**Крок 1: Налаштування структури проєкту**

* **Backend**: Для бекенду можна використовувати **ASP.NET Core** (оскільки ти знайома з C#), **Django (Python)** або **Node.js**. Це буде частина, яка обробляє запити на обробку зображень.
* **Frontend**: Використовуй **React**, **Vue.js** або простий інтерфейс з **HTML/CSS/JavaScript**, який дозволить користувачам завантажувати зображення і переглядати прогрес обробки.
* **База даних**: Використовуй **PostgreSQL**, **MySQL** або **SQLite** для зберігання історії задач користувачів та метаданих (назва файлу, розмір, статус обробки).

**Крок 2: Завантаження та валідація зображень**

* **Frontend**: Реалізуй форму для завантаження файлів, де користувачі можуть завантажувати зображення. Використовуй JavaScript, щоб обмежити розмір файлу або тип зображення перед відправленням на сервер.
* **Backend**: Приймай зображення, перевіряй його розмір/тип і зберігай у тимчасову папку для подальшої обробки.

**Крок 3: Обробка зображень**

Для обробки зображень можеш використовувати такі бібліотеки:

* **OpenCV (C# або Python)** для фільтрації (розмиття, підвищення чіткості, конвертація в чорно-біле).
* **TensorFlow або PyTorch (Python)**, якщо плануєш додати базове розпізнавання об'єктів.
* **SkiaSharp (C#)** для загальних маніпуляцій із зображеннями.

Залежно від запиту, сервер виконає наступне:

* Застосує потрібні фільтри (розмиття, підвищення чіткості).
* Конвертує зображення в чорно-білий формат (якщо потрібно).
* Додатково може виконати розпізнавання об'єктів (якщо додаш машинне навчання).

**Крок 4: Відстеження прогресу задач**

* **Звіт про прогрес**: Оскільки обробка зображень може займати час, реалізуй етапи задачі, такі як "завантаження", "обробка", "фільтрація", тощо. Сервер оновлюватиме статус на кожному етапі. На клієнті можеш використовувати WebSockets або періодичне опитування для отримання оновлень про стан задачі в реальному часі.

**Крок 5: Історія задач і скасування**

* Зберігай історію оброблених зображень у базі даних (наприклад, назву файлу, час обробки, посилання на результат).
* Додай можливість перегляду історії задач, скасування поточних задач та запуску нових.

**Крок 6: Авторизація з HTTPS**

* Реалізуй авторизацію користувачів за допомогою **JWT (JSON Web Tokens)** або **OAuth**. Забезпеч використання **HTTPS** для захисту даних користувачів, таких як завантажені файли та облікові записи.

**Крок 7: Балансування навантаження і масштабування**

* Налаштуй як мінімум два сервери (це можуть бути локальні або хмарні сервери). Використовуй **load balancer** (наприклад, **NGINX** або зворотний проксі у **ASP.NET**) для розподілу запитів на обробку зображень між серверами.
* Кожен сервер виконуватиме однакові задачі з обробки зображень, а балансувальник навантаження стежитиме за розподілом задач.

**Крок 8: Тестування та деплоймент**

* Перевір додаток на різних розмірах та типах зображень. Переконайся, що балансування навантаження працює коректно.
* Задеплой додаток на хмарні платформи, такі як **Azure**, **AWS** або **Google Cloud**, щоб забезпечити масштабованість.