ChatGPT

Проект сервиса для переработки контента карточек товаров (Яндекс Маркет)

Общая концепция и цели проекта

Цель проекта – разработать **сервис с ИИ-инструментами для массовой переработки контента товарных карточек** на Яндекс Маркете. Речь идёт прежде всего об автоматизированной обработке **основных фотографий товаров** (первое изображение в сниппете), чтобы привести их к единым требованиям платформы. Планируется реализовать сервис с помощью AI-моделей (Claude, GPT) в среде Cursor, а также интеграции с внешними API (например, через платформу **fal.ai** для генеративных моделей).

В рамках проекта предстоит решить несколько задач:

- **Обработка существующего контента** (миллионы фото уже размещённых товаров) – автоматическое улучшение и приведение изображений к стандарту. Для этого сначала нужно проанализировать имеющиеся фото и классифицировать их по сложности обработки.
- **Поддержка нового контента от мерчантов** – создание понятных **гайдов (инструкций)** для продавцов по подготовке фото, а также автоматические инструменты проверки/коррекции новых изображений.

Итоговое **целевое состояние** для каждого типа контента должно быть чётко определено: мы должны описать требования к главному фото в каждой категории (Home, СЕНАС, Beauty, Fashion и подкатегории) – например, фон, композиция, наличие/отсутствие моделей или инфографики, допустимые ракурсы и пр. На первом этапе концентрируемся на технической стороне обработки существующих фотографий, а затем можно переходить к разработке гайдов и модулей модерации для новых.

Кратко о требуемой обработке изображений: сервис должен уметь брать любое входное фото товара и преобразовывать его в стандартное выходное изображение. Необходимо: вырезать фон (сохранив объект), *при необходимости* восстановить или сгенерировать тень от объекта, откорректировать цветовую тональность (чтобы товар выглядел естественно, а фон соответствовал требуемому цвету), увеличить разрешение до стандарта (не ниже 1200×1600 или 1600×1600), подготовить версии на светлом и тёмном фоне (два варианта фона с заданными цветами #F5F4F2 для светлой темы и #D1CECB для тёмной). Дополнительно рассматривается возможность **генерации вращения товара** (новые ракурсы), хотя это опциональная и более сложная задача. В категории Fashion планируется особый подход: использование нейросети для автоматического “надевания” одежды на модель (виртуальный манекен), чтобы получить реалистичные фото одежды на человеке, сохраняя пропорции и принты (если товар уже сфотографирован на модели, то модель не заменяется).

Чтобы обеспечить универсальность, сервис должен учитывать форму/ориентацию товара при компоновке изображения: - **Вертикальные (высокие) объекты** – размещаются так, чтобы их верх и низ совпадали с границами кадра (вписываются по высоте). Для таких объектов планируется квадратный формат полотна 1:1 (например, 1600×1600) – т.е. объект будет касаться

верхнего и нижнего края сетки, а по бокам может быть фон. - **Квадратные или низкие/широкие объекты** – размещаются внутри квадрата, не доходя до краёв кадра. Для них фон целесообразно делать вертикальным прямоугольником 3:4 (например, 1200×1600), чтобы сверху и снизу оставались отступы. Иными словами, если товар не вытянут сильно по вертикали, его располагают в центральной области, а сам холст делает более вытянутым.

Такой подход обеспечит визуальную унификацию: высокие товары будут на квадратных изображениях (не оставляя больших полей сверху/снизу), а более компактные – на вертикальных прямоугольниках (чтобы не было слишком много пустого пространства по сторонам). В обоих случаях важна **центровка и масштабирование**: товар должен занимать большую часть кадра, но **«запретные зоны»** (если они определены гайдами – например, определённые поля от краёв для логотипов или текста) не должны нарушаться. После обработки первое фото товара будет соответствовать всем требованиям площадки по композиции, фону и качеству.

Анализ существующих решений (ресёрч рынка)

Перед тем как реализовывать свою систему, полезно изучить аналогичные решения на рынке. За последние пару лет появилось множество AI-сервисов, ориентированных на **автоматизацию обработки фотографий для e-commerce**. Вот некоторые заметные примеры и их возможности:

- **Claid.ai (Let's Enhance)** – комплексный AI-сервис для продуктовой фотографии. Он умеет автоматически улучшать качество и разрешение изображений, убирать фон и заменять на заданный, корректировать цвета и освещение, причём заточен именно под товары. Claid заявляет, что их ИИ способен проверять и править фото под требования площадок всего за 2–3 секунды, снижая затраты на редактирование на 80% [1] [2]. Сервис ориентирован на массовую обработку (через API) и сохранение **деталей продукта и брендинга** – например, логотипы, характерные формы товара не размываются при обработке [3] [4]. Claid.ai позволяет получать **студийное качество** снимков без ручного фотошопа, предоставляя инструменты: улучшение чёткости и цвета, повышение разрешения, удаление фона и генерация новых фонов по шаблонам или по описанию [5]. Такой сервис фактически решает схожую задачу – приведение пользовательских/продавецких фото к профессиональному стандарту в автоматическом режиме.

- **remove.bg** – узкоспециализированный сервис, который стал стандартом для **автоматического удаления фона**. Его нейросеть за 5 секунд отделяет объект от фона с высоким качеством, справляясь даже с сложными краями (волосы, мелкие детали) [6] [7]. Remove.bg широко используется в e-commerce: единообразный фон без лишних деталей повышает конверсию за счёт фокусировки внимания на продукте [8]. К примеру, площадки вроде Amazon требуют чисто-белый фон, eBay – белый или светло-серый [9], и автоматическое удаление фона облегчает соответствие этим правилам. По заявлениям разработчиков, обработка через remove.bg экономит массу времени (вручную вырезать фон могло бы занять минуты на каждое фото, а ИИ делает это за секунды) [10]. Есть возможность пакетной обработки тысяч изображений и интеграции через API в рабочий поток [11] [12]. Этот сервис демонстрирует востребованность функций авто-вырезки фона и подтвердил, что **унификация фона** (например, белый или нейтральный цвет) повышает доверие покупателей и удобство просмотра товаров [8].

- **DeepImage, Pixelcut, VanceAI и др.** – ряд облачных сервисов, предлагающих сходные функции (upscale, удаление фона, очистка дефектов, улучшение качества). Например, Deep-Image.ai позиционируется как *«Photo Enhancer for eCommerce»*, заявляя об улучшении

чёткости, увеличении разрешения и генерации новых фонов для товарных фото [13]. **Pixelcut.ai** и **Pixlr** предлагают бесплатные веб-инструменты на базе ИИ для обрезки фона и апскейлинга [14] [15]. Такие инструменты подтверждают, что технически *каждый этап* (удаление фона, апскейл, цветокоррекция) может быть реализован моделью или скриптом. Однако, по отдельности они решают только часть задач, и наш проект нацелен на объединение всех шагов в единый поток.

- **WeShop AI** – комплексное решение, ориентированное на **генерацию контента для продуктов**. Помимо удаления фона и улучшения качества, WeShopAI делает интересный упор на **виртуальные примерочные и модели**. Этот сервис позволяет загружать фото продукта (например, одежда на манекене или просто вещь) и автоматически накладывать на AI-сгенерированного человека-модель для «on-model» фото [16] [17]. По сути, встроена функция **Virtual Try-On**: можно создать реалистичного человека определённого типа (на выбор есть разные AI-модели или можно обучить свою) и примерить на него одежду за считанные секунды [18]. В обзорах отмечается, что WeShop AI предоставляет весь спектр функций: и удаление фона, и увеличение разрешения, и генерацию фоновых сцен, и даже магическое удаление нежелательных объектов на фото [16]. Особенно интересна возможность **создавать собственные AI-модели** брендов – т.е. тренировать ИИ на изображениях товаров, чтобы затем генерировать новые ракурсы или новые фоновые сцены с ними [19]. Генерация одного реалистичного снимка с моделью занимает около минуты [16]. Такой сервис показывает потенциал экономии: вместо дорогостоящих фотосессий с моделями можно *за доли стоимости* получать правдоподобные фото одежды на людях. Ключевой момент – WeShop (и аналогичные) позволяют **масштабировать** генерацию контента: их инфраструктура рассчитана на потоковое создание изображений (в обзоре упоминается, что они выдают результаты за 30 секунд, пакетами по нескольку изображений) [20] [21].

- **Flair.ai, Pebblely, Mokker** – новые стартапы, предлагающие **генеративную фотосъёмку товаров**. Например, **Pebblely** позволяет загрузить вырезанное изображение товара и сгенерировать для него фон по описанию или из шаблонов, причём автоматически добавляются тени и отражения для реализма [22] [23]. Они поддерживают удобное редактирование: можно менять размер холста, позицию продукта, убирать лишние объекты, накладывать логотипы – всё средствами ИИ [24] [25]. Особый акцент – **массовое создание контента**: Pebblely заявляет о генерации до 200 000 изображений в сутки по API [26], что показывает возможность обработки миллионов фото при достаточных вычислительных мощностях. **Flair.ai** аналогично ориентирован на командную работу с AI-дизайном: там есть и «AI Fashion Models» для примерки одежды на моделях, и генерация рекламных креативов, и API для интеграции в конвейеры [27] [28]. Появление таких инструментов подтверждает тренд: **ритейл стремится автоматизировать обработку визуального контента**, добиваясь экономии времени и единообразия стиля. Например, по данным компании Botika (специализируется на AI-моделях для фэшн) использование их нейросетей позволило сократить расходы на фотосъёмку на 90%, ускорить вывод товаров на рынок в 10 раз и слегка повысить конверсию и другие метрики [29] [30].

Вывод по рынку: Уже существуют мощные решения, способные закрыть часть или все необходимые функции – от простого удаления фона (remove.bg) до комплексной генерации готовых фото под нужный формат (Claid, WeShop, Pebblely и др.). Эти сервисы часто доступны через API и могут обрабатывать тысячи изображений **в автоматическом режиме**. Наш проект во многом аналогичен им, но есть нюансы: необходимо учесть **специфические требования Яндекс Маркета** (цвет фона, сетки компоновки, особенности разных категорий) и обеспечить более гибкую интеграцию в нашу инфраструктуру. Возможно, часть решений мы можем

переиспользовать: например, использовать готовые модели для вырезания фона или апскейла через API (для быстрого старта), а затем заменить или дообучить под свои данные. Но ключевое преимущество собственного сервиса – полный контроль над обработкой и возможность тонкой настройки под требования, которые могут отличаться от общепринятых (например, два варианта цветового оформления под светлую/тёмную темы – редкий кейс, который готовые продукты могут не поддерживать “из коробки”). Таким образом, мы двигаемся в русле современных трендов, но адаптируем решения под свои нужды.

Инструменты и технологии для реализации

Для реализации сервиса потребуется сочетание нескольких технологий: **компьютерное зрение, классические алгоритмы обработки изображений и генеративные нейросети**. Рассмотрим инструменты для каждого этапа нашего конвейера:

1. **Удаление фона (сегментация объекта):** это первый и критично важный шаг.
2. *Варианты реализации:* готовые API (remove.bg, ClipDrop от Stability AI, Claid API) или развёртывание собственной модели сегментации. Для собственного решения есть успешные открытые модели, например **U^2-Net** – модель сегментации, которая лежит в основе многих открытых тулов для удаления фэк. Существует библиотека `rembg` (Python), которая позволяет легко применить эту модель к изображению и получить прозрачный фон. Также можно рассмотреть **Segment Anything (SAM)** от Meta – универсальную модель сегментации. Однако SAM требует указания примерной области или точки внутри объекта, что затрудняет полностью автоматическое применение. Более специализированные под **salient object detection** модели (как U2-Net) умеют сами находить главный объект на фото. В контексте товарных фото, как правило, главный объект очевиден (один крупный предмет по центру), поэтому авто-сегментация работает хорошо.
3. *Качество и детали:* важно, чтобы при вырезании не терялись мелкие детали товара. Здесь помогут режимы высокоточного маттинга: некоторые инструменты (как remove.bg) делают несколько проходов для тонких структур вроде волос, прозрачных элементов и пр. Можно предусмотреть этап **улучшения маски**: сначала грубое выделение, потом **refinement** – например, с помощью морфологических операций или специализированных моделей для прозрачностей. Согласно описаниям remove.bg, их ИИ *«handles even the most challenging edges, tiny details... in an exceptional manner»* [7] . Нам следует стремиться к сопоставимому качеству, возможно обучив модель на датасете товарных фото, где есть истина по маскам.

4. *Инструментарий:* Python + OpenCV/Pillow для базовых операций, `rembg`/PyTorch для запуска модели U2-Net локально. Альтернативно, сервисы типа ClipDrop имеют готовый **background removal** API. Выбор будет зависеть от требований по скорости и стоимости: при миллионах изображений, внешний API может обойтись дорого, поэтому выгодно иметь хотя бы опцию локальной обработки на своих серверах или в облаке с оплаченной вычислительной мощностью.

5. **Сохранение деталей объекта:** после сегментации получаем контур объекта. Здесь надо убедиться, что **все части товара на месте**. Например, если на оригинальном фото несколько предметов (комплект) – модель должна выделить их все как единое целое. В случае ошибок (что-то не вырезалось или наоборот лишнее отхватилось) желательно либо доработать маску, либо иметь правило: например, брать *самый крупный контур* за объект. В ранжировании сложности мы, вероятно, разделим изображения на:

6. Простые: один чёткий объект на более-менее однородном фоне (легко вырежется правильно).

7. Средние: объект на сложном фоне, но контрастный (ИИ справится, возможно с мелкими огрехами).
8. Сложные: несколько разных объектов, модель, текст на фоне и т.п. (требует доп. обработки или даже ручной проверки).

На старте можно реализовать простую проверку: если маска разорвана (несколько больших отдельных областей), возможно, стоит объединить их (если объекты близко) или хотя бы предупредить, что контент сложный. Наша цель – не пропадали части товара. Инструментально можно использовать **контурный анализ** OpenCV: подсчитать, сколько областей в маске и их размеры, сравнить с ожидаемым. Кроме того, **сохранение цветовых деталей**: при вырезании в PNG важно не потерять полупрозрачности (например, стекло, тени от изначального фото если мы хотим их сохранить). Вероятно, лучше изначально вырезать на прозрачный слой, а не сразу красить фон, чтобы гибко добавить свой фон и тень.

1. **Генерация или восстановление тени:** когда объект вырезан и помещён на новый фон, часто требуется тень, чтобы картинка выглядела объёмно и натурально. Есть два подхода:
2. *Графический (правило):* взять силуэт объекта (маску), сместить её вниз/вбок как нужно (имитируя падающую тень), залить тёмно-серым и размыть (Gaussian Blur) – получится **простая падающая тень**. Этот метод реализуем скриптом (OpenCV/Pillow). Нужно подобрать параметры: направление и размер смещения зависят от предполагаемого освещения (обычно свет сверху, тень вниз под объектом). Также можно просто сделать тень прямо под объектом (контактная тень), особенно если требование – ощущение, что товар стоит на плоскости. Например, Clipping Magic (онлайн-редактор) в функции add shadow просто предлагает регулировать прозрачность и размытие тени вручную [31], но мы можем это автоматизировать.
3. *AI-подход:* использовать нейросеть для более реалистичных теней. Некоторые генеративные инструменты (как Pebblely) заявляют, что автоматически создают **подходящие отражения и тени** для сгенерированных сцен [22] [32]. В нашем случае фон однотонный, но можно применить, например, модель **normal map estimation** – по изображению объекта попытаться предсказать карту высот и освещения, чтобы тень легла физически правильно. Это, вероятно, избыточно сложно. Поэтому начнём с простого метода *drop shadow*. Главное – не переборщить: тень должна быть умеренной, соответствовать материалу (у прозрачных объектов тень светлее и размытее).
4. Желательно сделать тень **опциональной и настраиваемой**: для некоторых категорий (например, **мода** – одежда на модели) тень от фигуры не нужна, или для товаров, которые изначально “висят” (например, украшения на белом фоне) тень может быть неуместна. Значит, на уровне правил можно решать: для **Home/CEHAC** (бытовая техника, электроника) – тень под объектом придаст реализму, для **Beauty** (флаконы, косметика) – скорее тоже нужна легкая тень на плоскости, для **Fashion** – если модель стоит на ногах, тень естественно нужна на земле; если предмет одежды отдельно, возможно, тень минимальная.

5. Инструментарий: Pillow (ImageFilter.GaussianBlur), OpenCV (filter2D), numpy для манипуляций с альфа-каналом. Этот шаг не требует стороннего API, достаточно собственных функций.

6. **Цветокоррекция и гармонизация:** на этом этапе объект уже на новом фоне, нужно удостовериться, что **цвета выглядят естественно** и соответствуют действительности. Задачи:

7. **Сохранить цвет товара.** Категорически важно не исказить цвет продаваемого предмета (например, одежда должна остаться того же оттенка, иначе покупатель будет

разочарован). Поэтому цветокоррекция должна быть осторожной: скорее всего, не будем сильно менять **баланс белого** товара. Но что если оригинальное фото было с неправильным освещением (желтизна или тусклость)? Здесь можно применить автоматические улучшения: например, алгоритм выравнивания белого (gray world assumption или более продвинутый Deep White Balance), или просто лёгкую коррекцию уровня контрастности.

8. **Единый фон.** Мы уже знаем точные целевые цвета фона: светлый #F5F4F2 и тёмный #D1CECB. Эти цвета, видимо, выбраны дизайнерами Яндекс Маркета для оптимального вида на светлой и тёмной темах сайта. Нужно убедиться, что итоговое изображение имеет ровный фон именно этого цвета (без посторонних оттенков). То есть после наложения объекта и тени, весь фон залить заданным цветом (учитывая, что тень – полупрозрачная тёмно-серая, она по сути будет смешиваться с цветом фона).
9. **Общая гармония и освещение.** Объект, помещённый на нейтрально-серый фон, может выглядеть слегка “вырезанным”, если на нём самом были цветные блики от старого окружения. В идеале, можно применить нейросеть, которая **адаптирует освещение** объекта под новый фон (что-то похожее есть в ClipDrop *Relight*). Но на нашем однородном фоне важнее, чтобы не было заметно, что фон новый. Возможно, лёгкое размытие края (feather) или внутренняя тень по периметру объекта поможет. Также можно применить функцию **HDR toning** или просто кривыми подтянуть яркость товара, чтобы он не был слишком тёмным или пересвеченным относительно фона.
10. Инструментарий: OpenCV (например, cv2.cvtColor, cv2.equalizeHist для яркости), PIL ImageEnhance (контраст, цветность). Можно задействовать библиотеку **ImageAI** или **OpenCV dnn** для обнаружения, не исказился ли цвет – но скорее достаточно эмпирических настроек.

11. **Примечание:** Некоторые сервисы (Claid) прямо предлагают *“fix colors”* – т.е. привести цвета к стандарту [5] . В e-commerce под этим может пониматься убирание сильных фильтров, приведение фона к чистому белому и т.п. Нам важно, чтобы вся партия изображений выглядела единообразно по тону (например, если фон серовато-бежевый #F5F4F2, то и тени/объект не должны иметь случайного цветового оттенка, кроме естественного цвета товара). Это повысит **гармонизацию каталога** – как говорится в материалах remove.bg, единый стиль фонов *«убирает отвлекающие детали»* и позволяет покупателю сконцентрироваться на товаре [8] .

12. **Апскейл (увеличение разрешения):** технические требования Яндекс Маркета – главное фото не менее 1200×1600 (или квадрат 1600×1600). Многие старые/присланные изображения могут быть меньшего размера или сжатые. Поэтому встроим этап повышения разрешения с помощью AI-суперрезолюшна:

13. **Модели повышения детализации:** Существует несколько открытых моделей, напр. **ESRGAN / Real-ESRGAN** для общего улучшения фото. Они умеют воссоздавать недостающие детали при увеличении, обучены на фотодатасетах. Также есть специализированные модели для графики, но нам нужна универсальная для фотографий товаров. Real-ESRGAN можно запустить локально (PyTorch) или через API (например, у DeepImage и многих сервисов есть функция upscale).
14. **Интеграция через fal.ai:** Платформа fal.ai предоставляет доступ к множеству генеративных моделей через API. Возможно, там уже есть готовый эндпоинт для апскейла (надо проверить библиотеку моделей – fal.ai заявляет «Run image, 3D, video AI models 4x faster»). Если есть ESRGAN, Stable Diffusion Upscaler или что-то подобное на fal.ai, можно использовать их.

15. **Процесс:** Для надежности, лучше увеличивать постепенно. Например, если нужно увеличить в 2-3 раза, Real-ESRGAN справится за один проход. Если вдруг нужно сильнее, можно несколько раз. Но учитывая, что у нас требуется конечный размер ~1600px по большей стороне, большинство исходников, наверное, уже близки. Апскейл пригодится скорее для мелких изображений (скажем, 800px -> 1600px).
16. **Внимание к качеству:** После апскейла можно ещё раз проверить резкость. Иногда ИИ создает артефакты или «рисует» несуществующие детали. Лучше подобрать модель: например, ESRGAN имеет варианты для разных содержимых (для лиц, для текстур). В контексте товаров универсальная модель должна подойти. В Claid.ai отмечалось, что после их улучшалки изображение стало *«кристально чётким»* [33] [34] – мы стремимся к тому же. Особенно критично это для **текста на продуктах** (этикетки, логотипы): ИИ должен сохранить читабельность. Claid хвалится, что их модель **сохраняет логотипы и надписи** [3] – возможно, она комбинирует распознавание + дорисовку. Нам можно заложить проверку: если на товаре есть текст, после апскейла OCR-двигатель (типа Tesseract) мог бы проверить читаемость. Но это скорее улучшение на будущее.

17. Инструментарий: подключить **Real-ESRGAN** через Python (есть репозитории и даже пакеты), либо использовать готовый сервис. Также есть API у **Let's Enhance** (компания Claid) – они продают кредиты на обработку (например, $59 за 1000 снимков [35]). На миллион фото это дорого, поэтому упор на open-source решение на своих серверах.

18. **Формирование выходных вариантов:** после выполнения всех обработок над базовым изображением, сервис должен **собрать финальные версии**:

19. **Два цветовых варианта фона:** как указано, нужно выпустить изображение на светлом фоне (#F5F4F2) и на тёмном варианте (#D1CECB). Это значит, по сути, заменить слой фона и, возможно, цвет тени (тень на более тёмном фоне может быть менее заметна, возможно её прозрачность надо чуть увеличить для контраста). Два файла можно генерировать параллельно. Логично хранить их с явным обозначением темы (например, `[SKU]_main_light.jpg` и `[SKU]_main_dark.jpg`).
20. **Размер и формат:** убедиться, что полученные изображения соответствуют нужному разрешению. Если у нас холст 1600×1600 или 1200×1600 – сохранить именно в этих размерах (или чуть больше, если решили делать с запасом "и выше", главное сохранить соотношение). Формат JPEG предпочтителен для сайта (баланс качества и веса файла). Но на случай прозрачности (если вдруг временно сохраняем PNG), итог всё равно будет непрозрачный фон, так что JPEG нормально.
21. **Оптимизация:** можно сразу прогнать сохранённые JPEG через оптимизатор (например, mozjpeg) чтобы уменьшить вес без заметной потери качества – при миллионах картинок это значительно экономит место и ускоряет загрузку на стороне пользователей.

22. **Хранение/выгрузка:** На период разработки не так важно, куда складывать – хоть локально, хоть на Google Drive. Но в перспективе интеграция с хранилищем компании или Яндекс Облаком, чтобы автоматизировать заливку на Маркет.

23. **Итеративный контроль качества:** было пожелание – получать промежуточный результат после каждого этапа. Это очень полезно в отладке:

24. Сервис можно настроить так, чтобы сохранять (или отображать в интерфейсе) изображение после вырезания фона (например, объект на прозрачном фоне или на шахматном фоне), потом после добавления тени, потом после цветокоррекции. Благодаря этому, команда сможет сразу видеть, где происходит сбой (например, после сегментации

пропал кусок объекта – видим на первом же превью и можем оперативно поправить настройки маски).

25. Возможно, для этого реализуем *режим отладки*, когда при обработке одного товара по запросу возвращаем ZIP с набором изображений всех стадий. При массовой обработке такое хранить накладно, поэтому по умолчанию – только финал, но в режиме разработки – полная раскладка.

26. Реализовать показ этапов можно даже через Telegram-бот: после каждого шага отправлять изображение. Но это по желанию – на первом этапе можно просто логировать.

27. **Опционально: генерация вращения товара (3D):** Это сложная задача, но раз она обозначена, отметим возможные пути:

28. Без специальных данных получить новый ракурс товара сложно. Один из подходов – попытаться воспользоваться генеративной моделью вроде **Stable Diffusion** с ControlNet: подаём исходное изображение товара и какой-то ориентир нового угла, генерируем новый ракурс. Но качество и идентичность товара не гарантируются (модель может изменить форму или детали).
29. Более надёжный профессиональный подход: использование **NeRF (Neural Radiance Fields)** или стерео-моделей. Если бы у нас было несколько фотографий товара под разными углами, можно было бы обучить 3D-модель и рендерить вращения. Но у нас, вероятно, только по 1 фото. Есть исследования по восстановлению 3D-формы из одного изображения (категорийные, например для обуви или мебели), но пока это не готово для массового использования без существенных ошибок.
30. Практически, можно реализовать хотя бы **поворот на плоскости** (например, зеркальное отражение изображения – чтобы показать левый/правый вид симметричного предмета). Или если товар плоский, можно «наклонить» (но это уже будет графическое искажение, не всегда приемлемо).
31. Вероятно, эту функцию отложим на будущее, либо ограничимся integration с внешним сервисом, если появится доступный инструмент. (Например, компания **Cappasity** раньше делала 3D-виджет для товаров, но там требовалась съёмка 360°). Сейчас можно мониторить появления новых AI, которые умеют генерировать 3D-объекты по фото.

32. Главное, что архитектура нашего сервиса модульная – можно потом встроить модуль генерации новых ракурсов, если он появится.

33. **Особые случаи категорий (Fashion и др.):** Отдельно рассматривается категория **Fashion (одежда)**. Здесь первые фото товаров могут быть:

34. На модели (живой человек или манекен).
35. Без модели (лишь вещь на вешалке или разложена). Наш сервис должен уметь в первом случае просто **вырезать фон, не трогая саму модель**, во втором – **«надевать» вещь на цифровую модель**. Для реализации последнего можно использовать подход, аналогичный Botika/WeShop:
    - Иметь несколько предобученных **нейронных манекенов** (мужчина, женщина, разный тип фигуры) или обучать под свой бренд.
    - Через генеративную сеть (например, Diffusion или специальную модель для виртуальной примерки) совмещать изображение одежды и модель. Одно из решений – модель типа **TryOnGAN** или **Virtual Try-On networks**, которые существуют в исследовательской сфере. Коммерческие продукты (Botika) добились уже высокого качества: у них из фото вещи на манекене + выбранная модель

генерируется реалистичное фото, причём узоры и форма сохраняются [17]. Они заявляют экономию времени и денег на фотосъёмке до 90% [29].
    - Возможно, на начальном этапе мы не будем сразу разрабатывать свою GAN для одежды, а интегрируем через API: например, у **fal.ai** или других платформ могут быть готовые модели для виртуальной примерки. Либо можно воспользоваться **Diffusers** (HuggingFace) где есть модель *Stable Diffusion* fine-tuned на fashion try-on.
    - Если товар уже пришёл на модели, то мы оставляем модель, лишь чистим фон/улучшаем качество. Здесь важно не «обрезать» голову/ноги модели при компоновке – т.е. вертикальное выравнивание по верх/низ кадра особенно актуально.
36. Для категорий **Beauty** (косметика) и **Home/CEHAC** (товары для дома, электроника) специфических AI-генераций не требуется, кроме, возможно, случаев когда хочется сгенерировать декоративный фон. Но по заданию, фон у нас будет стандартизованным (цветным нейтральным), а не сложной сценой, поэтому основной упор – тень, правильный цвет. В перспективе можно подумать: например, для косметики иногда делают **инфографику** на главной фото (показывают кисточку рядом с флаконом и каплю крема и т.д.). Такие композиции, если нужны, скорее всего будут подготовлены дизайнерами вручную или же можно генерировать через тот же Stable Diffusion с описанием. Но это вне рамок минимально рабочего продукта.
37. **Учет требований подкатегорий:** Вы упоминали, что внутри категорий есть разные требования. Например, возможно, для крупной бытовой техники фон допускается чуть темнее или нужен небольшой отражающий пол под объектом; для мебели – возможно, объект должен занимать не менее X% кадра; для косметики – разрешены декоративные элементы (цветочки, брызги) на первом фото или нет. Эти нюансы нужно собрать из внутренних документов или разработать совместно с категорийными менеджерами. Далее их можно формализовать в виде параметров для нашего сервиса (например, флаг «allow_props» – можно ли оставлять доп. предметы на фото; или «require_scale_marker» – нужен ли линейка/размер). На первом шаге можно игнорировать, но иметь в виду, что архитектура должна позволять ветвление логики по категории товара.

Предлагаемая архитектура и этапы разработки

Теперь сведём всё вместе. Проект разобьём на **модули**, каждый из которых реализует определённый шаг обработки. Ниже представлен примерный **поток данных** от сырого изображения к результатам, а также план реализации по этапам:

1. **Модуль классификации сложности (предобработка):** скрипт, который анализирует входное изображение и решает, по какому сценарию его обрабатывать. На основе простых эвристик (размер изображения, наличие прозрачности, однотонность фона, число объектов) можно присвоить уровень сложности. Например, с помощью OpenCV проверить гистограмму фона – если почти однородная, случай простой; если много разных цветов и деталей по всему кадру, случай сложный. Также можно использовать предварительно обученный **классификатор** (CNN), который выдаёт вероятность, что фото «студийное» либо «обычное». На старте это необязательно – можно вручную проглядеть несколько сотен изображений и эмпирически определить правила. **Цель этого модуля**: пометить сложные случаи, чтобы либо отправить их на усиленную обработку, либо отложить для ручной проверки, а простые пропускать автоматически. Например, если фото содержит человека (что можно определить моделью обнаружения людей), а категория не Fashion, то это отклонение – такое фото возможно пропустим через алгоритм, но флагнем, что там посторонний элемент.

Реализация: написать функцию `assess_image(image) -> complexity_level`, использовать OpenCV, возможно, библиотеку типа PIL.Exif для метаданных (если снимок слишком мал в пикселях – тоже проблема).

1. **Модуль вырезания фона:** функция `remove_background(image) -> (object_mask, cutout_image)`. Здесь можно интегрировать `rembg` (который под капотом использует модель глубокого обучения). На вход – исходное изображение, на выход – бинарная маска и изображение с прозрачным фоном.

Разработка: - Установить и протестировать несколько подходов (rembg vs собственный inference U2Net vs API remove.bg) на различных примерах из наших категорий. Отладить параметры: например, у rembg можно выбирать модель (есть модели побольше для качества, поменьше для скорости). - Сделать пост-обработку маски: заполнить дыры внутри объекта, отфильтровать мелкий мусор (иногда алгоритм может оставить кусочки фона отдельно – их убирать). - Проверить сложные случаи: прозрачный товар (стеклянная ваза) – маска должна быть полупрозрачной. Возможно, потребуется маттинг-алгоритм (например, **Alpha Matting** by KNN or other). Если инструмент не справляется, можно просто пока оставить как есть или пометить такие случаи. - **Выход:** объект без фона (например, в формате PNG с альфа-каналом). Ещё и саму маску хранить, т.к. она понадобится для тени.

1. **Модуль компоновки и холста:** функция `compose_on_canvas(cutout_image, mask, category) -> image_on_bg`. Задача – поместить вырезанный объект на новый холст нужного размера/соотношения.

Логика: - Определить габариты объекта (по маске вычислить bounding box). Отношение ширины к высоте объекта подскажет, вертикальный он или более квадратный. - В зависимости от этого, выбрать формат: - Если высота >> ширины (скажем, объект вытянут более чем на 1.3–1.5 раза) – **фон 1:1**. Создаём квадратное полотно (1600×1600) нужного цвета (пока берём светлый вариант, тёмный сделаем копированием потом). - Если объект близок к квадрату или широковат – **фон 3:4** вертикальный. Создаём, например, 1200×1600 (или 1600×2133, если хотим 4:3? Но они указали 1200×1600 явно, то есть 3:4 вертикально). Вопрос: можно ли использовать 1600×1600 даже для квадрата? Да, допускается «и выше», но лучше придерживаться заданных стандартов для единообразия. - Расположить объект: - Для квадратного холста: масштабировать объект так, чтобы он почти касался верхнего и нижнего края (как и требовалось для вертикальных предметов). Оставляем небольшой отступ (например, 5-10px) чтобы не прям впритык. По горизонтали он будет с полями. - Для вертикального холста: вписать объект в центр, причем если объект не вытянут, он не должен касаться краёв. Можно взять максимальное увеличение, при котором объект целиком вписывается **внутрь квадрата**, вписанного в этот прямоугольник. Т.е. оставляем примерно равные поля сверху и снизу (inner square concept). По бокам может доезжать почти до краёв если объект широкий. - Таким образом, мы соблюдем правило: высокий товар – тянется от верха до низа кадра (на квадрате), а более низкий – окружён полем на вертикальном прямоугольнике. - После определения размеров – масштабируем вырезанное изображение объекта (с сохранением пропорций) до нужной высоты (для квадрата – почти высота холста, для прямоугольника – по ширине/высоте как решено) и отрисовываем его на новом фоне. Пока без тени. - **Вывод:** изображение уже заданного размера, с объектом на однородном фоне (без тени пока). Этот модуль по сути отвечает за **правильную кадрировку** по требованиям сетки.

1. **Модуль генерации тени:** функция `add_shadow(image_on_bg, mask, params) -> image_with_shadow`. Здесь `image_on_bg` – это результат предыдущего шага (объект на фоне). Маска объекта применяется, чтобы нарисовать тень.

Детали реализации: - Создать копию маски объекта. Можно взять только сам *силуэт* (чёрно-белую маску). - Заполнить маску сплошным чёрным (или тёмно-серым). Затем сместить её: например, на 10% от высоты вниз и 5% вправо – имитируя источник света слегка сверху слева. Или просто вниз, если предполагаем свет прямо над объектом. - Размыть маску (GaussianBlur) с радиусом, зависящим от предполагаемой «высоты» объекта над поверхностью. Например, у обуви или техники тень резкая ближе к объекту и размытие по краям. Мы можем немного схитрить: сделать две тени – одну очень размытую большую с непрозрачностью скажем 30% (окружная рассеянная тень) и вторую поострее прямо под объектом (контактная тень) с непрозрачностью 60%. Но на первых порах хватит одной средней. - Наложить эту тень под слой объекта. Так как у нас фон однотонный, можем рисовать тень прямо на фон (например, комбинируя изображения с альфа-каналом: mask_blur as alpha for a gray color). - Контролировать интенсивность: цвет тени можно взять не чисто чёрный, а примерно #000000 с прозрачностью ~0.5 (50%) или эквивалентно заполнить маску серым 128 и не делать прозрачность. - Убедиться, что тень не выходит за границы изображения (если объект близко к краям, смещение может «обрезать» тень – но это не страшно, просто тень уйдёт за кадр). - Результат – картинка, где товар как будто стоит/лежит на однородной поверхности, есть ощущение объёма. - *Пример:* в обзоре WeShop автор отметил, что при генерации сцены с бутылкой сервис **сам добавил тень от бутылки на песке** [36] – мы делаем аналогичное, только на однородном фоне. - Параметры (params) могут задаваться от категории: скажем, для **Fashion** (человек в полный рост) тень будет длиннее и более размытой, для **Beauty** (маленький флакон) тень короткая и резкая. Но можно начать с одного универсального и потом при необходимости ветвить.

1. **Модуль цветокоррекции:** функция `harmonize_colors(image_with_shadow) -> image_corrected`. Хотя фон уже нужного цвета, нужно проверить сам объект:
2. Если исходное фото было темновато, можно чуть повысить экспозицию объекта. Инструментально: создать маску объекта и применить к области объекта фильтр повышения яркости/контраста. Возможно, библиотека PIL или OpenCV (cv2.convertScaleAbs) – добавив небольшое значение.
3. Проверить **баланс белого** на объекте: например, если объект белого цвета, а на фото выглядит желтоватым, стоило бы убрать желтизну. Это можно сделать, вычислив средний цвет объекта (по маске) и сместив его по color temperature. Но этот шаг рискован: вдруг желтоватый оттенок и есть реальный цвет (например, кремовый товар). Поэтому скорее **не трогаем цвета товара** без крайней необходимости.
4. Можно реализовать опцию "*neutralize background influence*": если в исходном фото на объекте были цветные рефлексы от окружения, мы их все равно почти убрали, убрав фон. Осталось, возможно, скорректировать **общий тон**. Тут может помочь простейший прием: слегка повысить контраст (чтобы белое стало белее, черное чернее) – часто фото от любителей бывают немного плоскими.
5. Убедиться, что фон остался заданного HEX. После наложения тени фоновые пиксели где нет тени = точно #F5F4F2. Где тень – они темнее. Это ок.
6. Этот же модуль может заниматься конвертацией цветового профиля: привести к sRGB если не было, убрать возможный прозрачный канал (для JPEG) – то есть финальное приготовление для веб.

7. *Примечание:* Claid.ai заявляет, что они *«автоматически регулируют качество и освещение продукта для лучшего результата»* [37] – видимо, это аналог наших шагов тень+коррекция. Нам, конечно, полностью автоматические сложные фильтры (типа AI enhancement) можно подключить, но осторожно. К примеру, есть фильтры от Adobe (Photoshop auto tone) – можно протестировать их аналоги. Пока заложим простую коррекцию.

8. **Модуль upscale (при необходимости):** функция `upscale_if_needed(image) -> image_upscaled`. Если после всех операций размер изображения меньше 1200×1600, применяем модель супер-разрешения.

9. Используем Real-ESRGAN через локальный запуск или API. Перед подачей убедиться, что изображение в формате без альфа (т.е. JPEG или RGB).
10. Получаем увеличенное изображение, затем можно даунскейл до ровно 1200x1600 или 1600x1600 если вышло больше, чтобы соответствовать требованию (лучше унифицировать, например, всегда делать 1600 по большей стороне).
11. Проверяем, что качество хорошее (если появляются артефакты – можно попробовать другой алгоритм или ограничить кратность увеличения).

12. В большинстве случаев upscale потребуется, если оригинал был очень маленьким. Возможно, имеет смысл на шаге 1 (классификации) отметить изображения с низким разрешением как **особо сложные**, потому что их улучшение – отдельная задача (сильно мыльные фото даже после апскейла будут средними; иногда лучше попросить мерчантов прислать заново, но если нет возможности, делаем максимум возможного).

13. **Финальный сбор результатов:** после всех шагов у нас есть финальное изображение (напр. светлый фон). Мы клонируем его и меняем фон на второй вариант:

14. Можно было заранее сделать оба параллельно: например, на этапе compose_on_canvas сразу делать два холста разного цвета и накладывать объект. Но чтобы не дублировать все преобразования, проще взять уже готовое изображение на светлом фоне + отдельную копию маски тени.
15. Создать копию фонового слоя с цветом #D1CECB, положить ту же тень и объект. Объект у нас уже вырезан и цвет скорректирован, тень, возможно, нужно отрисовать заново (потому что на другом фоне её интенсивность может быть слегка подрегулирована, но можно и ту же).
16. В итоге получаем два файла. Если нужно, переименовываем/номеруем их.
17. **Сохранение**: записываем файлы JPEG с высоким качеством (например, quality=90). Если требуются миниатюры или еще какие-то производные, это уже за рамками – у Яндекс Маркета, скорее всего, есть свой генератор превью для каталога.

18. Логируем успех обработки, переходим к следующему.

19. **Автоматизация и масштабирование:** обернуть всё в скрипт/сервис, который берет на вход список изображений (или папку, или ссылки) и обрабатывает по конвейеру. На уровне архитектуры, возможно, реализуем очереди заданий, параллельную обработку (несколько потоков/воркеров, учитывая, что ИИ-модели могут задействовать GPU – надо оптимально загружать).

20. Можно написать простейший REST API: отправляешь запрос с изображением, получаешь результат. Но миллионы изображений вручную через API гонять трудно – вероятно, интеграция будет с базой данных: сервис вынимает запись о товаре, обрабатывает, сохраняет обратно.
21. На старте, как оговорено, место выдачи не важно – хоть локально. Значит, можно сделать скрипт, пробегающий по каталогу файлов, и сохраняющий рядом готовые.

22. Обязательно предусмотреть **логи ошибок**: если что-то не удалось (модель упала, файл поврежден, и т.п.), записать это, чтобы потом не потерять эти товары.

23. **Модуль обучения/настройки нейросетей (параллельно):** Для части задач, возможно, потребуется обучение своих моделей:

24. Например, **модель для fashion (виртуальные модели)** – это отдельный R&D. Можно в фоне изучать открытые решения, собрать датасет (фото одежды + фото людей) для fine-tune.
25. **Дополнить модель сегментации**: если стандартный U2Net ошибается на специфических данных, можно разметить сотню-другую фото и дообучить модель под наш формат (например, иногда товар на белом фоне, U2Net может "съесть" белые края товара).
26. **GAN для фона?** Нам вроде не нужно генерировать фон, но если бы захотели уникальные фоновые сцены (например, для маркетинговых баннеров) – можно подключить генеративные модели по описанию, как делает Claid (у них есть AI Backgrounds по текстовому описанию [38]). Но повторюсь, сейчас нам важнее единообразие, чем креативность.

Использование Claude и GPT для разработки

Весь проект можно реализовывать итеративно, получая помощь от AI-ассистентов (Claude, GPT-4) на каждом шаге. План такой: **разбивать работу на небольшие задачи и формулировать промпты** для ИИ, чтобы ускорить кодирование и генерацию идей. Ниже – примеры, как можно задействовать Claude/GPT:

- **Проектирование модулей:** сперва можно попросить AI помочь составить шаблон кода для общей структуры. *Пример промпта: «Claude, помоги спроектировать Python-класс ImageProcessor с методами remove_background, add_shadow, upscale и т.д. Опиши, какие входы/выходы у каждого метода.»* Модель сможет предложить каркас, который затем можно скорректировать под наши нужды.

- **Интеграция модели для удаления фона:** *«Как с помощью библиотеки rembg удалить фон с изображения в формате PIL Image? Приведи пример кода.»* – на такой запрос GPT/Claude выдадут конкретный код, например, как использовать `rembg.remove()` и затем сохранить результат. Это сэкономит время на чтение документации. Далее, *«Как доработать маску, чтобы убрать мелкие шумовые области? Можешь показать с использованием OpenCV?»* – AI подскажет, например, использовать `cv2.findContours` и фильтровать по площади.

- **Добавление тени:** можно описать задачу: *«Есть маска объекта (numpy array), хочу создать тень: сместить маску, размыть и нарисовать на фон. Напиши код с использованием OpenCV/PIL.»* Ассистент, вероятно, предложит конкретные шаги (например, `cv2.GaussianBlur` и alpha-композитинг). Мы получим готовый фрагмент кода для вставки.

- **Цветокоррекция:** *«Как слегка повысить яркость и контраст изображения PIL? А как откорректировать баланс белого?»* – AI может посоветовать использовать `ImageEnhance` или numpy манипуляции. Или даже предложит готовый алгоритм Gray World для WB. Мы можем экспериментировать: *«Изобрази код, который вычисляет средний оттенок самого светлого участка объекта и вычитает его из изображения (тем самым нейтрализуя фон)»* – т.е. использовать AI как консультанта по алгоритмам.

- **Апскейл через API или локально:** *«Как вызвать Real-ESRGAN модель в Python? Есть ли готовая библиотека?»* GPT может указать, например, на использование `realesrgan`

через pip или `cv2.dnn_superres`. Также можно попросить: *«Напиши функцию, которая отправляет изображение на API fal.ai и получает увеличенное до 1600px»* – если у fal.ai есть публичный пример, AI может сгенерировать шаблон HTTP-запроса.

- **Отладка pipeline:** здесь удобно воспользоваться возможностями Cursor (если он позволяет запуск кода). Можно попросить: *«Вот код функций A, B, C, и основной цикл обработки – помоги найти ошибку, почему изображения получаются тёмными.»* AI может проанализировать код. Либо: *«Предложи, как оптимизировать выполнение, если нужно обрабатывать 1 миллион изображений – как распараллелить?»* – ассистент подскажет, например, использовать multiprocessing или batch-обработку GPU.

- **Разработка гайдов для мерчантов:** GPT-4 отлично подходит для генерации читабельного текста на основе наших требований. Мы можем передать ему список требований (например, «фон только однородный, предмет полностью в кадре, никаких водяных знаков, формат JPEG, размеры такие-то, примеры хороших/плохих фото») и попросить: *«Составь инструкции для продавцов категории Beauty по съёмке главной фотографии товара.»* Он сгенерирует структурированный текст, который мы потом подправим. Аналогично для других категорий: *«Напиши гайд: Как фотографировать одежду для маркетплейса, чтобы потом нейросеть смогла одеть модель?»* – здесь нужен совет мерчантам, чтобы, например, фотографировали вещь на нейтральном фоне, в определённых ракурсах. GPT может на основе общего опыта сформулировать рекомендации.

- **Примеры промптов для поэтапной реализации:**

- *«Начнём с простого: напиши функцию load_image_paths(directory), которая соберёт все пути к изображениям (jpg/png) в заданной папке и вернёт список. Затем напиши цикл, который по каждому пути загружает картинку через PIL и вызывает пока что пустую функцию process_image().»* – получим каркас перебора файлов.
- *«Теперь реализуем process_image(img): сначала получаем mask = remove_bg(img). Я буду использовать rembg. Дай код с использованием rembg.remove().»*
- *«У меня есть mask (PIL Image или numpy), и вырезанный объект obj_img (PIL с прозрачностью). Как поместить obj_img на новый белый фон 1600x1600 с центровкой? Дай код.»*
- *«Как вычислить bounding box непрозрачных пикселей на obj_img? Можно ли через numpy? Покажи.»* – для масштабирования.
- *«Напиши функцию add_shadow(base_img, mask) на PIL, которая создаст тень. Используй GaussianBlur из PIL.ImageFilter.»* – GPT вернёт код.
- *«Как поменять фон цвет (#D1CECB) на уже получившемся изображении? (т.е. заменить все пиксели фона, не затронув объект и тень)»* – может предложить пройтись по маске: где mask=0, красить цветом.
- *«Мне нужно увеличить изображение до 1600x1600, если оно меньше. Как это сделать с помощью OpenCV интерполяции LANCZOS? Код.»* – GPT даст пример (cv2.resize).
- *«Приведи пример, как использовать RealESRGAN через командную строку или Python API на изображении.»* – если знает, может рассказать про realesrgan-ncnn-vulkan tool или Python wrapper.

- *«Напиши docstring к функции process_image, описывающий весь процесс.»* – получим хорошо оформленный комментарий, полезно для команды.

- **Совместная работа с AI:** Нужно помнить, что модель может ошибаться, но для ускорения рутины она очень полезна. Мы можем итеративно: написать часть сами, затем *«Claude,*

посмотри на код, найди потенциальные проблемы.» или *«Предложи улучшения.»*. Он может подсказать, например, «здесь утечка памяти – лучше очистить объект PIL» или «эта операция можно векторизовать».

- **Контроль версии промптов:** Хорошей практикой будет сохранять удачные промпты и ответы от AI, чтобы формировать своего рода **базу знаний**. Например, однажды составив удачный запрос для тени, его текст можно включить в документацию для команды, чтобы другие разработчики тоже имели подсказку. Можно даже настроить внутри Cursor сценарии, когда по ключевому слову вызывается определённый prompt (если Cursor IDE такое позволяет).

В итоге, используя Claude и GPT как ассистентов, мы значительно ускорим разработку каждого компонента. Они возьмут на себя значительную часть написания кода и черновиков документов (гайдов, README, комментариев), а нам останется проверять и дорабатывать под реальные данные. Таким образом, план реализации – **маленькими шагами**, постоянно консультируясь с ИИ:

1. Настроить окружение (установить нужные библиотеки) – *с помощью AI, если нужно*.
2. Реализовать по одному модулю, тестировать на образцах – *AI поможет с кодом и отладкой*.
3. Собрать конвейер, протестировать на нескольких товарах из разных категорий – выявить проблемные случаи.
4. Улучшить алгоритмы или правила на проблемных случаях.
5. Подготовить описанные **гайды для мерчантов** на основе технических требований – *сгенерировать с AI и согласовать с командой*.
6. Итеративно расширять функционал (например, подключить модуль Fashion-try-on, если будет готов, или автоматическую классификацию входных фото по категориям с помощью нейросети, etc.).

Не стесняйтесь привлекать AI на каждом из этих шагов – как на этапе планирования, так и непосредственно в кодинге. Это соответствует методике быстрого прототипирования: **AI-в-петле разработки**. Каждый мелкий успех (например, корректно вырезанный и вписанный на фон объект) закрепляем, затем движемся к следующему.

Заключение

Мы рассмотрели структурированный план создания сервиса и проанализировали аналоги на рынке. Проект включает в себя множество компонентов – от CV-алгоритмов до интеграции генеративных моделей – но разобрав их по отдельности, видим, что каждую задачу можно решить доступными инструментами. Конечный результат – автоматизированный **конвейер обработки изображений** – позволит обновить миллионы карточек товаров, повысив их качество и единообразие. Это, в свою очередь, улучшит восприятие товара покупателями и может положительно сказаться на конверсии (как отмечали в remove.bg, гармонизация фото повышает доверие и удобство выбора [8]).

Дальнейшие шаги: уточнить требования каждой категории (получить те самые документы, о которых вы упомянули) и на их основе скорректировать детали реализации. Готовность начать поэтапную реализацию есть – можем с помощью Claude приступить с первого шага. Если есть дополнительные вводные данные (например, примеры изображений, конкретные требования подкатегорий), то самое время ими поделиться – это поможет точнее настроить алгоритмы под ваш сценарий.

:Пожалуйста, сообщите, если нужна дополнительная информация или уточнения по какому-либо из описанных этапов.

---

1 2 3 4 5 37 AI Product Photography Suite - Claid.ai
https://claid.ai/

6 7 8 9 10 11 12 E-Commerce Photography & Product Photo Optimization – remove.bg
https://www.remove.bg/g/ecommerce

13 Deep-Image.ai - AI Image Enhancer to Generate & Upscale
https://deep-image.ai/

14 Free AI Image Upscaler | Enhance Photo Quality to 4K - Pixelcut
https://www.pixelcut.ai/image-upscaler

15 Mokker AI - Instant AI Background Replacement
https://mokker.ai/

16 17 18 19 20 21 33 34 35 36 38 I Tested 20+ AI Tools for Product Photography. Here Are the Top 7 for 2025 | by Anangsha Alammyan | What is the best AI | Medium
https://medium.com/what-is-the-best-ai/best-ai-tools-for-product-photography-2025-212a3dcad0a8

22 23 24 25 26 32 Pebblely AI Product Photography | Create beautiful product photos in seconds with AI
https://pebblely.com/

27 28 AI Product Photo Generator & Editor | Create E-Commerce Images | Flair.ai
https://flair.ai/

29 30 Botika - AI generated models for fashion
https://botika.io/

31 Clipping Magic: Remove Background From Image
https://clippingmagic.com/