1. Opis projektu

Projekt „SmartFlashcards” jest pilotażowym wydaniem aplikacji służącej do nauki za pomocą fiszek, korzystając z nowych rozwiązań technologicznych. W ramach tego projektu użytkownik może:

- Dodawać nowe fiszki poprzez ręczne wypełnienie formularza (pytanie, odpowiedź, kategoria, źródło).
- Edytować i usuwać już istniejące fiszki.
- Grupować fiszki w kategorie.
- Wykorzystywać wbudowany moduł OCR do automatycznego wyodrębniania tekstu z obrazów i konwertowania go na fiszki.
- Generować fiszki z dowolnego tekstu dzięki integracji z OpenAI (NLP).
- Uczyć się w trybie „Learning Mode”.

2. Główne funkcjonalności i sposób działania

a) Zarządzanie fiszkami (CRUD)

- Formularz do dodawania nowej fiszki (AddFlashcardView) waliduje wprowadzone dane, umożliwia wybranie istniejącej kategorii bądź dodanie nowej. Edytowanie (EditFlashcardView) działa analogicznie, zaś usuwanie (DeleteFlashcardView) wyświetla okno potwierdzenia, loguje akcję w plikach logów.
- Panel administracyjny (Django Admin) pozwala również na zarządzanie fiszkami bezpośrednio z poziomu interfejsu wbudowanego w Django.

b) Lista fiszek pogrupowanych w kategorie

- Widok FlashcardListView wczytuje wszystkie kategorie i fiszki, a następnie dokonuje paginacji (5 kategorii na stronę, 5 fiszek na stronę).

c) Tryb nauki – „Learning Mode”

- Użytkownik wybiera kategorię. Aplikacja wyszukuje fiszki z tej kategorii i wykorzystuje LearningService.get_next_flashcard() do wybrania kolejnej fiszki do nauki. Gdy użytkownik odsłoni odpowiedź, może oznaczyć ją jako „poprawną” lub „niepoprawną”. Fiszka aktualizuje swoje statystyki (pola correct_answers i incorrect_answers). Na podstawie liczb błędów i trafień aplikacja w prosty sposób zwiększa prawdopodobieństwo ponownego pojawienia się trudniejszych fiszek.

d) Generowanie fiszek z tekstu (NLP)

- Widok GenerateFromTextView umożliwia wklejenie tekstu przez użytkownika. Następnie FlashcardFacade.generate_flashcards_from_text() uruchamia NLPProcessor. Wygenerowane fiszki pojawiają się na stronie, lecz jeszcze nie są zapisane w bazie. Użytkownik może zaznaczyć, które z nich chce zachować (i ewentualnie je poprawić).

e) Generowanie fiszek z obrazu (OCR)

- Widok UploadPhotoView lub OCRResultsView służy do przesyłania zdjęć. Następnie OCRProcessor.extract_text(image_path) wyodrębnia tekst z obrazu. Tekst jest przetwarzany podobnie jak w przypadku zwykłego tekstu (również można skorzystać z NLP, gdyż FlashcardFacade łączy obie usługi). Po wyświetleniu wyniku OCR użytkownik może zadecydować, które fiszki zapisać.

f) Rendering fiszek (tekst / audio)

- Każda pojedyncza fiszka może być wyświetlona jako tekst (TextRenderer) bądź w formie pliku audio (AudioRenderer). Audio generowane jest przy użyciu biblioteki gTTS, a gotowy plik MP3 jest zwracany jako HttpResponse z typem MIME audio/mpeg. Obsługiwane języki to na ten moment angielski i polski.

3. Technologie i biblioteki

Django 5.1.3 – główny framework webowy.
SQLite3 – domyślna baza danych, łatwa w konfiguracji na potrzeby prototypu.
pytesseract + PIL (Pillow) – silnik OCR do wyodrębniania tekstu z obrazów.
OpenAI – biblioteka do komunikacji z modelem GPT-3.5 w module nlp.py.
gTTS – biblioteka generująca pliki dźwiękowe (MP3) z tekstu (Text-to-Speech).

4. Konfiguracja i uruchomienie

Zainstaluj wymagane pakiety (np. w wirtualnym środowisku):

pip install django==5.1.3 openai pillow pytesseract django-extensions gTTS

Skonfiguruj Django:

Uzupełnij w nlp.py klucz openai.api_key.
Upewnij się, że w settings.py opcja DEBUG jest ustawiona na True dla środowiska deweloperskiego, a SECRET_KEY jest bezpieczny w środowisku produkcyjnym.

Wykonaj migracje i uruchom serwer:

```
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
```

W przeglądarce aplikacja dostępna pod adresem http://127.0.0.1:8000/

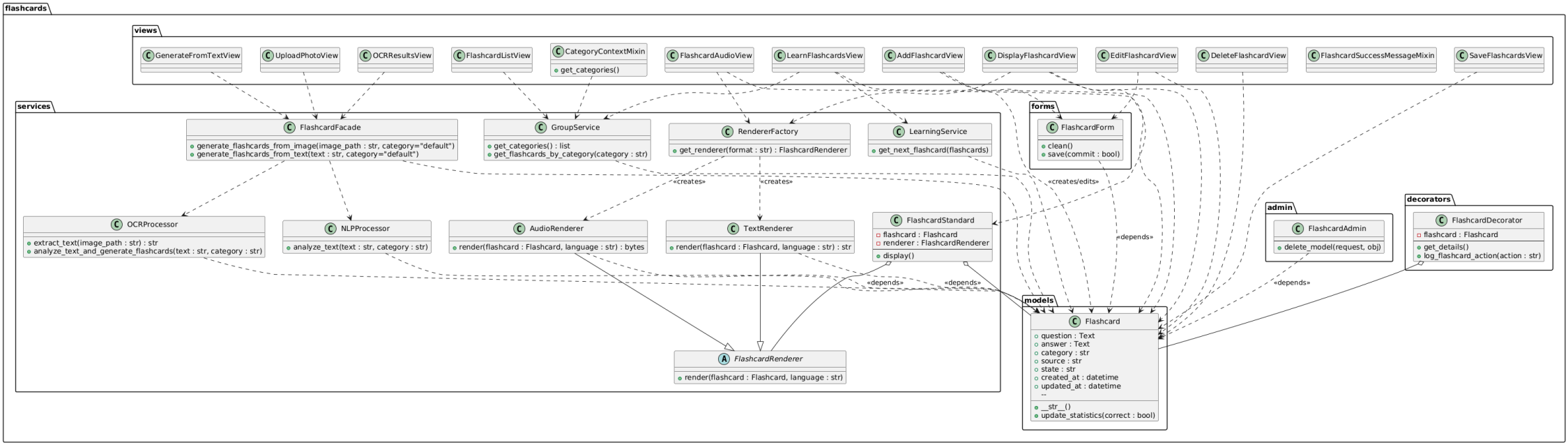
Dostęp do panelu admina – pod adresem:

http://127.0.0.1:8000/admin

Po wcześniejszym utworzeniu superusera (python manage.py createsuperuser).

5. Możliwe kierunki dalszego rozwoju

- Rozbudowa modelu Flashcard: można dodać pole trudności, priorytetu nauki, wsparcie dla języków (np. transkrypcje).
- Algorytmy nauki (SRS – Spaced Repetition System): Przy dalszej rozbudowie można je wdrożyć i dodać algorytm Leitnera lub SM-2 (Anki).
- Integracja z front-end: obecnie wykorzystywany jest głównie Bootstrap. Można wdrożyć React/Vue/Angular i komunikować się z API (rest_framework).
- Obsługa wielojęzyczna: w plikach .po i .mo, można przygotować tłumaczenia, by interfejs był dostępny np. w języku polskim, angielskim i innych.


flashcards
views
GenerateFromTextView
UploadPhotoView
OCRResultsView
FlashcardListView
CategoryContextMixin
+ get_categories()
FlashcardAudioView
LearnFlashcardsView
AddFlashcardView
DisplayFlashcardView
EditFlashcardView
DeleteFlashcardView
FlashcardSuccessMessageMixin
SaveFlashcardsView
services
FlashcardFacade
+ generate_flashcards_from_image(image_path : str, category="default")
+ generate_flashcards_from_text(text : str, category="default")
GroupService
+ get_categories() : list
+ get_flashcards_by_category(category : str)
RendererFactory
+ get_renderer(format : str) : FlashcardRenderer
LearningService
+ get_next_flashcard(flashcards)
forms
FlashcardForm
+ clean()
+ save(commit : bool)
«creates/edits»
«creates»
«creates»
«depends»
OCRProcessor
+ extract_text(image_path : str) : str
+ analyze_text_and_generate_flashcards(text : str, category : str)
NLPProcessor
+ analyze_text(text : str, category : str)
AudioRenderer
+ render(flashcard : Flashcard, language : str) : bytes
TextRenderer
+ render(flashcard : Flashcard, language : str) : str
FlashcardStandard
- flashcard : Flashcard
- renderer : FlashcardRenderer
+ display()
«depends»
«depends»
«depends»
admin
FlashcardAdmin
+ delete_model(request, obj)
«depends»
decorators
FlashcardDecorator
- flashcard : Flashcard
+ get_details()
+ log_flashcard_action(action : str)
models
Flashcard
+ question : Text
+ answer : Text
+ category : str
+ source : str
+ state : str
+ created_at : datetime
+ updated_at : datetime
--
+ __str__()
+ update_statistics(correct : bool)
FlashcardRenderer
+ render(flashcard : Flashcard, language : str)