Plan Projektu MVP: Ekstraktor Przepisów z AI (The Recipe AI Extractor)

Cel: Stworzenie pełnozakresowej, sterowanej zdarzeniami aplikacji webowej do zarządzania bibliotekami przepisów użytkowników oraz automatyzacji ekstrakcji składników i kroków przygotowania z linków zewnętrznych.

Lista Wymaganych Elementów (Checklista Zaliczeniowa):

- Minimum 4 Tabele (4x4 CRUD zaimplementowane)
- Onion/Hexagonal Architecture (Etap 1)
- Obiekty Transferu Danych (DTOs) (Etap 1)
- Renderowanie po Stronie Serwera (SSR) (Etap 2)
- Wzorzec MVC/MVVM (Etap 2)
- Job Scheduling System (Celery/APScheduler) (Etap 2)
- Aplikacja Jednostronicowa (SPA) (Etap 3)
- React/Angular/Vue choice (**React + TypeScript**) (Etap 3)
- RxJS do komunikacji HTTP (Etap 3)
- AWS Lambda / Azure Function integration (Etap 4)

Zalecany Stos Technologiczny (Ostateczny)

Komponent	Technologia	Rationale
Backend / API	**Python (FastAPI / Flask)**	Doskonały do logiki biznesowej i integracji z AI (Gemini API).
Harmonogramer Zadań	**Celery / APScheduler**	Niezbędny do asynchronicznej obsługi zadań w Pythonie.
Frontend / SPA	**React + TypeScript**	Nowoczesny, typowany frontend.
Funkcja Chmurowa	**AWS Lambda (Python Runtime)**	Bezdyskowy (serverless) Ekstraktor AI.
Baza Danych	PostgreSQL / SQL Server	Baza relacyjna dla 5 znormalizowanych tabel.

Schemat Bazy Danych (5 Znormalizowanych Tabel)

Poprawka Normalizacyjna: Użycie tabel **CatalogItem** i **RecipeItem** eliminuje powtarzanie się nazw składników.

#	Entity (Tabela)	Purpose	Relacje Klucz Obcy (FK)
1	**User**	Konta użytkowników.	Brak
2	**Recipe**	Główna encja przepisu.	UserID (FK do User)
3	**CatalogItem**	**KATALOG UNIKALNYCH SKŁADNIKÓW** (tylko nazwy).	Brak
4	**RecipeItem**	**POWIĄZANIE** przepisu z konkretnym składnikiem, jego ilością i jednostką.	RecipeID (FK do Recipe), ItemID (FK do CatalogItem)
5	**ProcessingJob**	Dziennik statusów ekstrakcji AI.	UserID (FK do User)

Detale Kluczowych Kolumn

- **Tabela Recipe:** Będzie zawierać kolumnę **PreparationSteps** typu **TEXT** (zapisany w formacie Markdown dla łatwego renderowania na frontendzie).
- **Tabela RecipeItem:** Zawiera unikalne dane dla przepisu, takie jak **Quantity** (ilość) i **Unit** (jednostka).

MVP Breakdown by Project Stage (Etap)

Etap 1: Web API & Hexagonal Architecture (Weeks 3-6)

Focus: Czysty kod i DTO

1. **Architektura:** W Pythonie stwórz warstwy domain, application (porty/serwisy) i infrastructure (adaptery).

2. **API & DTO:** Zaimplementuj pełne **CRUD (16 operacji)** dla 5 tabel. Użyj Pydantic DTOs dla kontroli danych wejściowych i wyjściowych.

Etap 2: SSR Frontend & Harmonogramowanie (Weeks 7-9)

Focus: Asynchroniczna Obsługa Zadań

1. **SSR Admin Portal (MVC/MVVM):** Utwórz widok renderowany przez serwer (np. Jinja2), który służy jako panel administracyjny do **monitorowania rekordów z tabeli ProcessingJob** (dziennik zadań).
2. **Job Scheduling Setup:** Zintegruj Celery/APScheduler. Żądanie API na link tworzy rekord PENDING w ProcessingJob, a harmonogramer uruchamia **asynchroniczny proces** w tle.

Etap 3: SPA Frontend (Weeks 10-13)

Focus: Dynamiczne UI i RxJS

1. **React + TypeScript:** Główna aplikacja SPA.
2. **Śledzenie Statusu (RxJS):** Po dodaniu linku, aplikacja używa **RxJS Observables** do **pollingu** (cyklicznego odpytywania) API o status JobID.
3. **Wyświetlanie:** Wyświetl przepis po statusie COMPLETED, pobierając Kroki (PreparationSteps) i powiązaną listę Składników (RecipeItem + CatalogItem).

Etap 4: Cloud Integration (Week 14)

Focus: Serverless Extraction Logic

1. **AWS Lambda Deployment:** Wdróż funkcję Python (Ekstraktor AI) jako **AWS Lambda**.
2. **Finalny Proces:** Harmonogramer wywołuje **AWS Lambda** (przekazując URL), która zwraca ustrukturyzowany JSON. Harmonogramer zapisuje dane w tabelach **Recipe**, **CatalogItem** (sprawdzając unikalność) i **RecipeItem**, a następnie oznacza zadanie jako COMPLETED.