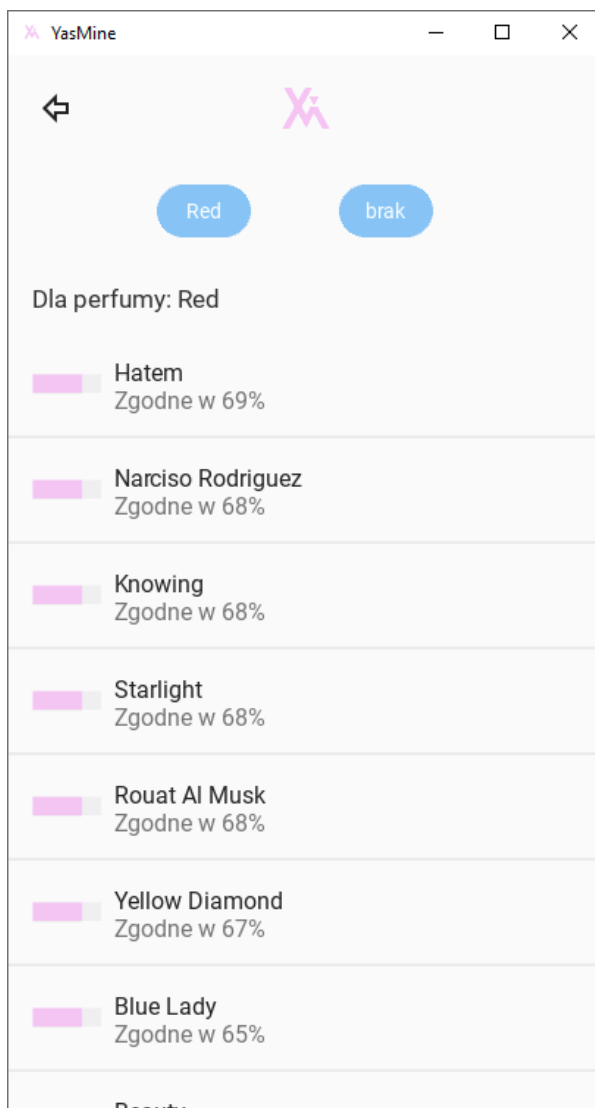
Dla 1 i 2: Błąd w pisowni lub brak perfumy w bazie


Zapisz i odkryj
nowe zapachy

Rysunek 7.20 Komunikat błędu dla pierwszej i drugiej pozycji zapachowej
[źródło: opracowanie własne]

7.1.7. Moduł rekomendacyjny

Moduł rekomendacyjny składa się z ekranu (na rys. 7.21), który pozwala użytkownikowi zapoznać się z wygenerowanymi specjalnie dla niego, rekomendowanymi pozycjami.

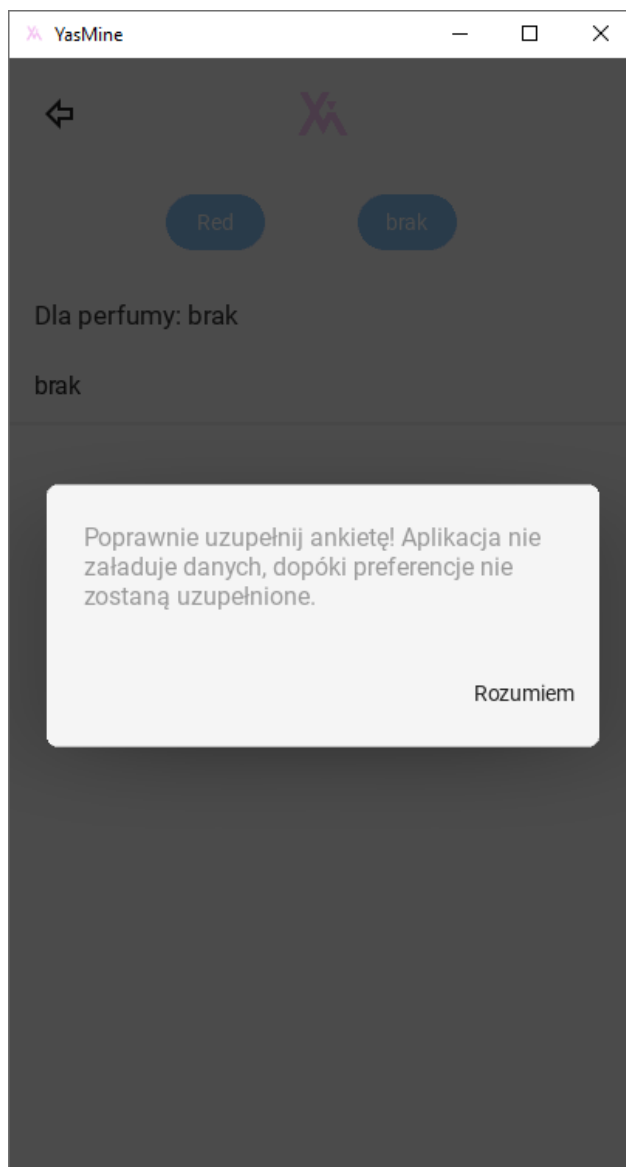


*Rysunek 7.21 Ekran modułu rekomendacyjnego
[źródło: opracowanie własne]*

Na górze ekranu po lewej stronie znajduje się strzałka odpowiedzialna za powrót do ekranu głównego, a na środku ukazano sygnet aplikacji. Niżej są dwa przyciski, opisane nazwami perfum, jakie zostały zapisane w ankiecie preferencji – to one umożliwiają poruszanie się między listami rekomendacyjnymi. Dla każdej perfumy jest osobna lista, na której widać procent zgodności danej pozycji z perfumami wzorcowymi (tymi z ankiety) w formie paska postępu oraz procentu.

Kliknięcie w element listy, przenosi użytkownika do ekranu pozycji perfumy, gdzie może się on zapoznać z pełną specyfikacją zapachu.

Dodatkowo, kiedy ankieta preferencji nie zostanie uzupełniona lub będzie niepełna, moduł rekomendacyjny oznajmia użytkownikowi, czemu nie otrzymał dostępu do pełnej funkcjonalności modułu poprzez informację w wyskakującym okienku. Omawiane zabezpieczenie zostało zaprezentowane na rysunku 7.22.



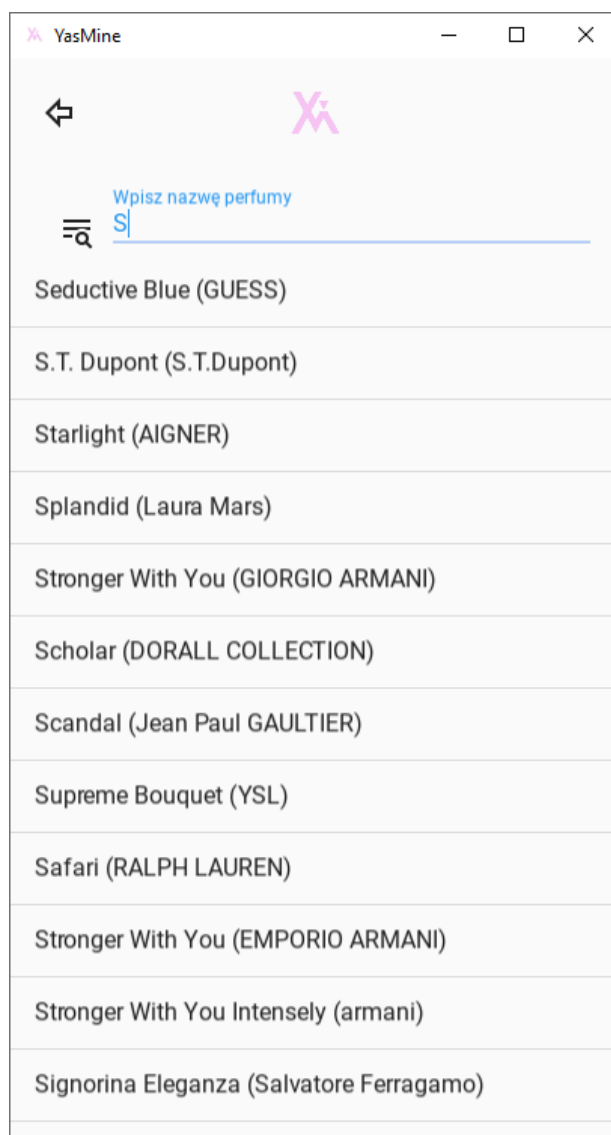
*Rysunek 7.22 Komunikat o błędzie w module rekomendacyjnym
[źródło: opracowanie własne]*

7.1.8. Moduł wyszukiwania

Moduł wyszukiwania składa się z ekranu (na rys. 7.23), jaki umożliwia użytkownikowi wyszukanie odpowiedniej pozycji w bazie danych po nazwie perfumy.

Na ekranie na samej górze po lewej stronie, znajduje się strzałka, umożliwiająca powrót do ekranu głównego, a na środku widnieje sygnet aplikacji. Pod nim jest najważniejszy element tego modułu, czyli pole odpowiedzialne za wyszukiwanie produktów po ich nazwie. Gdy użytkownik wchodzi z nim w interakcję, ono podświetla się na niebiesko.

Sam proces wyszukiwania jest prosty – do pola wprowadza się cyfry lub litery o dowolnej wielkości, a aplikacja prezentuje wszystkie pozycje, jakie zaczynają się na wpisaną przez użytkownika frazę. Kiedy z wyszukiwarki zostanie usunięta wartość, pojawia się lista wszystkich dostępnych w bazie perfum. Naciśnięcie na wybraną pozycję, przekierowuje osobę do ekranu pozycji perfumy.



*Rysunek 7.23 Ekran modułu wyszukiwania
[źródło: opracowanie własne]*

7.2. Backend

W tym podrozdziale zostanie przedstawiona logika aplikacji, czyli wybrane funkcjonalności wraz z kodem.

7.2.1. Logowanie i rejestracja

Operacje związane z kontem użytkownika zostały zdefiniowane w pliku `login_firebase.py`, który stanowi część aplikacji odpowiedzialną za komunikację z bazą danych.

Na listingu 7.1 przedstawiona została pełna konfiguracja połączenia z bazą z poziomu kodu przy użyciu technologii Pyrebase. Dodatkowo pod konfiguracją zostały zainicjalizowane dwie usługi Firebase: baza danych oraz uwierzytelnianie, dzięki którym mamy pełen dostęp do funkcjonalności, które pozwolą na połączenie się aplikacji z bazą danych czasu rzeczywistego.

```
firebaseConfig = {
    'apiKey': "AIzaSyB8m_z6nVqJNy-p4s8aagnxQA-4xctsh5Q",
    'authDomain': "yasmine-17.firebaseio.com",
    'projectId': "yasmine-17",
    'storageBucket': "yasmine-17.appspot.com",
    'messagingSenderId': "920360490783",
    'appId': "1:920360490783:web:7f468a2326f11a7c62822a",
    'measurementId': "G-10HWJTC42T",
    'databaseURL': "https://yasmine-17-default-rtdb.firebaseio.com/"
}

firebase = pyrebase.initialize_app(firebaseConfig)
auth = firebase.auth()
db = firebase.database()
```

Listing 7.1 Konfiguracja połączenia z bazą

Listing 7.2 prezentuje metodę `registration`, jaka używa zdefiniowanego na listingu 7.1 uwierzytelniania, aby umożliwić rejestrację nowego konta. Użytkownik może stworzyć konto, gdy poda odpowiedni email, a także hasło poprzez metodę `create_user_with_email_and_password`. Następnie używana jest metoda `sign_in_with_email_and_password`, aby zalogować się do konta i otrzymać aktualny token (tzw. żeton) oraz lokalny indeks użytkownika, jakie wykorzystuje się, aby dodać do obiektu `users` w bazie danych nowego użytkownika o specyfikacji, którą definiujemy w obiekcie `new_user`.

Metoda `change_screen` przekierowuje zalogowanego użytkownika do ekranu głównego. Pod nią znajdują się obsłużone wyjątki w momencie popełnienia przez użytkownika błędu w czasie rejestracji.

```
def registration(email, password, rep_pass):
    global local_id
    global token_id
```



```

if password == rep_pass:
    try:
        auth.create_user_with_email_and_password(email, password)
        new_user = {
            'email': email,
            'login_date': date.today().strftime("%m/%d/%Y"),
            'is_active': True
        }
        auth_token = auth.sign_in_with_email_and_password(email, password)
        token_id = auth_token['idToken']
        local_id = auth_token['localId']
        db.child('users').child(local_id).set(new_user, token_id)
        App.get_running_app().change_screen("yasminemain")

    except requests.exceptions.HTTPError as e:
        error_json = e.args[1]
        error = json.loads(error_json)['error']['message']
        if error == "EMAIL_EXISTS":
            App.get_running_app().root.ids['registration'].ids[
                'reg_error_label'].text = "Użytkownik z takim emailem już
istnieje!"
        if error == "WEAK_PASSWORD : Password should be at least 6
characters":
            App.get_running_app().root.ids['registration'].ids['reg_error_label'].text =
"Słabe hasło!"
        if error == "INVALID_EMAIL":
            App.get_running_app().root.ids['registration'].ids[
                'reg_error_label'].text = "Nieprawidłowy email!"
    else:
        App.get_running_app().root.ids['registration'].ids[
            'reg_error_label'].text = "Wpisane hasła nie są identyczne!"

pass

```

Listing 7.2 Metoda registration

Na listingu 7.3 znajduje się metoda `login`, dzięki której użytkownik jest się w stanie zalogować do wcześniej utworzonego konta. Logowanie następuje analogicznie jak w metodzie `registration`, również w tym miejscu zostają przypisane do zmiennych globalnych token oraz indeks lokalny, aby w trakcie aplikacji użytkownik mógł czytać oraz wpisywać dane do bazy. Metoda `change_screen` ponownie przenosi zalogowanego użytkownika do ekranu głównego, umożliwiając dostęp do innych funkcjonalności aplikacji.

```

def login(email, password):
    global local_id
    global token_id
    try:
        user = auth.sign_in_with_email_and_password(email, password)
        token_id = user['idToken']
        local_id = user['localId']
        App.get_running_app().change_screen("yasminemain")

    except:
        App.get_running_app().root.ids['login'].ids['login_error_label'].text =

```

```
"Błędny email lub hasło!"  
pass
```

Listing 7.3 Metoda login

Możliwość wylogowania z konta obsługuje prosta metoda `logout`, widoczna na listingu 7.4, której logika skupia się na wyczyszczeniu danych uwierzytelniających – aktualnego użytkownika, tokena oraz lokalnego indeksu – oraz przeniesieniu osoby, korzystającej z aplikacji ponownie do ekranu startowego.

```
def logout():  
    global local_id  
    global token_id  
    try:  
        auth.current_user = None  
        token_id = None  
        local_id = None  
        App.get_running_app().change_screen("start")  
    except requests.exceptions.HTTPError as e:  
        error_json = e.args[1]  
        error = json.loads(error_json)['error']['message']  
pass
```

Listing 7.4 Metoda logout

7.2.2. Moduł ankiety preferencji

Na listingu 7.5 przedstawiona została klasa `SurveyWindow`, która stanowi logikę modułu ankiety preferencji w omawianej aplikacji.

```
class SurveyWindow(Screen):  
    which_one = 0  
  
    def on_pre_enter(self):  
        global survey_obj  
        survey_obj = LoginFirestore.get_survey()  
        self.ids.error_lbl.text = ""  
        if survey_obj is None:  
            pass  
        else:  
            if not survey_obj[1]:  
                self.ids.male.active = True  
            else:  
                self.ids.female.active = True  
  
            self.ids.first_lbl.text = survey_obj[2]  
            self.ids.sec_lbl.text = survey_obj[3]  
  
    def check_if_exists(self):  
        is_first_good = True  
        if self.ids.first_lbl.text == "" and self.ids.sec_lbl.text == "":  
            self.add_survey_and_change_sc("", "")  
        else:  
            first_survey_obj =
```

```

LoginFirebase.get_perfumes_by_name(self.ids.first_lbl.text)
    sec_survey_obj =
LoginFirebase.get_perfumes_by_name(self.ids.sec_lbl.text)

    if first_survey_obj == "brak" and sec_survey_obj == "brak":
        self.ids.error_lbl.text = "Dla 1 i 2: Błąd w pisowni lub brak
perfumy w bazie"
    else:
        if self.ids.first_lbl.text == "":
            first_perf = ""
            pass
        elif first_survey_obj == "brak":
            self.ids.error_lbl.text = "Dla 1: Błąd w pisowni lub brak
perfumy w bazie"
            is_first_good = False
        else:
            first_perf = self.ids.first_lbl.text

        if self.ids.sec_lbl.text == "" and is_first_good:
            sec_perf = ""
            self.add_survey_and_change_sc(first_perf, sec_perf)
            pass
        elif sec_survey_obj == "brak" and is_first_good:
            self.ids.error_lbl.text = "Dla 2: Błąd w pisowni lub brak
perfumy w bazie"
        else:
            if is_first_good:
                sec_perf = self.ids.sec_lbl.text
                self.add_survey_and_change_sc(first_perf, sec_perf)

    def add_survey_and_change_sc(self, first_perf, sec_perf):
        LoginFirebase.add_survey(self.ids.female.active, first_perf, sec_perf)
        MDApp.get_running_app().change_screen("recommendation")

```

Listing 7.5 Klasa SurveyWindow

Klasa posiada metodę `on_pre_enter`, wywoływaną, zanim użytkownikowi załaduje się ekran ankiety, aby móc uzupełnić zawartość ekranu. Wewnątrz metody pobierany jest obiekt (ankieta) z bazy danych, metodą `get_survey` ukazaną na listingu 7.6, i zapisywany do zmiennej `survey_obj`. Potem sprawdzana jest zawartość `survey_obj`, by zasilić zawartość pól w ankiecie danymi z bazy. Jeżeli pobrany obiekt jest pusty, ankietę nie zostanie uzupełniona.

W klasie istnieje również metoda `check_if_exists`, która jest odpowiedzialna za weryfikację wpisanych do pól perfum. Istnieje jako zabezpieczenie przed literówkami, a także sprawdza czy w bazie danych istnieje pozycja zapachowa o podanej przez użytkownika nazwie.

```

def get_survey():
    try:
        survey = db.child('surveys').child(local_id).get(token_id)
        return survey.val()
    except requests.exceptions.HTTPError as e:
        error_json = e.args[1]
        error = json.loads(error_json)['error']['message']
    pass

```

Listing 7.6 Metoda get_survey

Natomiast metoda `add_survey_and_change_sc` odpowiada za wywołanie metody `add_survey` (listing 7.7), aby zapisać sprawdzoną ankietę. Przekierowuje również użytkownika od razu do modułu rekomendacji, żeby mógł zobaczyć wygenerowane listy rekomendacyjne.

```
def add_survey(gender, perf_first, perf_sec):
    try:
        users_survey = {
            '1': gender,
            '2': perf_first,
            '3': perf_sec
        }
        db.child('surveys').child(local_id).set(users_survey, token_id)
    except requests.exceptions.HTTPError as e:
        error_json = e.args[1]
        error = json.loads(error_json)['error']['message']
    pass
```

Listing 7.7 Metoda add_survey

7.2.3. Moduł rekomendacyjny

Na listingu 7.8 zaprezentowano metodę `rec_func`, która odpowiada za logikę systemu rekomendacyjnego omawianej aplikacji. Jej sposób działania w formie pseudokodu został przedstawiony dokładnie na listingu 6.1 w podrozdziale 6.2.

```
def rec_func(gender_bool, chosen_name, perfumes):
    json_string = json.dumps(perfumes)
    perfumes_data = pd.read_json(json_string)

    if not gender_bool:
        gender = "Men"
    else:
        gender = "Women"

    selected_traits = ['base_note', 'middle_note', 'scents']

    for trait in selected_traits:
        perfumes_data[trait] = perfumes_data[trait].fillna('')

    def combine_features(row):
        return row['base_note'] + " " + row['middle_note'] + " " + row['scents']

    perfumes_data["combined_features"] = perfumes_data.apply(combine_features,
axis=1)

    cvectorizer = CountVectorizer()
    count_matrix = cvectorizer.fit_transform(perfumes_data["combined_features"])
    cosine_sim = cosine_similarity(count_matrix)

    def get_id_from_name(name):
        return perfumes_data[perfumes_data.name == name]['index'].values[0]
```

```

perfume_id = get_id_from_name(chosen_name)
similar_perfumes = list(enumerate(cosine_sim[perfume_id]))
sorted_similar_perf = sorted(similar_perfumes, key=lambda x: x[1],
reverse=True)

def get_title_from_index(index):
    if perfumes_data.at[perfumes_data.index[index], 'department'] == gender:
        return perfumes_data[(perfumes_data.at[perfumes_data.index[index],
'department'] == gender) and (
            perfumes_data.index == index)][ 'name'].values[0]
    elif perfumes_data.at[perfumes_data.index[index], 'department'] ==
"Unisex":
        return perfumes_data[(perfumes_data.at[perfumes_data.index[index],
'department'] == "Unisex") and (
            perfumes_data.index == index)][ 'name'].values[0]
    else:
        return ''

title_list = []

for perfume in sorted_similar_perf:
    title = get_title_from_index(perfume[0])
    perf_factor = perfume[1]
    if title != '':
        title_list.append((title, perf_factor))

seen = set()
result = []
for item in title_list:
    if item[0] not in seen:
        seen.add(item[0])
        result.append(item)

return result

```

Listing 7.8 Metoda rec_func

Natomiast listing 7.9 ukazuje klasę ekranu rekomendacyjnego, na którym wyświetlone zostają listy wygenerowane przez metodę `rec_func`. Klasa posiada trzy metody.

Pierwszą z nich jest `on_pre_enter`, dzięki której pobierane są odpowiedzi z ankiety dla konkretnego użytkownika. Nazwy perfum zostają wyodrębnione i wpisane do przycisków, umożliwiających zmianę lokalizacji między listami. Jeżeli użytkownik pozostawił w ankiecie preferencji pole z nazwą pozycji puste, to wtedy przycisk otrzymuje napis: „brak”.

Drugą metodą jest `pressed`, która umożliwia zmianę ekranu rekomendacji na ekran pozycji perfumy, po naciśnięciu na element listy rekomendacyjnej. Pozwala ona również na zapisanie, z jakiego modułu dostajemy się do specyfikacji perfumy, aby użytkownik mógł w prosty sposób do niego wrócić.

Ostatnia i najważniejsza metoda to `set_list`, dzięki której zostaje utworzona lista dla konkretnej perfumy, jakiej wybór dokonuje się poprzez naciśnięcie odpowiedniego przycisku na ekranie. Następnie do zmiennej `list_of_all_perfumes` zostaje przypisana wygenerowana przez moduł poleceń lista pozycji wraz z procentem zgodności. Proces wiodący do otrzymania owej listy zaczyna się od metody `get_perfumes_by_preferences`, która odpytuje bazę, ażeby otrzymać listę wszystkich dostępnych perfum i szczegóły ankiety

potrzebne metodzie `rec_func` – to ona zwraca posortowaną listę z rekomendowanymi perfumami oraz ich współczynnikiem zgodności.

Jeżeli zwrócona lista jest pusta, wtedy wyskakuje komunikat, że należy poprawnie uzupełnić ankietę, ponieważ preferencje nie są zgodne z wymaganiami modułu rekomendacyjnego. Jeśli zwrócona lista jest zasilona pozycjami, wtedy zaczyna się inicjalizowanie `CustomListItem`, która tworzy każdy element listy z osobna, dodając odpowiednią nazwę perfumy, procent zgodności oraz stosowny pasek postępu w zależności od procentu podobieństwa.

```
class RecommendationWindow(Screen):
    def on_pre_enter(self):
        survey_ans = LoginFirebase.get_survey()
        if survey_ans[2] == '':
            self.ids.first_btn.text = 'brak'
        else:
            self.ids.first_btn.text = str(survey_ans[2])

        if survey_ans[3] == '':
            self.ids.sec_btn.text = 'brak'
        else:
            self.ids.sec_btn.text = str(survey_ans[3])

        self.set_list(1)

    def pressed(self, value):
        global perfume_name
        global previous_sc
        self.ids.container.clear_widgets()
        perfume_name = value.text
        previous_sc = "recommendation"
        MDApp.get_running_app().change_screen("perfume")
        perfume = PerfumeWindow()

    def set_list(self, first_or_sec):
        self.dialog = None
        self.ids.container.clear_widgets() # refresh list
        list_of_all_perfumes =
LoginFirebase.get_perfumes_by_preferences(first_or_sec)
        if list_of_all_perfumes[0] == "brak":
            self.ids.for_perf_id.text = "Dla perfumy: " + "brak"
            self.ids.container.add_widget(
                OneLineListItem(text="brak")
            )
        if not self.dialog:
            self.dialog = MDDialog(
                text="Poprawnie uzupełnij ankietę! Aplikacja nie załaduje
danych, dopóki preferencje nie zostaną uzupełnione.",
                buttons=[
                    MDFlatButton(
                        text="Rozumiem",
                        on_release=lambda _: self.dialog.dismiss()
                    )
                ],
            )
        self.dialog.open()
```

```

else:
    first_el = True
    for perf in list_of_all_perfumes:
        if first_el:
            self.ids.for_perf_id.text = "Dla perfumy: " + perf[0]
            first_el = False
        else:
            proc = round(perf[1], 2)
            if 0 <= proc <= 0.4:
                custom_icon = "img\min_bar.png"
            if 0.41 <= proc <= 0.6:
                custom_icon = "img\half_bar.png"
            if 0.61 <= proc <= 1:
                custom_icon = "img\max_bar.png"
            self.ids.container.add_widget(
                CustomListItem(text=perf[0], secondary_text="Zgodne w "
+ str(proc*100).split('.', 1)[0] + "%", icon=custom_icon, on_press=self.pressed)
            )

```

Listing 7.9 Klasa RecommendationWindow

7.2.4. Moduł wyszukiwania

Na listingu 7.10 zaprezentowana została klasa `DiscoverMain`, która odpowiada za obsługę modułu wyszukiwania. Posiada dwie kluczowe metody.

Pierwszą z nich jest `pressed`, która pozwala na zmianę ekranu wyszukiwania na ekran pozycji perfumy, po kliknięciu w element listy.

Drugą metodą jest to `set_list`, w jakiej następuje utworzenie listy, dostosowanej do wyszukiwanego warunku. Do zmiennej `list_of_all_perfumes` przypisywana jest lista zwracana przez metodę `get_perfumes_titles`, przedstawioną na listingu 7.11, następnie sprawdzane są wszystkie pozycje listy, które zaczynają się na wpisaną w wyszukiwarkę frazę. Dodatkowo dopuszczone zostały różnice w wyszukiwaniu po dużych oraz małych literach – dzięki metodzie `lower` nazwy perfum z bazy oraz wpisana treść stają się ciągiem małych liter. To sprawia, iż proces szukania dla użytkownika robi się łatwiejszy, ponieważ nie musi on pamiętać o używaniu różnego rozmiaru liter.

```

class DiscoverMain(Screen):
    def pressed(self, value):
        global perfume_name
        global previous_sc
        self.ids.container.clear_widgets()
        self.ids.search_field.text = value.text
        perfume_name = value.text
        previous_sc = "discovermain"
        MDApp.get_running_app().change_screen("perfume")
        perfume = PerfumeWindow()

    def set_list(self, text=" "):
        self.ids.container.clear_widgets()
        list_of_all_perfumes = LoginFirebase.get_perfumes_titles()
        for icon in list_of_all_perfumes:
            if icon.lower().startswith(text.lower()):

```

```

        self.ids.container.add_widget(
            OneLineListItem(text=icon, on_press=self.pressed)
        )

```

Listing 7.10 Klasa DiscoverMain

```

def get_perfumes_titles():
    try:
        new_list = []
        making_title_list = []
        perfumes_list = db.child('perfumes').get(token_id)
        for perfume in perfumes_list:
            new_list.append(perfume.val())
        for perfume in new_list:
            making_title_list.append(perfume['name'] + ' (' +
perfume['brand'] + ')')
        seen = set()
        result = []
        for item in making_title_list:
            if item not in seen:
                seen.add(item)
                result.append(item)
        return result
    except requests.exceptions.HTTPError as e:
        error_json = e.args[1]
        error = json.loads(error_json)['error']['message']
    pass

```

Listing 7.11 Metoda get_perfumes_titles

7.2.5. Tłumaczenie informacji ze zbioru danych

W tym podrozdziale zaprezentowano, w jaki sposób zostają przetłumaczone angielskie słowa ze zbioru danych na język polski, aby użytkownik nie miał problemu ze zrozumieniem składów perfum.

Przyciski widoczne na rysunku 7.12 odpowiadają trzem metodom, których logika jest identyczna. Autor omówi jedną z nich, aby przedstawić ich ogólny sposób działania wraz z zastosowaną biblioteką Googletrans, odpowiedzialną za tłumaczenia.


```

def set_items_base(self):
    global perfume_obj
    new_list = perfume_obj['base_note'].split(",")
    translated_list = []
    translator = Translator()
    for item in new_list:
        out = translator.translate(item, dest="pl")
        translated_list.append(out.text)
    menu_items = [
        {
            "text": f"{i}",
            "viewclass": "Item",
            "height": dp(54),
            "pos_hint": {"center_x": .5, "center_y": .640}
        } for i in translated_list
    ]
    self.menu = MDDropdownMenu(
        caller=self.ids.scent,
        items=menu_items,
        width_mult=4,
    )
    self.menu.open()

```

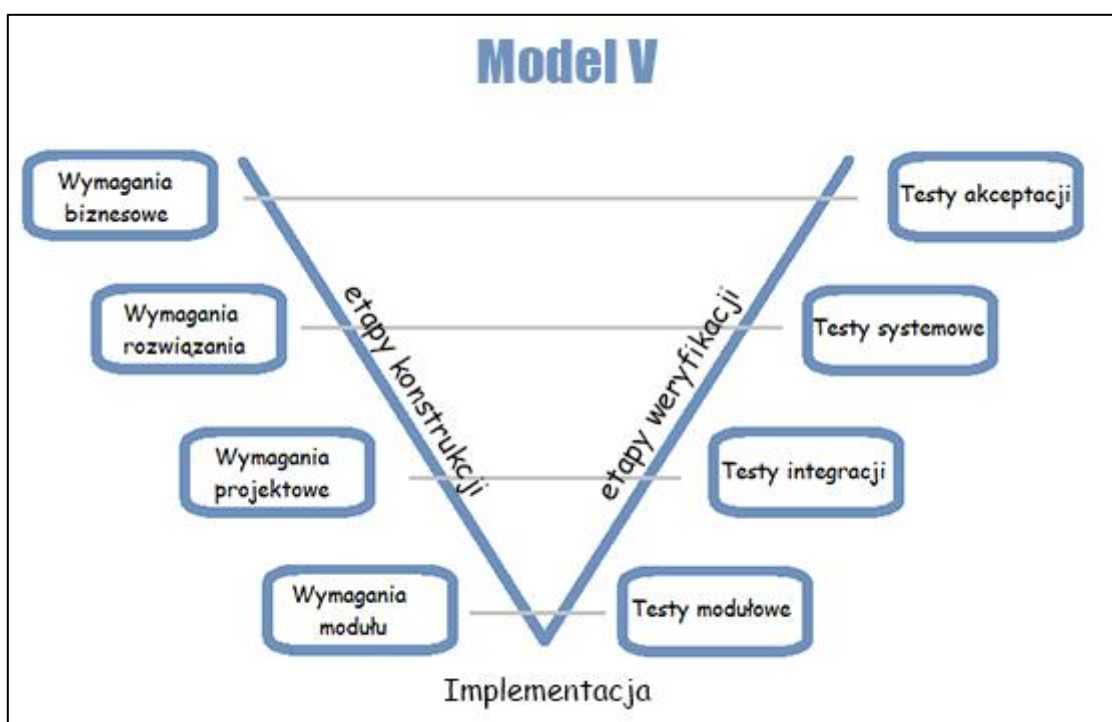
Listing 7.12 Metoda set_items_base

Metoda `set_items_base`, przedstawiona na listingu 7.12, służy do wyświetlania nut bazy dla konkretnej pozycji zapachowej. Do zmiennej `new_list` zostaje przypisana lista, utworzona poprzez podzielenie ciągu tekstowego, pochodzącego z pola `base_note` z obiektu `perfume_obj`, w miejscu wystąpienia przecinków. W `new_list` przechowywane są wszystkie nuty górne dla wybranego zapachu w języku angielskim. Następnie został stworzony obiekt `Translator`, dzięki któremu autor uzyskał możliwość dostępu do metody `translate`. Pozwala ona, po zdefiniowaniu języka, na przetłumaczenie wprowadzonego tekst. Przetłumaczone elementy dodano do listy `translated_list`, a następnie utworzono z niej menu, które zawiera wszystkie nuty zapachowe w języku polskim.

8. Testowanie

Testowanie to jeden z najważniejszych etapów tworzenia aplikacji, ponieważ to on definiuje stopień poprawności działania wszystkich funkcjonalności, a także wskazuje miejsca, które zawierają błędy.

Popularnym modelem sekwencyjnym do testowania oprogramowania jest Model V, którego główną ideę stanowi równoległe wykonywanie budowy oraz weryfikacji aplikacji [Model V, 2023]. Omawiana aplikacja została przetestowana poprzez wykorzystanie Modelu V (na rysunku 8.1), którego strukturę specjalnie dostosowano do projektu. Rozgraniczenie poszczególnych etapów wytwarzania oraz testowania umożliwiło dokładne wyodrębnienie błędów oraz problemów, które posiada aplikacja. Dzięki wczesnemu rozpoczęciu testowania, systematycznie wprowadzane poprawki usprawniły proces implementacyjny.



*Rysunek 8.1 Model V
[źródło: Modele Wytwarzania Oprogramowania 2023]*

8.1. Problem z danymi

Wady dobranego zestawu danych przedstawiono szerzej w podrozdziale 1.2 i 9.1, w których omówiono brak kluczowej dla systemu rekomendacji kolumny nut górnych (głowy). Problem został rozwiązany poprzez zastąpienie jej kolumną, jaka przechowuje dane o ogólnym zapachu perfumy. To pozwoliło na zaprojektowanie systemu rekomendacji, którego mechanizm będzie na tyle uniwersalny, że gdy do zbioru danych dołączona zostanie kolumna z nutami górnymi, wystarczy drobna modyfikacja (zamiany nazwy kolumny) w kodzie, aby dopasować działanie systemu do pełnej specyfikacji zapachowej.

Kolejną wadą był język wybranego zestawu danych. Omawiana aplikacja została stworzona w języku polskim, jednak wszystkie dostarczone do niej dane były w języku angielskim. Ten problem rozwiązała biblioteka Googletrans, jaką autor wykorzystał do skutecznego tłumaczenia składników perfum.

8.2. Problemy implementacyjne

Najważniejszym problemem implementacyjnym był brak podstawowych zabezpieczeń podczas testowania modułów: logowania oraz rejestracji. Przykładowo, gdy użytkownik wpisywał złe hasło, nie dostawał informacji zwrotnej, która pojawiała się tylko na poziomie programu, toteż nie wiedział, co się dzieje i czemu nie może dostać się do swojego konta. Ten problem został rozwiązany poprzez wprowadzenie odpowiednich zabezpieczeń, czyli wyróżnionych kolorem czerwonym informacji, które sygnalizowały użytkownikowi poczynione niepoprawności.

Kolejny problem stanowiło zaimplementowanie animacji, jakiej zadaniem było zwiększenie przyjemności korzystania z aplikacji poprzez wzbogacenie odbioru estetycznego interfejsu. Po dodaniu jej do ekranu startowego zauważono brak płynności, która nie występowała przy osobnym jej odtwarzaniu. Finalnie brak zaawansowanej znajomości frameworka Kivy oraz KivyMD spowodował, iż autorka zrezygnowała z wdrożenia utworzonego wcześniej pliku w formacie GIF, pozostawiając na ekranie startowym tylko logo aplikacji.

Podczas testowania modułu rekomendacji zauważono, że dla niektórych perfum generowana lista rekomendacyjna posiadała powtarzające się pozycje. Autorka pracy zastosowała w kodzie eliminację duplikatów o tej samej nazwie, dzięki czemu pozbyto się niepotrzebnego namnożenia tych samych produktów.

8.3. Problemy wydajnościowe

Zaobserwowane zostały dwa problemy wydajnościowe. Pierwszy z nich dotyczy zbyt wolnego działania aplikacji, a drugi o za małej ilości danych.

Problem z szybkością aplikacji pojawia się w dwóch miejscach – w module wyszukiwania oraz rekomendacji. Wpierw autor podejrzewał komplikacje na poziomie komunikacji programu z bazą lub algorytmu rekomendacyjnego, jednakże po zbadaniu drogi przepływu danych, okazało się, iż aplikacja zwalnia, kiedy musi utworzyć listę poprzez użycie `OneListItem` lub `TwoLineAvatarListItem`. Niestety niewystarczająca znajomość frameworka Kivy oraz KivyMD uniemożliwiła usprawnienie kreacji listy. Jednakże gdyby autor pracy posiadał więcej czasu na zaznajomienie się z technologią, najprawdopodobniej znalazłby optymalny sposób implementacji listy.

Choć z pozoru mogłoby się wydawać, iż zbiór perfum, który posiada 1002 pozycji, jest dużym zestawem danych, tak nie jest, zwłaszcza patrząc z perspektywy eliminacyjnych właściwości systemu rekomendacyjnego. Podczas prowadzenia testów w module

rekomendacyjnym, zauważono, iż nie wszystkie perfumy mają chociaż jeden polecony produkt o zgodności większej niż 50%. To powoduje, że użytkownik, chcący wybrać zapach z listy poleceń może trafić na produkty, które nie będą odpowiadać jego preferencjom. Jedynym rozwiązaniem problemu jest powiększenie zbioru danych o dodatkowe pozycje, co niestety nie było wykonalne – głównie przez przewidywaną pracochłonność tego zadania.

8.4. Testy akceptacji użytkowników

Dzięki przeprowadzonym testom aplikacji przez grupę użytkowników, zostały wyłonione problemy, które autorka rozważyła podczas implementacji oprogramowania i wprowadzania zmian. Użytkownicy zwrócili uwagę na szybkość działania aplikacji, która już została głębiej omówiona w podrozdziale 9.3., toteż autor pozwoli sobie w tym miejscu pominąć tę obserwację.

Dodatkowo zostało zauważone, iż interfejs w module rekomendacyjnym nie był wystarczająco intuicyjny. Przycisków, dotyczących zmiany list nie podpisano, przez co użytkownik mógł zapomnieć, jakie produkty wybrał w ankiecie preferencji. Autorka dodała odpowiednie napisy do przycisków, zmieniających lokalizację, aby interfejs był jak najbardziej przejrzysty.

W tym samym module istniał problem z nieoczywistą kolejnością elementów listy, dlatego też dodano paski postępu oraz procenty zgodności, aby użytkownik widział, która pozycja ma największą zgodność, a jaka najmniejszą.

Podczas testów zwrócono również uwagę na kolorystykę aplikacji – okazało się, że dobrane barwy mogą przeszkadzać w odbiorze aplikacji, ponieważ kontrast między białymi napisami a kolorowym tłem zdaje się być zbyt mały. Po rozważeniu opinii na temat barw przewodnich interfejsu, autor zdecydował, iż jednak zostawi pastelowe kolory wraz z jasnym motywem przewodnim.

9. Badanie systemu rekomendacji

W niniejszym rozdziale zostanie przeprowadzone badanie działania wdrożonego do aplikacji systemu rekomendacyjnego, czyli omówienie wykorzystanego zbioru danych oraz skuteczności i rzetelności systemu poprzez przetestowanie rekomendacji na konkretnych pozycjach. Potrzeba przeprowadzenia badania jest ważna dla zdefiniowania istotności systemu oraz zgodności z rzeczywistością i potrzebami użytkowników.

9.1. Analiza zestawu danych

W czasie projektowania aplikacji zbiór danych pobrany z serwisu Kaggle, który autorka pracy omówiła dokładnie w podrozdziale 1.2., został przeanalizowany pod kątem zgodności z wymaganiami aplikacji. Ze wszystkich dostępnych w serwisie zestawów wykorzystany zbiór okazał się być najbardziej stosowny, jednak zawierał też pewne wady.

Największym problemem była niepełna specyfikacja zapachowa, czyli brak nut górnych (głowy), o których również wspomniano w podrozdziale 1.2. wraz z wyjaśnieniem, czym są nuty. Niestety nie była to kwestia, jaką dało się w szybki sposób uzupełnić. Autor musiałby przeszukać inne dostępne źródła, traktujące o perfumach, i zebrać odpowiednie informacje na temat nut górnych, ażeby uzupełnić braki w ponad tysiącu pozycji.

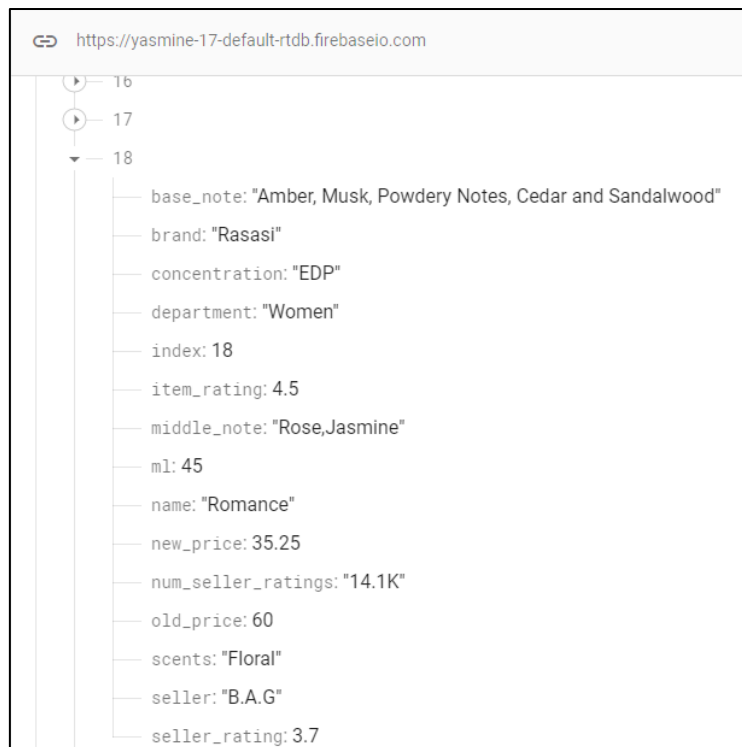
W związku z tym, nuty górne zostały zastąpione kolumną dotyczącą zapachu. To spowodowało, iż system do analizy finalnie i tak wykorzystał trzy kolumny, zamieniając nuty głowy na ogólny zapach danej perfumy. Jest to rozwiązanie optymalne przez wzgląd na późniejszą możliwość uzupełnienia zbioru danych dodatkową kolumną nut górnych. Dzięki prostej podmianie nazwy kolumny w kodzie, modyfikacje nie będą stanowiły problemu, a system wzbogaci się o możliwość precyzyjniejszej rekomendacji.

9.2. Badanie konkretnych pozycji zapachowych

W tym podrozdziale zostaną rozpatrzone trzy przypadki, aby ukazać działanie systemu rekomendacyjnego. Autor zaprezentuje dane z debuggera, czyli narzędzia, analizującego dynamicznie program [Debugger, 2023], aby udowodnić zgodność polecanych pozycji. Zostaną zbadane trzy grupy perfum

9.2.1. Badanie perfumy dla kobiet: Romance

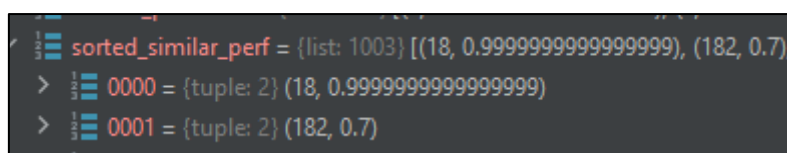
W pierwszym badaniu wzięła udział jedna z perfum dla kobiet: Romance, której skład jest przedstawiony na rysunku 9.1. Trzy pola – `base_note`, `middle_note`, `scents` – prezentują kolumny, jakie zostały wykorzystane przez system rekomendacyjny do stworzenia rekomendacji.



Rysunek 9.1 Pozycja o indeksie 18 – Romance
[źródło: opracowanie własne]

Następnie, w trakcie generowania listy rekomendacyjnej, autorka użyła debuggera, aby zaprezentować jej zawartość wraz ze współczynnikami zgodności.

Lista zbudowana jest z posortowanych po współczynnikach (od największego do najmniejszego) elementów, na które składają się dwie wartości – indeks perfumy oraz współczynnik zgodności. Na rysunku 9.2 ukazano element 0000, który składa się z indeksu 18 oraz posiada zgodność, jaka w zaokrągleniu wynosi 100%.

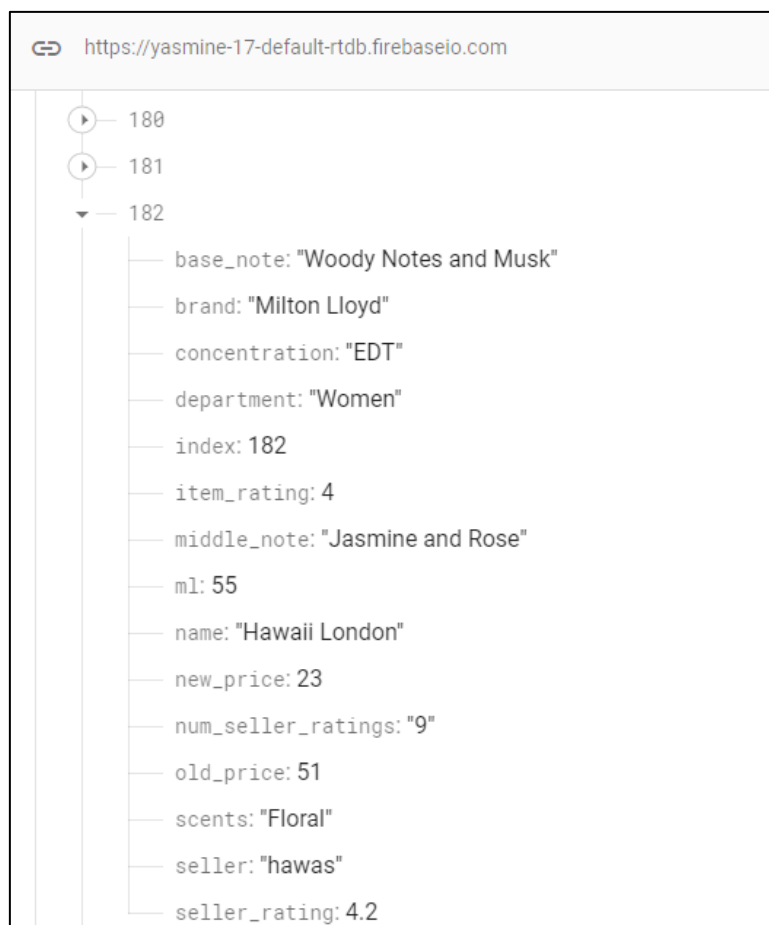


Rysunek 9.2 Debugger – posortowana lista dla perfumy Romance
[źródło: opracowanie własne]

W bazie danych pozycja 18 z obiektu perfumes to oczywiście Romance, co pokazuje, iż system trafnie wskazał, że największe podobieństwo (100%) produkt ma sam ze sobą. Wynik jest jak najbardziej poprawny, ponieważ tylko pozycja z identycznym składem, otrzymałaby tak wysoki współczynnik podobieństwa – w takim przypadku porównywane wektory nakładają się na siebie, co oznacza, że mają najmniejszy kąt między sobą.

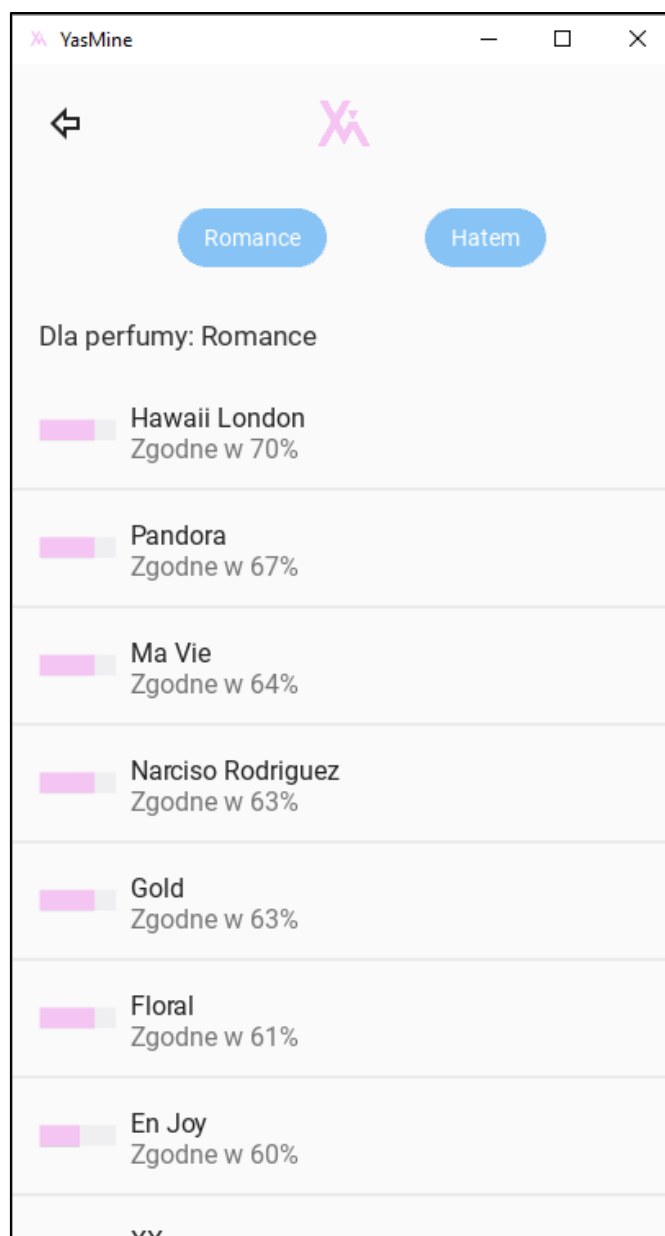
Rysunek 9.2 zaprezentował również element z innym indeksem (182), którego współczynnik zgodności jest równy w zaokrągleniu 70%. To jest druga najbardziej podobna pozycja do perfumy Romance.

Na rysunku 9.3 ukazano specyfikację pozycji o indeksie 182 – Hawaii London. Pola, które wykorzystano przy procesie rekomendacji (*base_note*, *middle_note*, *scents*), posiadają te same składniki jak ukazany na rysunku 9.1 skład perfumy Romance, co bezpośrednio udowadnia podobieństwo ich składów, a co za tym idzie – zgodność produktów.



Rysunek 9.3 Pozycja o indeksie 182 - Hawaii London
[źródło: opracowanie własne]

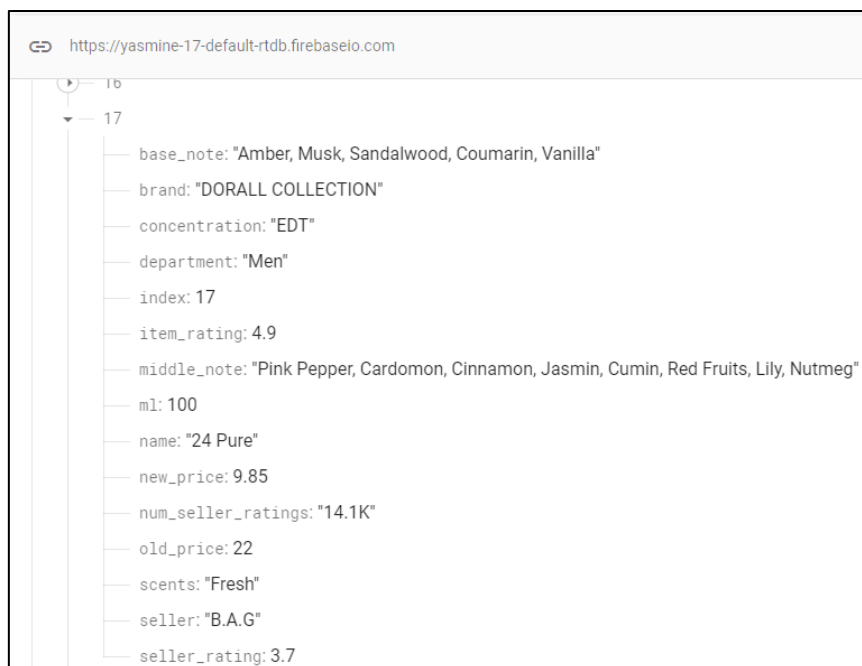
Rysunek 9.4 przedstawia ekran w module rekomendacji dla perfumy Romance. Pierwszym elementem listy jest omówiona pozycja Hawaii London o indeksie 182 ze współczynnikiem zgodności równym 70%. Zaraz za nią znajduje się reszta pozycji z posortowanej listy wraz z odpowiednimi współczynnikami.



*Rysunek 9.4 Lista rekomendacji dla perfumy Romance
[źródło: opracowanie własne]*

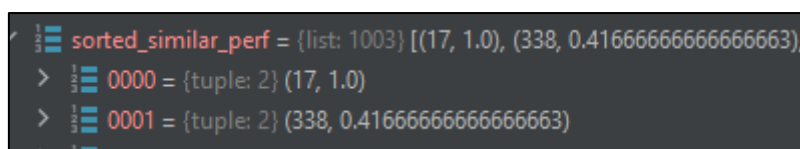
9.2.2. Badanie perfumy dla mężczyzn: 24 Pure

W drugim badaniu wzięła udział jedna z perfum dla mężczyzn 24 Pure. Proces badania tej pozycji przebiegł analogicznie do poprzedniego – autorka zaprezentowała skład tej perfumy (na rysunku 9.5), a następnie przeanalizowała pierwsze dwa elementy, posortowanej malejąco po współczynnikach zgodności, listy.



Rysunek 9.5 Pozycja o indeksie 17 – 24 Pure
[źródło: opracowanie własne]

Na rysunku 9.6 przedstawiono zawartość elementów posortowanej listy. Pierwszy element, czyli pozycja z indeksem 17 (24 Pure), posiada współczynnik zgodności równy 100%, co potwierdza poprawne działanie systemu, który rozpoznał największe podobieństwo produktu samego ze sobą.



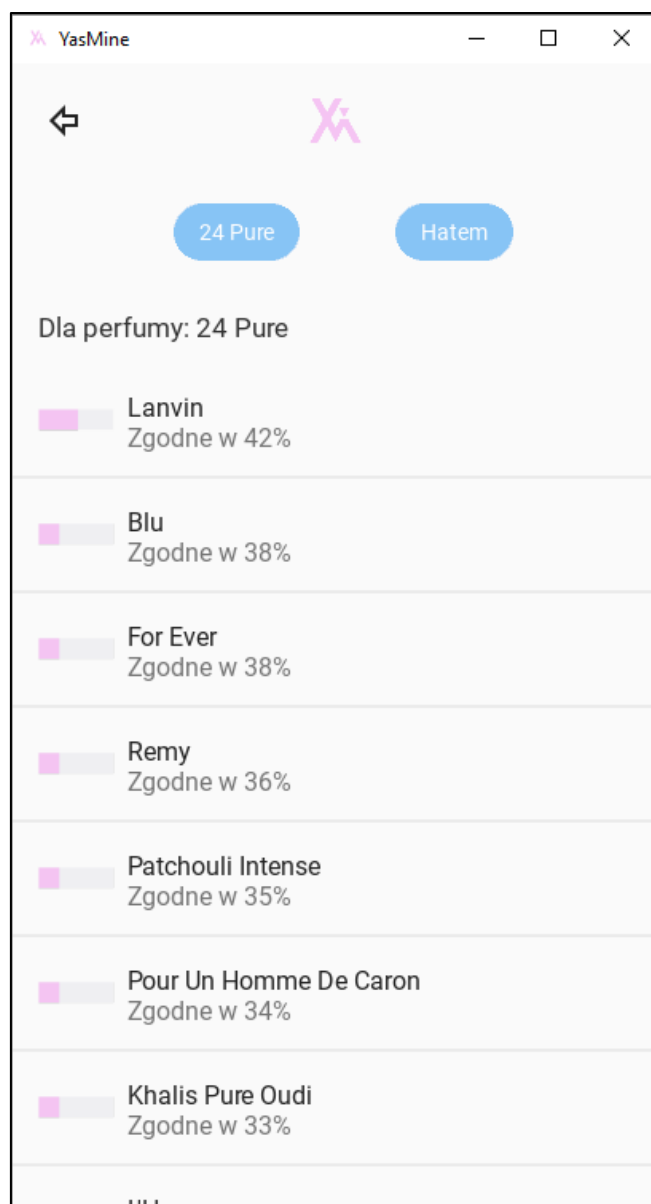
Rysunek 9.6 Debugger – posortowana lista dla perfumy 24 Pure
[źródło: opracowanie własne]

Drugi element, już tylko z 40% współczynnikiem zgodności, wskazuje na pozycję o indeksie 338 (Lanvin), jaka stanowi pierwszy najbardziej podobny produkt do badanej perfumy, jednocześnie nie będąc nią. Na rysunku 9.7 zaprezentowano skład pozycji Lanvin, który w pewnym stopniu pokrywa się z przedstawionym na rysunku 9.5 składem badanego 24 Pure. Niemniej jednak jednocześnie posiada zbyt wiele różnych nut zapachowych, żeby zdobyć chociaż 50% zgodności.



*Rysunek 9.7 Pozycja o indeksie 338 – Lanvin
[źródło: opracowanie własne]*

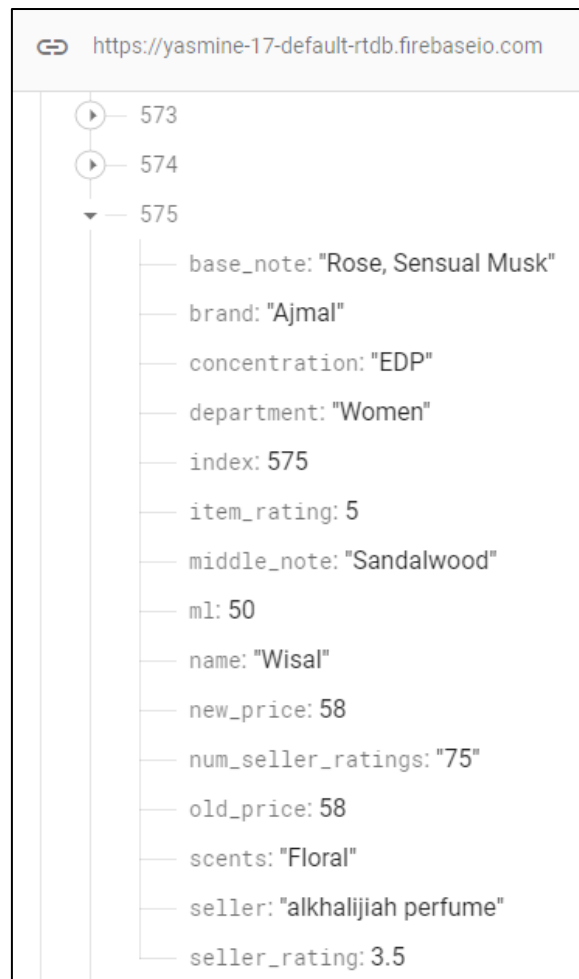
Rysunek 9.8 przedstawił jak prezentuje się reszta listy oraz procentów zgodności dla badanej pozycji 24 Pure.



*Rysunek 9.8 Lista rekomendacji dla perfumy 24 Pure
[źródło: opracowanie własne]*

9.2.3. Badanie perfumy unisex: Wisal

Trzecim i ostatnim badaniem będzie analiza perfumy unisex Wisal, której specyfikację ukazano na rysunku 9.9. Proces badania zostanie przeprowadzony analogicznie do poprzednich, poprzez porównanie składu perfumy wzorcowej ze składem pozycji o największym współczynniku zgodności. Autorka również omówi pierwsze dwa elementy, posortowanej malejąco po współczynnikach zgodności, listy.



Rysunek 9.9 Pozycja o indeksie 575 – Wisal
[źródło: opracowanie własne]

Na rysunku 9.10 zaprezentowano dwa pierwsze elementy listy. Element 0000, który ma w zaokrągleniu współczynnik równy 100%, posiada indeks o wartości 575. Stuprocentowa zgodność po raz kolejny dotyczy porównania perfumy samej ze sobą, ponieważ naturalnie jest to jedyna pozycja w bazie, jaka mogłaby zagwarantować całkowitą zgodność.

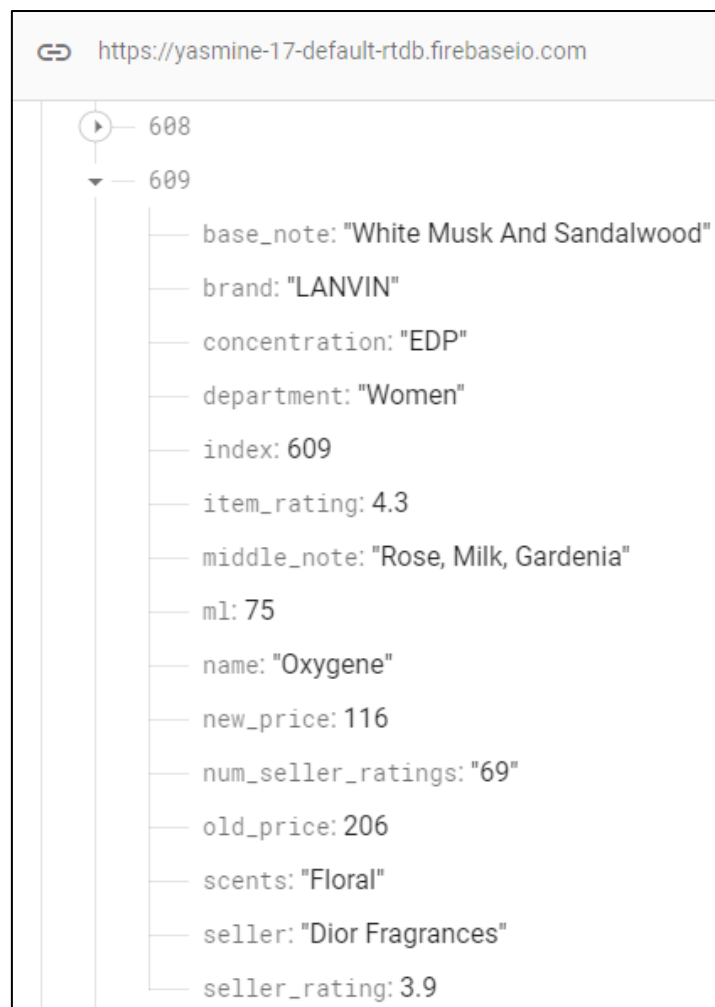
```

✓ sorted_similar_perf = {list: 1003} [(575, 0.9999999999999999), (609, 0.6324555320336758),
> 0000 = {tuple: 2} (575, 0.9999999999999999)
> 0001 = {tuple: 2} (609, 0.6324555320336758)

```

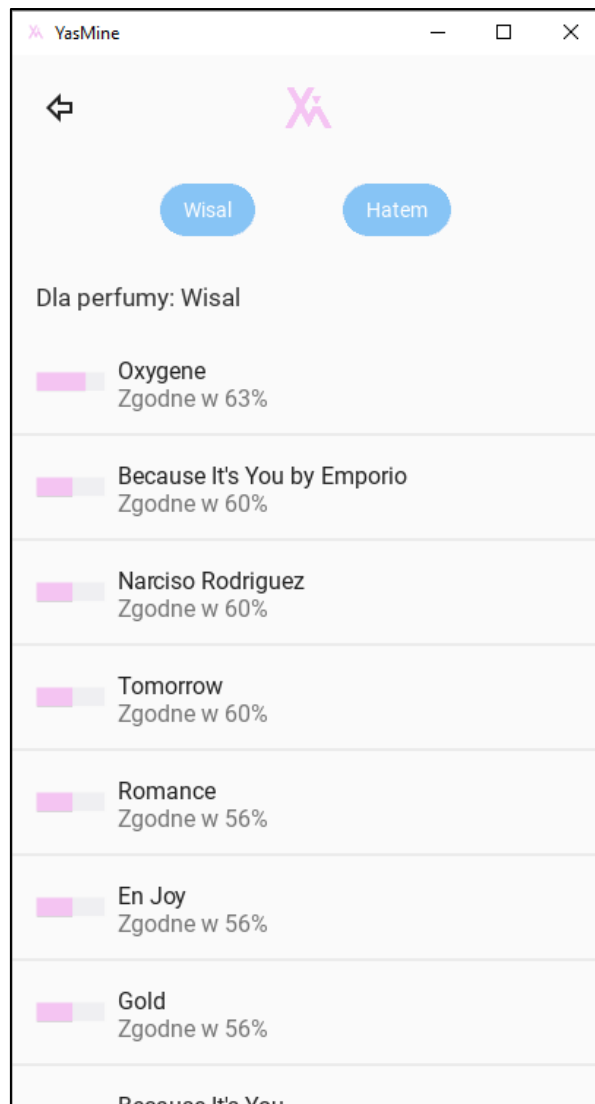
Rysunek 9.10 Debugger – posortowana lista dla perfumy Wisal
[źródło: opracowanie własne]

Drugi element listy składa się z indeksu 609, który wskazuje na produkt o nazwie Oxygene, oraz współczynnika wynoszącego 63%. Na rysunku 9.11 pokazano dokładną specyfikację zapachową perfumy Oxygene, jaka częściowo pokrywa się ze składem zaprezentowanym na rys. 9.9.



*Rysunek 9.11 Pozycja o indeksie 609 – Oxygene
[źródło: opracowanie własne]*

Dodatkowo na rysunku 9.12 prezentuje się reszta elementów wygenerowanej listy rekomendacji dla perfumy unisex Wisal.



*Rysunek 9.12 Lista rekomendacji dla perfumy Wisal
[źródło: opracowanie własne]*

9.3. Podsumowanie

Po wykonaniu badań i przeanalizowaniu zbioru danych, autorka niniejszej pracy stwierdziła, iż mimo braku ważnej (dla specyfikacji perfumy) kolumny, dotyczącej nut górnych, system działa dostatecznie dobrze. Prawidłowo wyłania najbardziej zgodne pozycje z bazy danych. Zazwyczaj pierwszy polecony produkt posiada współczynnik zgodności większy niż 50%, co umożliwia użytkownikowi otrzymanie wiarygodnej rekomendacji.

Warto wspomnieć, iż gdy pierwsza rekomendowana pozycja posiada współczynnik mniejszy niż 50%, nie wynika to z niepoprawnego działania systemu rekomendacyjnego, a ze zbyt małej liczby danych, pochodzących z wybranego zestawu. Należy pamiętać, że kompozycje zapachowe są różne i bardzo bogate, a nieco ponad tysiąc pozycji, które tworzą bazę produktów w omawianej aplikacji, nie jest w stanie objąć większości możliwych zestawień nut zapachowych.

10. Podsumowanie

W ramach realizowanej pracy dyplomowej zaprojektowano, zaimplementowano oraz przetestowano aplikację na urządzenia mobilne z systemem Android, która rekomenduje dobór perfum na podstawie preferencji i składników. Do zbudowania aplikacji oraz systemu rekomendacyjnego posłużono się językiem Python. Ponadto interfejs został stworzony, dzięki frameworkowi Kivy oraz KivyMD.

Aplikacja umożliwia użytkownikom utworzenie konta oraz skonstruowanie unikatowego profilu preferencji, w postaci ankiety, dzięki której będą mogli otrzymać dostęp do funkcjonalności rekomendacji. Moduł rekomendacyjny zrealizowano poprzez wykorzystanie modelu filtrowania opartego na treści, a także mechanizmów analizy danych cosine similarity. Dodatkowo każdy użytkownik posiada możliwość wyszukiwania oraz przeglądania dostępnych w bazie perfum. Aplikacja stanowi rzetelne narzędzie dla wszystkich osób, których wiedza w dziedzinie perfumiarstwa nie jest zaawansowana, a potrzebują w prosty sposób odnaleźć perfumy dla siebie lub swoich bliskich.

Zaimplementowana aplikacja spełnia wszystkie założenia oraz wymagania funkcjonalne i нефункционалне, które określono w fazie projektowej. System rekomendacyjny działa wystarczająco poprawnie, aby wskazać podobne pozycje zapachowe, a co za tym idzie pomóc potencjalnemu użytkownikowi wybrać perfumy. Warto wspomnieć, że autorka dostosowała oprogramowanie w sposób elastyczny do przewidywanej w przyszłości zmiany zestawu danych, Zbiór danych, na którym zbudowana została omawiana w niniejszej pracy wersja systemu rekomendacyjnego jest niepełny, co dokładniej przedstawia podrozdział 8.1. Zestaw nie posiada ważnego elementu kompozycji zapachowych, przez co wskazywane przez aplikację produkty mogą nieznacznie się różnić od zdefiniowanych preferencji użytkownika. Dodatkowo zbiór posiada 1002 pozycje, co w perspektywie olbrzymich baz zapachowych konkurencyjnych aplikacji ogranicza jego możliwości. Dlatego też ewentualna zmiana zestawu na taki, który posiada pełną specyfikację zapachową i większą ilość perfum, polepszyłaby jego działanie.

Autorka widzi możliwość rozwoju aplikacji, wprowadzając kolejne jej wersje z udoskonalonymi funkcjonalnościami oraz stosownymi poprawkami. Omawiana wersja tworzy solidną podstawę do rozbudowy systemu w najbliższej przyszłości, a także poprawienia problemów z wydajnością. Zauważona w podrozdziale 2.2 skuteczność hybrydowych systemów rekomendacji pozwala na stwierdzenie, iż opcjonalne rozszerzenie systemu rekomendacji, poprzez połączenie metody opartej na treści z metodą grupowego filtrowania uatrakcyjniłoby aplikację i umożliwiłoby konkurowanie z innymi popularnymi produktami, jakie omówiono dokładnie w podrozdziale 2.1.

BIBLIOGRAFIA

[**Adobe Photoshop, 2022**] *Adobe Photoshop*. [online] Dostępne na: <<https://www.adobe.com/products/photoshop.html>> [Udostępniono 26.12.2022].

[**Amazon, 2022**] *Amazon*. [online] Dostępne na: <<https://www.amazon.com/>> [Udostępniono 26.12.2022].

[**Berners-Lee, 2022**] Tim Berners-Lee. [online] Dostępne na: <<https://www.w3.org/People/Berners-Lee/>> [Udostępniono 17.06.2022].

[**Bin Taleb, M., 2022**] *noon perfume*. [online] Dostępne na: <<https://www.kaggle.com/datasets/monirahabdulaziz/noon-perfume>> [Udostępniono 26.12.2022].

[**Branża e-commerce w Polsce, 2023**] *Branża e-commerce w Polsce – charakterystyka i statystyki*. [online] Dostępne na: <<https://semcore.pl/branza-e-commerce-w-polsce-charakterystyka-i-statystyki/>> [Udostępniono 16.01.2023]

[**Colour Psychology, 2022**] *Colour Psychology – Pastel Colours*. [online] Dostępne na: <<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/>> [Udostępniono 02.01.2023].

[**Cosine Similarity, 2023**] *Cosine Similarity*. [online] Dostępne na: <<https://deeptai.org/machine-learning-glossary-and-terms/cosine-similarity>> [Udostępniono 02.01.2023].

[**CSV, 2022**] What is a CSV file? [online] Dostępne na: <<https://docs.fileformat.com/spreadsheet/csv/>> [Udostępniono 02.01.2023].

[**CTR, 2023**] Co to jest CTR? [online] Dostępne na: <<https://ks.pl/slownik/co-to-jest-ctr>> [Udostępniono 02.01.2023].

[**Darshan, M., 2023**] *What is cosine similarity and how is it used in machine learning?* [online] Dostępne na: <<https://analyticsindiamag.com/cosine-similarity-in-machine-learning/>> [Udostępniono 02.01.2023].

[**Debugger, 2023**] *Czym jest debugowanie?* [online] Dostępne na: <<https://jaktestowac.pl/lesson/pt1-mk6-s02-02/>> [Udostępniono 06.01.2023].

[**E-commerce i COVID-19, 2023**] *E-commerce w czasie pandemii Covid-19*. [online] Dostępne na: <<https://www.gov.pl/web/oecd/e-commerce-w-czasie-pandemii-covid-19>> [Udostępniono 10.01.2023].

[**Firebase, 2022**] *Firebase*. [online] Dostępne na: <<https://firebase.google.com/>> [Udostępniono 26.12.2022].

[**Firebase Realtime Database, 2022**] *Firebase Realtime Database*. Dostępne na: <<https://firebase.google.com/products/realtime-database>> [Udostępniono 29.12.2022].

[**Git, 2022**] *PyCharm*. [online] Dostępne na: <<https://git-scm.com/>> [Udostępniono 26.12.2022].

[**GitHub, 2022**] *GitHub*. [online] Dostępne na: <<https://github.com/>> [Udostępniono 26.12.2022].

[**Googletrans, 2022**] *Googletrans*. [online] Dostępne na: <<https://pypi.org/project/googletrans/>> [Udostępniono 26.12.2022].

[**Harvard, 2022**] *Anglia Ruskin University Library – Harvard System*. [online] Dostępne na: <<https://library.aru.ac.uk/referencing/harvard.htm>> [Udostępniono 17.06.2022].

[**Historia Internetu, 2022**] *Historia Internetu*. [online] Dostępne na: <https://pl.wikipedia.org/wiki/Historia_Internetu> [Udostępniono 17.06.2022].

[**Holewa, K., 2022**] *We know what you like! Perks of recommendation systems in business*. [online] Dostępne na: <<https://www.miquido.com/blog/perks-of-recommendation-systems-in-business/>> [Udostępniono 26.12.2022].

[**JSON, 2022**] *JSON: Wprowadzenie*. [online] Dostępne na: <<https://www.json.org/json-pl.html/>> [Udostępniono 26.12.2022].

[**Kaggle, 2022**] *Kaggle*. [online] Dostępne na: <<https://www.kaggle.com/>> [Udostępniono 26.12.2022].

[**Kalam, N., 2022**] *Why is Firebase very Popular?* [online] Dostępne na: <<https://nazhimkalam.medium.com/why-is-firebase-very-popular-a60922a4ff51>> [Udostępniono 26.12.2022].

[**Kivy, 2022**] *Kivy*. [online] Dostępne na: <<https://kivy.org/>> [Udostępniono 26.12.2022].

[**KivyMD, 2022**] *KivyMD*. [online] Dostępne na: <<https://kivymd.readthedocs.io/en/1.1.1/>> [Udostępniono 26.12.2022].

[**Krysik, A., 2022**] *Silnik rekomendacji produktowych w pigułce: co to jest i jak działa?* [online] Dostępne na: <<https://recostream.com/pl/blog/jak-dziala-silnik-rekomendacji-produktowej/>> [Udostępniono 26.12.2022].

[**Kumar, A., 2022**] *Recommender Systems in Machine Learning: Examples*. [online] Dostępne na: <<https://vitalflux.com/recommender-systems-in-machine-learning-examples/>> [Udostępniono 22.12.2022].

[**Logo, 2022**] *Logo, logotyp, sygnety – na czym polegają różnice?* [online] Dostępne na: <<https://soluma.pl/studio-grafiki,ac157/logo-logotyp-sygnety-na-czym-polegaja-roznice,1106>> [Udostępniono 26.12.2022].

[**Model V, 2023**] *Model V*. [online] Dostępne na: <<https://testerzy.pl/baza-wiedzy/model-v>> [Udostępniono 06.01.2023].

[**Modele Wytwarzania Oprogramowania, 2023**] *Modele Wytwarzania Oprogramowania*. [online] Dostępne na: <<https://tester.milenabednarczyk.pl/modele-wytwarzania-oprogramowania/>> [Udostępniono 06.01.2023].

[**Mohammad S., 2022**] Mohammad S., et al. *Content and history based movie recommendation system*. AIP Conference Proceedings, 2022, DOI: 10.1063/5.0081896; s. 8

[**Odległość Hamminga, 2023**] *Odległość Hamminga*. [online] Dostępne na: <<https://www.e-biotechnologia.pl/Artykuly/Odleglosc-Hamminga/>> [Udostępniono 10.01.2023].

[**Pandas, 2022**] *Pandas*. [online] Dostępne na: <<https://pandas.pydata.org/>> [Udostępniono 26.12.2022].

[**Peszka, D., 2022**] *Front-end a back-end aplikacji*. [online] Dostępne na: <<https://smartbees.pl/blog/frontend-backend-aplikacji>> [Udostępniono 02.01.2023].

[**PMR, 2022**] *PMR: Rekordowa dynamika rynku kosmetycznego w 2022 roku*. [online] Dostępne na: <<https://www.pmrmarketexperts.com/pmr-rekordowa-dynamika-rynku-kosmetycznego-w-2022-roku>> [Udostępniono 22.12.2022].

[**Pseudokod, 2023**] *Pseudokod*. [online] Dostępne na: <<https://datascience.eu/pl/programowanie-komputerowe/pseudokod/>> [Udostępniono 02.01.2023].

[**PyCharm, 2022**] *PyCharm*. [online] Dostępne na: <<https://www.jetbrains.com/pycharm/>> [Udostępniono 26.12.2022].

[Pyrebase, 2022] *Pyrebase*. [online] Dostępne na: <<https://github.com/thisbejim/Pyrebase/>> [Udostępniono 26.12.2022].

[Python, 2022] *Python*. Dostępne na: <<https://www.python.org>> [Udostępniono 26.12.2022].

[Python Virtual Environment, 2022] *Czym jest Python Virtual Environment (venv)?* [online] Dostępne na: <<https://jaktestowac.pl/lesson/pt1-mk5-s01-l02/>> [Udostępniono 02.01.2023].

[Requests, 2022] *Requests*. [online] Dostępne na: <<https://requests.readthedocs.io/en/latest/>> [Udostępniono 26.12.2022].

[Rośnie rynek kosmetyków, 2022] *Rośnie rynek kosmetyków premium. Konsumenci stawiają na skuteczne, wysokiej jakości produkty*. [online] Dostępne na: <<https://www.wiadomoscikosmetyczne.pl/artykuly/rosnie-rynek-kosmetykow-premium-konsumenci-stawiaj,70221>> [Udostępniono 22.12.2022].

[Rozwój branży kosmetycznej, 2022] *Rozwój branży kosmetycznej w Polsce i na świecie*. [online] Dostępne na: <<https://www.egospodarka.pl/171483,Rozwoj-branzy-kosmetycznej-w-Polsce-i-na-swiecie,1,117,1.html>> [Udostępniono 22.12.2022].

[Scikit-learn, 2022] *Scikit-learn*. [online] Dostępne na: <<https://scikit-learn.org/>> [Udostępniono 26.12.2022].

[Sklep Google Play, 2022] *Znajdowanie aplikacji Sklep Google Play*. [online] Dostępne na: <<https://support.google.com/googleplay/answer/190860>> [Udostępniono 22.12.2022].

[Spotify, 2022] *Spotify*. [online] Dostępne na: <<https://open.spotify.com/>> [Udostępniono 26.12.2022].

[Tulibacka, A., 2022] *Czym jest Material Design?* [online] Dostępne na: <<https://grafmag.pl/artykuly/czym-jest-material-design-teoria-zasady-materialy-i-przyklady/>> [Udostępniono 26.12.2022].

[XML, 2023] *XML introduction*. [online] Dostępne na: <https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction> [Udostępniono 10.01.2023].

[YouTube, 2022] *YouTube*. [online] Dostępne na: <<https://www.youtube.com/>> [Udostępniono 26.12.2022].

[Zasady tworzenia kompozycji zapachowych, 2022] *Zasady tworzenia kompozycji zapachowych*. [online] Dostępne na: <<https://www.mazidla.com/zasady-tworzenia-kompozycji-zapachowych>> [Udostępniono 26.12.2022].