## Фаза 1: Планиране и настройка

1. **Определяне на изискванията на проекта**:
   - Създаване на списък с всички функционалности (например качване на изображения, разпознаване на обекти, съхранение и извличане на данни от база данни).
   - Избиране на типа база данни (SQLite) според нуждите..
2. **Настройка на средата**:
   - Инсталиране на необходимите инструменти и библиотеки:
     - Python
     - Django, Flask и Streamlit
     - Библиотеки за разпознаване на изображения (TensorFlow, OpenCV).
     - Софтуер за база данни (SQLite).
   - Създаване на виртуална среда за управление на зависимостите:

     ```
     bash
     Copy code
     python -m venv env
     env\Scripts\activate     # Windows
     ```

3. **Начална структура на проекта**:
   - Създаване на директории за:
     - Бекенд (Django и Flask приложения). (Streamlit)
     - Фронтенд (Streamlit приложение).
     - Модели и помощни модули (логика за разпознаване на изображения).

---

## Фаза 2: Разработка на бекенд

1. **Настройка на базата данни**:
   - Проектиране на схема на базата данни с таблици за:
     - `Objects` (ID на обекта, име и метаданни).
     - `Images` (детайли за файла на изображението, свързани ID на обекти).
   - Свързване на базата данни с Django или Flask (Streamlit).
2. **Django бекенд**:
   - Създаване на Django проект и приложение.
   - Имплементиране на модели за обекти и изображения на базата на схемата на базата данни.
     - Съхранение на резултатите от разпознаването.
     - Търсене и извличане на обекти от базата данни.
3. **Flask API**:
   - Разработване на API за:
     - Приемане на качени изображения от фронтенда.
     - Стартиране на логиката за разпознаване на обекти.
     - Връщане на разпознатите обекти към Django за съхранение.

---

## Фаза 3: Логика за разпознаване на обекти

1. **Избор на предварително обучен модел**:
   - Избиране на модел като предварително обучените TensorFlow модели или OpenCV за разпознаване на обекти.
2. **Създаване на модул за разпознаване**:
   - Разработване на скрипт за:
     - Зареждане и обработка на качените изображения.
     - Откриване и класифициране на обекти в изображението.
     - Присвояване на уникални идентификатори на обекти за съхранение.
3. **Тестване на модула**:
   - Тестване на модула с различни примерни изображения, за да осигури точност на разпознаването и правилна интеграция с Flask API.

---

## Фаза 4: Разработка на фронтенд (Streamlit)

1. **Изграждане на потребителски интерфейс**:
   - Създаване на прост Streamlit интерфейс за:
     - Качване на изображения.
     - Показване на резултатите от разпознаването и предишни данни за обекти.
   - Добавяне на обработка на грешки за невалидни или неподдържани файлове.
2. **Интеграция с бекенда**:
   - Конфигуриране Streamlit за:
     - Изпращане на качените изображения към Flask API.
     - Показване на резултатите, получени от бекенда (разпознати обекти и метаданни).

---

## Фаза 5: Интеграция и тестване

1. **Интегриране на всички компоненти**:
   - Проверка на Flask API, Django бекенд и Streamlit фронтенд работят безпроблемно заедно.
   - Проверка дали записите и четенията от базата данни са правилни и ефективни.
2. **Тестване**:
   - Тестване на цялостната функционалност, като качване на изображения и проверка на:
     - Коректно разпознаване.
     - Съхранение и извличане на данни за обекти.

        - Правилно визуализиране на резултатите в интерфейса.
3. **Дебъгване**:
    - Отстраняване на проблеми или забавяния (например бавно разпознаване, бъгове в интерфейса).

---

## Ред на изпълнение:

1. Настройка на средата и проектиране на базата данни (Фаза 1 и 2).
2. Логика за разпознаване на обекти (Фаза 3).
3. Бекенд API-та и фронтенд интерфейс (Фаза 2 и 4).
4. Интеграция, тестване и дебъгване (Фаза 5).