вобщем хочу сделать сайт канву где ты создаешь типа как запрос он в таком боксике по середине холста в запросе ты пишешь что-нибудь например замок, и потом кликаешь на специальную кнопочку и тебе предлагается несколько вариантов , из боксика выходят как отростки с другими блёстками на которых написано замок сай фай, замок прерий, ходячий замок хоула, ,и например замок под водой, и средневековый замок, и можно ещё и ещё раз тыкать и количество вариантов замка будет расти, и например у меня есть огромный текст текст к ролику 10 минутному, и мне нужно разбить автоматически этот текст на сцены, то есть в боксик я помещаю текст , и мне из боксика появляются ответвления. первая сцена, вторая сцена третья, то есть ллм, должна определить где например длинные сцены с ведущим будут показаны а где будут вставки которые должны проиллюстрировать его слова которые он говорит по тексту, и ллм должна вывести эти сцены вокруг центрального блока, вобщем хочу сделать вот такой проект помоги доработать идею и нормально ее описать чтоб подробно может какие то моменты не совсем ясны? в целом суть в том чтобы можно было бы получить например текст сценария какованибудь короткого видео предположим научно популярного, по итогу мне нужно будет разделить часть с крупными планами ведущего общими планами ведущего, разметить это время в на таймлайне некоем, мне нужно чтобы ллм прочитала внимательно сценарий и поняла бы что можно проиллюстрировать из того что говорит ведущий наглядно, и ещо я должна иметь возможность стиль задать, наапример модель увидела в тексте возможность проиллюстрировать какую то метафору например о сложном выборе, я должна получить место этой метафоры на таймлайне и вариант подсказки того как это можно визуализировать имея в виду нещо и общий стиль проекта, вот пока так могу обьяснить помоти пожалуйста додумать чего я там хочу скинула картиночку проверь

**ChatGPT сказал:**

Размышлял о проекте автоматизированного разбора сценариев для 6 секунд

Ниже пример, как можно более структурированно описать идею и выделить основные моменты, чтобы проект был понятным и целостным. Я специально включаю разные аспекты: интерфейс, функциональность, логику работы с текстом и возможные расширения.

**1. Общая концепция**

Проект представляет собой **интерактивную «канву» или визуальный холст** (похоже на mind map или карту идей), где в центре у пользователя есть «бокс» – центральный блок. Пользователь вводит в этот блок запрос (например: «замок», «текст сценария» и т. д.) и нажимает кнопку для генерации идей/подпунктов.

* **Ключевая особенность**: при каждом нажатии специальной кнопки к центральному блоку «распускаются» дополнительные ветви (ответвления). Это могут быть:
  + Разные варианты одной и той же идеи (напр. «замок сай-фай», «замок под водой», «замок в прерии», «Ходячий замок Хоула» и т. д.).
  + Автоматически сгенерированные подсекции текста (сцены), если в блок загружен большой сценарий.
  + Иные варианты контента, основанные на том, что анализирует модель (LLM).

Таким образом, система строит иерархическую или радиальную диаграмму, где каждая ветвь – это «подзапрос» или «подидея».

**2. Разметка сценария (ключевой кейс)**

Один из главных сценариев использования – **работа с длинным текстом** (например, научно-популярный сценарий на 10 минут). Пользователь вставляет весь текст в центральный блок на канве, и модель (LLM) должна:

1. **Разбить сценарий на сцены** (напр. «Сцена 1 – вступление ведущего», «Сцена 2 – демонстрация эксперимента», «Сцена 3 – закрывающая часть» и т. п.).
2. **Определить**, где появляются какие элементы:
   * Длинные/крупные планы ведущего.
   * Общие планы ведущего.
   * Места, где можно вставить иллюстрации, графику, метафоры.
3. **Сгенерировать подсказки** по визуализации. Например:
   * «Здесь ведущий говорит о сложном выборе – можно вставить визуальный образ весов»
   * «Рассказывается о научном эксперименте – стоит показать крупным планом колбы, реагенты»
4. **Распределить сцены по таймлайну** (как вариант, LLM может предложить приблизительное время в секундах или просто логический порядок «Сцена 1 с 00:00 по 01:00», «Сцена 2 с 01:00 по 02:15» и т. д.).

**2.1. Взаимодействие с LLM**

* Пользователь загружает текст → LLM «читает» и делит на сцены.
* Визуально на канве в центре по-прежнему один бокс (весь текст), а вокруг появляется несколько ветвей: Сцена 1, Сцена 2, Сцена 3 и т. д.
* Каждый такой блок-сцена можно «раскрыть», чтобы увидеть предложенные детали визуализации/метафоры.

**3. Гибкая настройка стиля**

Для иллюстраций и визуальных вставок важно иметь возможность задавать **стиль**, чтобы LLM учитывала его при генерации подсказок:

1. **Общий стиль проекта**: например, «строгий научный стиль», «фэнтези-анимационный стиль», «винтажная стилистика», «стиль аниме» и т. п.
2. **Частные уточнения**: если в тексте появляется метафора о «сложном выборе», пользователь может настроить, какой именно визуал ему ближе: «более реалистичный», «абстрактный», «юмористический» и проч.

Тогда LLM, проанализировав это, будет предлагать конкретные визуальные решения, соответствующие заданной «атмосфере» ролика и жанру.

**4. Механика работы с ветвями**

1. **Начальный ввод**: пользователь вводит в центральный блок либо короткое слово («замок»), либо длинный текст (сценарий).
2. **Генерация ответвлений**: нажатие кнопки запускает LLM, которая генерирует несколько уточняющих/дополняющих вариантов (замки разных типов, разные сцены в тексте и т. д.).
3. **Многократное расширение**: при желании пользователь может нажимать кнопку и «добиваться» всё новых ветвей, собирая более детализированную структуру.
   * Например, если первая генерация для «замок» выдала 5 вариантов, можно нажать кнопку ещё раз, и при желании получить вторую волну вариантов («замок на Луне», «замок-музей будущего» и т. д.).
4. **Управление блоками**: каждый блок на канве (большой или маленький) можно «схлопывать» или «раскрывать», а также перемещать, чтобы удобно организовывать структуру.

**5. Пользовательские сценарии (кейсы)**

1. **Быстрая генерация идей**: пользователь пишет слово «замок» – получает несколько визуальных концептов. Это может пригодиться художникам, дизайнерам, писателям.
2. **Декомпозиция сценария**: пользователь загружает сценарий → получает разбиение на сцены и идею визуализации. Это нужно авторам видео, режиссерам, контент-мейкерам, которым важно увидеть логику и хронометраж.
3. **Постановка визуальных эффектов**: пользователь хочет понять, **какая часть речи ведущего** должна быть «подкреплена» вставкой графики или анимацией. LLM предлагает места, описывает, как это может выглядеть, и пользователь уже решает, какие вставки делать.
4. **Учет стиля**: при желании пользователь выбирает «научно-популярный» стиль → система генерирует более строгое оформление и реалистичные примеры визуализации. Если «сказочный/фэнтези» – тогда будут магические элементы, яркие цвета и т. д.

**6. Вопросы/идеи для дальнейшего развития**

1. **Интеграция таймкодов**:
   * Хотите ли вы, чтобы LLM сама ставила примерные таймкоды? Или нужен ручной режим, где пользователь сам укажет продолжительность каждой сцены, а модель только помогает «подписать» эти интервалы?
2. **Дополнительные блоки с метаданными**:
   * Можно ли будет к каждой ветви прикреплять заметки, изображения, аудиофайлы, ссылки? Это может быть удобно в процессе реальной сценарной работы.
3. **Генерация связей между сценами**:
   * Может ли LLM искать «переклички» между сценами (где одна сцена ссылается на другую) и предлагать переходы? Например, «В конце Сцены 2 уже подводится мысль к Сцене 3…».
4. **Настройка шаблонов**:
   * Например, есть типовой шаблон для «коротких научпоп-роликов», другой – для «художественных фильмов». Модель использует этот шаблон для разметки сцены (реквизиты, локации, костюмы).
5. **Совместная работа пользователей**:
   * Можно ли будет несколько человек пригласить на этот холст и вместе двигать блоки, править и генерировать?
6. **Экспорт**:
   * Интересно ли сделать экспорт в формат, понятный видеоредакторам (например, XML для DaVinci Resolve / Adobe Premiere), чтобы потом автоматически расставить сцены?

**7. Пример итогового описания (укороченный вариант)**

**Описание:** «Canvas Story» – это интерактивная система на базе LLM, позволяющая создавать структуры и сценарии прямо на визуальном холсте. В центре холста пользователь вводит запрос или сценарный текст. Система с помощью LLM анализирует ввод и генерирует вокруг центрального блока ветвящиеся подпункты или сцены.

**Применение:**

* Создание mind map для идей (напр. разные интерпретации замков).
* Разбиение длинного текста на логичные сцены и раскадровку видеоролика.
* Автогенерация подсказок, как иллюстрировать отдельные моменты в сценарии, в том числе выбор стиля (фэнтези, sci-fi, строго-научный и т. д.).
* Возможность настраивать временные метки для каждой сцены, чтобы облегчить пост-продакшн видео.

**Интерфейс:**

* Центральный блок с исходным текстом.
* «Ветви» ответвлений, генерируемые моделью, которые можно «раскрывать» и «сворачивать».
* Возможность вручную редактировать, менять структуру, добавлять примечания и файлы.

**Планы развития:**

* Интеграция таймкодов.
* Автоматические переходы и подсказки для монтажа.
* Настройка шаблонов по жанрам и стилистике.

Таким образом, проект превращается в **мощный инструмент** для раскадровки, сценарной разметки или генерации идей.

**Итого**

1. **Уточните, какие именно функции самые важные** (например, нужна ли обязательно авторасстановка таймкодов? Или главное – разбиение на сцены?).
2. **Подумайте, как хранить и показывать результаты**: будет ли это как mind map-вид, или нужна таймлиния в нижней части экрана?
3. **Рассмотрите формат входящих данных**: тексты, картинки, видео, аудио? Пока вы упомянули текст, но со временем проект может дополняться мультимедийными средствами.
4. **Заложите логику редактирования**: чтобы пользователь мог корректировать автоматическую разбивку, объединять сцены, переименовывать ветви и т. д.

Надеюсь, это помогает обрисовать **общую идею** более системно и полно. Далее уже можно обсуждать, как технически реализовать каждый модуль, как именно обучать/использовать LLM, как строить визуальный интерфейс и т. п.

**Вы сказали:**

по поводу таймкодов прекрасный вопрос у меня ответ такой, пусть если не предаставлен голос озвучивающий текст то ллм сама ставит таймкоды на глаз, да в низу экрана должна быть тайммап с понятными подсветками и если навести на какую то часть то нода на канве с этим кусочком пусть подсвечивается да точно нужен аудио и картинки и тексты, а видео не надо. давай отдельно обсудим часть с редактированием что ты имеешь в виду? вот типа ллм прочитала текст большой вот она делает первое предложение с разбиением она например увидела в начале как крупный план ученого очки сверкнули и он поднес к лицу бутылку с прозрачной жидкостью сделал паузу и заговорил загадочным голосом а некоем новом лекарстве, мне например не понравилось что ллм выбрала такой уж крупный план в самом начале я например хочу посторить логику так что на протяжении всего ролика мы как бы постепенно погружаемся в материал и каждый раз когда нам будут показывать профессора то камера будет наезжать все ближе и ближе и вот в конце я уже точно хочу крупны план, как мне сообщить о своих хотелках ллм? должен быть какой то предварительный промпт, или так я могу несколько раз сгенерить что то посмотреть что она видит сформировать свой вариант соединить с теми идеями которые она накидала и этот финальный вариан как то использовать? помоги понять как это реализовать это кажется выглядит более приемлемо чем либо соглашатся на все или на все не соглашатся и править руками? как можно будет вылавливать и редактировать идеи и пробовать подмешивать их к новым генерацияям?

**ChatGPT сказал:**

Рассуждала для секунда

Ниже – идеи, как можно организовать **редактирование** предложений LLM и взаимообмен между пользователем и моделью, чтобы гибко дорабатывать результат.

**1. Базовая логика правок**

1. **Первичная генерация**:
   * Пользователь загружает сценарий → ЛЛМ делает **первый автопроход**, разделяя текст на сцены/планы/визуализации.
   * Например, предлагает «Сцена 1: крупный план учёного», «Сцена 2: общий план лаборатории» и т. п.
2. **Отдельная панель/список сцен**:
   * После генерации мы видим перечень сцен (а также таймкоды, если голосовой дорожки нет).
   * Допустим, ЛЛМ предлагает «Сцена 1: Камера сразу наезжает на лицо учёного, крупный план».
3. **Ручная правка**:
   * Пользователь кликает на какую-то сцену, видит описание от ЛЛМ: «Крупный план учёного. Он держит в руках бутылку…».
   * Если не нравится крупный план или хочешь другую композицию (например, начать с дальнего плана), **просто редактируешь текст** (прямо в интерфейсе) и пишешь: «Начинаем со среднего/общего плана учёного, камера чуть сверху, медленный зум».
4. **Отклик ЛЛМ**:
   * Система может предложить «перегенерировать» текущее описание сцены, учитывая вручную внесённые коррективы.
   * То есть ты меняешь формулировку (или словесно пишешь «Хочу постепенно приближаться к лицу учёного, а крупный план - только в финале»), нажимаешь кнопку «Обновить» или «Учесть правки», и ЛЛМ перестраивает описание следующих сцен с учётом нового подхода.
5. **Интерактивное дообучение**:
   * В результате ЛЛМ может подхватить изменение (что крупный план только к концу ролика) и **автоматически скорректировать** дальнейшие сцены.
   * Например, сцена 2, 3, 4 пересматриваются – если модель видит, что в сцене 3 раньше она хотела «ещё раз крупный план», она поменяет на «средний план», чтобы сохранить общую логику постепенного приближения.

**2. Механизмы взаимодействия (редактирование идей)**

**2.1. Ручная правка с “фиксирующими заметками”**

В каждом блоке (сцене) есть возможность оставлять заметку/комментарий:

* «Сделать план более общим»
* «Не показывать лицо учёного крупно – сохранять интригу»
* «Переставить этот момент ближе к концу»

При **перегенерации** ЛЛМ учитывает эти комментарии как «жёсткие правила» (hard constraints).

**2.2. Итеративные промпты**

Пользователь может писать «второй промпт» или «коррекцию» глобально, например:

«Переделай сценарий так, чтобы **сначала** был общий план, мы видим лабораторию, **постепенно** камера приближается к учёному. В первой минуте не показываем крупный план его лица. Крупный план – только в конце, когда учёный делает заключение».

Модель анализирует это условие и **перестраивает** все сцены так, чтобы удовлетворить новым правилам.

**2.3. Локальное перегенерирование**

* Если только одна сцена не устраивает, можно нажать «Перегенерировать сцену» с уточнённым промптом.
* Если всё устраивает, кроме сцены 1, в которой нужно заменить «крупный план» на «общий план», можно задать поправку лишь на эту сцену.
* В идеале модель помнит контекст и дальше понимает, что последующие сцены надо чуть скорректировать (если в сцене 2 она ссылалась на «переход с крупного плана» – теперь этот переход будет с общего).

**2.4. Режим «соединить» или «мердж»**

* Иногда ЛЛМ выдаёт **альтернативные** версии (скажем, первая генерация и вторая генерация).
* Пользователю может понравиться часть идей из первой и часть – из второй. Можно сделать «мердж» сцен: выбрать из двух вариантов сцены 1 тот, что больше по душе, а из второй генерации – сцену 2.
* Технически это может выглядеть как список вариантов ( «Вариант 1» / «Вариант 2» ) у каждой сцены, между которыми пользователь переключается, а в итоге «фиксирует» нужный вариант.

**3. Как сообщать о «хотелках» ЛЛМ**

1. **Глобальные настройки проекта**:
   * Перед генерацией пользователь задаёт/прописывает ключевые принципы: «Мы идём от общего к частному, крупный план – только в конце, стиль – научно-популярный, без излишней мрачности, с долей юмора» и т. д.
   * ЛЛМ уже изначально формирует структуру, руководствуясь этими настройками.
2. **Итеративная донастройка**:
   * После генерации смотришь, что вышло, и пишешь краткие комментарии: «Не нравятся крупные планы в начале», «Нужно больше динамических переходов» и т. п.
   * ЛЛМ пересобирает структуру.
3. **Комбинация автоматического и ручного**:
   * Иногда удобнее просто «руками» поправить текст сцены (сделать ‘правку вживую’).
   * Если нужно, чтобы эти правки **системно** отразились во всех следующих сценах, жмёшь «Пересобрать далее» (или «Применить правило глобально»), и ЛЛМ сама подстраивает продолжение.

**4. Организация в интерфейсе**

1. **Тайммап внизу**:
   * Ползунок, где условно есть 0:00 – 10:00. Сцены расположены друг за другом по времени.
   * Наводишься на сцену (пусть там условно «1:00–2:00»), она подсвечивается на канве.
2. **Редактирование прямо на канве**:
   * Щёлкаешь по блоку «Сцена 1», видишь текст описания – можешь поправить.
   * Или в отдельной боковой панели редактирования (можно открывать список сцен с описаниями, вносить правки).
3. **Кнопка «Обновить с учётом правок»**:
   * Когда вносишь изменения, система запрашивает: «Перегенерировать автоматически все сцены ниже?» (чтобы сохранять логику).
   * Можно согласиться или отказаться (если хочешь, чтобы только эта сцена была изменена).
4. **Место для заметок**:
   * Если пользователь **не хочет** сразу менять текст, а просто оставить ЛЛМ «хинт» (совет), это может быть отдельная «шторка» замечаний. Например, при следующей глобальной генерации эти замечания будут учтены.

**5. Пример пошагового сценария правок**

1. **Шаг 1**: Вставляешь текст (10 минут), нажимаешь «Сгенерировать».
2. **Шаг 2**: Модель выдаёт 5 сцен, каждая со своим описанием, примерными таймкодами.
   * Сцена 1 (0:00–1:00): «Крупный план учёного, очки сверкают…»
3. **Шаг 3**: Ты видишь, что хочешь другой ракурс.
   * Два варианта действий:
     1. **Прямо правишь**: «Сцена 1: общий план лаборатории, профессор стоит у стола, медленный наезд…»
     2. **Говоришь ЛЛМ**: «Пожалуйста, не начинай с крупного плана, а поставь общий план. Также сделай упор на постепенное погружение в эксперимент».
4. **Шаг 4**: Нажимаешь «Применить правку». Модель перестраивает Сцену 1 и заодно смотрит, нет ли в Сцене 2 ссылок на крупный план начала. Если там что-то противоречит новой концепции, она это тоже правит.
5. **Шаг 5**: Теперь смотришь, что получилось. Может быть, всё стало гармонично. Если нет, повторяешь цикл уточнений.

**6. Пример конечной реализации (в упрощённом виде)**

1. **Проект «Canvas Story»** – интерфейс с «канвой» (центральный блок, ветвящиеся сцены) и ниже – **таймлиния**.
2. **Сцены** – каждый узел/ветвь соответствует сцене + описание. При выборе сцены на таймлинии подсвечивается соответствующий блок, и наоборот.
3. **Редактирование** – двойной клик по сцене открывает редактор:
   * Можно отредактировать текст вручную.
   * Можно нажать «Попросить ЛЛМ перегенерировать по заданным критериям».
   * Можно оставить комментарии/заметки.
4. **Итеративный процесс** – после каждой правки ЛЛМ может «дотягивать» логику соседних сцен, чтобы быть консистентной.

**Итоговая суть**

* **Не нужно** «соглашаться со всем» или «отказываться от всего». У нас есть **гибкое рабочее пространство**, где ЛЛМ – ассистент. Она даёт авторазметку, но пользователь **в любой момент** может вносить правки (словесно через дополнительный промпт или напрямую в тексте), а потом попросить «обновить» всю структуру.
* Таким образом, получается **процесс итеративного «сотворчества»** – модель помогает, генерирует, перестраивает, а человек принимает финальные решения и может вносить точечные правки.

Так мы учитываем желания пользователя (поэтапное приближение камеры, нужные стилистические детали, расстановка таймкодов «на глаз» и т. п.) и одновременно используем преимущества автоматической генерации.

**Вы сказали:**

хотела добавить что мне бы потом уже после того как разбиение на сцены произведено, описание сцен завершено и утверждено, нужно сделать следующий шаг это промптвизация полученных сцен, промптивизация это когда мы берем сценарное художественное описание сцены и превращаем это описание в промпт для генерации картинок что нужно учитывать при промптивизации, все тексты должны консистентно описывать персонажей внутри одного сценария, тоесть если мы описывали в начале доктора как седого старика в очках то он везде так и должен быть описан так же должны быть консистенты и описания комнаты и всего такого генеративная моддель не может видеть весь сценарий целиком поэтому там нельзя написать что то вроде он увидел свое отражение в зеркале нужно писать что седой ученый в белом халате смотрит на себя в зеркале зеркало находится в научной лаборатории кругом раставленны колбы и минзурки итд. и тп. тоесть промптивизация текста должна занимать довольно много места и быть подробной учитывать контекст сцены даже если конкретно в сцене которую она описывает уже нет подробного описания сцены потому что например ее ллм художественно описла вначале

**ChatGPT сказал:**

Рассуждал о процессе промптивизации сценария для 29 секунд

Ниже описан подход к следующему этапу – **«промптивизации»** (то есть превращению каждого описания сцены в развернутый prompt для генеративной модели), с учётом необходимости **сохранения консистентности** (прическа, возраст, обстановка) и **максимальной детализации** (ведь генеративная модель не «видит» всего сценария целиком).

**1. Основная идея промптивизации**

1. **Отдельный «библиотечный» слой**:
   * Мы храним **ключевые факты** о персонажах (имена, внешний вид, одежда) и **обстановке** (тип комнаты, освещение, стиль предметов) в специальном «досье» или «Character & Setting Bible».
   * При генерации prompt для каждой сцены мы **подхватываем** из этой библиотеки все важные детали, чтобы они не потерялись и были единообразны.
2. **Подробное описание в каждом prompt**:
   * Генеративная модель обычно видит **только** тот текст, что мы ей передаём при конкретном запросе. Она не помнит, что было в предыдущих сценах (в отличии от ЛЛМ, у которой может быть контекст).
   * Поэтому в **каждом prompt** повторяем/дублируем ключевые элементы:
     + «Седой учёный в очках, в белом халате, в научной лаборатории, вокруг колбы и мензурки…»
   * Если в сцене что-то меняется (например, учёный снимает очки или лаборатория полна дыма), это тоже надо явно указать.
3. **Единый стиль проекта**:
   * Если у нас есть общий художественный стиль (например, реалистичный, футуристический, мультяшный и т. п.), мы добавляем это как **глобальную часть prompt** к каждой сцене.
   * Аналогично – детали освещения, атмосферы, палитры: «сумрачная атмосфера», «яркие, сочные цвета» и т. п.

**2. Общая схема процесса**

1. **Собрать итоговый сценарий**
   * Когда сцены (и их описания) выверены и зафиксированы, у нас есть окончательная разбивка.
2. **Создать «персонажную/сеттинг-библиотеку»**
   * Короткая таблица/список:
     + **Доктор / Профессор N**: седой, 60+ лет, носит очки, белый халат, лёгкая сутулость…
     + **Лаборатория**: большая, белые стены, оборудование, пробирки, стерильные столы.
     + **Дополнительные персонажи** или **ключевые объекты** (если есть).
   * Можно хранить уточнения вроде «Эмоциональные особенности», «Манера держаться», «Тембр голоса» (даже если мы не генерируем звук, это влияет на визуальную подачу).
   * Можно добавить глобальные стилистические пометки: «Жанр – научно-фантастический», «Общий тон – слегка мрачный, загадочный», «Минималистический дизайн интерьера» и т. д.
3. **Итерировать по каждой сцене**
   * Для каждой сцены берём её **литературное** описание (из утвердённого сценария).
   * **«Обогащаем»** это описание нужными деталями (которые мы, возможно, упоминали в первой сцене, но здесь не повторили).
   * Учитываем изменения, если они прописаны: например, «В сцене 5 учёный в противогазе» → уже не пишем «очки сверкают», а пишем «маска», «тёмные линзы», но всё остальное (седые волосы, возраст, телосложение) может оставаться фоном.
4. **Формируем финальный prompt**
   * Структурно может выглядеть так:

css

КопироватьРедактировать

[Глобальный стиль/жанр]

[Описание сцены в контексте сценария]

[Информация о персонажах из библиотеки (чтобы сохранить консистентность)]

[Особые изменения/дополнения для этой сцены]

[Доп. настройки: ракурс, освещение, композиция и т. д.]

* + Например:

bash

КопироватьРедактировать

"Realistic sci-fi style, cinematic lighting.

Scene: Professor Gray, a 60-year-old scientist with silver hair, wearing a white lab coat, stands in a large laboratory filled with shelves of colorful flasks and high-tech equipment.

He is carefully looking at a small vial in his hand, the expression on his face is focused and slightly tense.

The atmosphere is futuristic and slightly ominous, with dim overhead lights reflecting off the steel surfaces.

The camera angle is a medium shot, capturing the professor from the waist up, showing the lab background around him."

1. **Отправляем prompt в генеративную модель**
   * Генеративная модель выдаёт изображение.
   * При желании сохраняем несколько вариантов и выбираем лучший.

**3. Детали для сохранения консистентности**

1. **Единая именованная сущность**
   * Если во всём сценарии фигурирует «Доктор Смит», используем **одно** имя и **одни** характеристики (например, «Dr. Smit – a tall, gray-haired scientist…») во всех сценах.
   * Избегать разночтений: «седой учёный» → «седой учёный» (а не «брюнет» или «в возрасте 40 лет» в другой части).
2. **Список ключевых визуальных особенностей**
   * Внешность: цвет волос, одежда, аксессуары (очки, кольцо на руке, кулон и т. д.).
   * Обстановка: лаборатория, кабинет, декорации.
   * Тон/эмоциональное состояние: «волнуется», «решительно смотрит», «улыбается».
3. **Если герой что-то меняет по ходу сюжета**
   * Например, снимает халат, надевает защитный костюм → это явно прописывается именно в тех сценах, где это происходит, и всех последующих (если костюм остаётся).
4. **Отслеживание инвентаря**
   * Если персонаж что-то держит в руках (бутылку/пробирку) или меняет предмет (был синий прибор – стал красный), надо уточнять это в prompt, чтобы не получилась нестыковка.

**4. Как быть с художественным описанием, «размазанным» по разным сценам**

* Часто в начале сценария есть **развёрнутое описание** комнаты, а в следующей сцене это описание сокращено, потому что «и так понятно, что она в лаборатории».
* Для «промптивизации» нужно **снова** включить ключевые детали, даже если они не описаны в самой сцене, но являются важными.
  + Иначе модель может сгенерировать «голую» сцену или вообще переместить персонажа в пустое пространство.

Пример:

* Сцена 1 (литературно): «Учёный заходит в просторную лабораторию, вдоль стен приборы…»
* Сцена 2 (литературно): «Учёный оживлённо беседует с ассистентом. Она протягивает ему пробирку».

**Prompt для Сцены 2** всё равно должен упоминать: «They stand in a spacious modern laboratory, with scientific instruments lining the walls…», чтобы генеративная модель правильно визуализировала контекст.

**5. Техническая реализация промптивизации (пример)**

1. **Хранение данных**:
   * Файл/таблица «characters.json» со схемой:

json

КопироватьРедактировать

{

"Professor\_Gray": {

"age": 60,

"hairColor": "silver",

"hairStyle": "short",

"appearanceDetails": "wears thick glasses, white lab coat",

"personality": "focused, slightly obsessive"

},

"Assistant\_Anna": {

...

}

}

* + Файл «settings.json» с описанием локаций:

json

КопироватьРедактировать

{

"main\_lab": {

"size": "large",

"style": "modern, bright, futuristic",

"equipment": "shelves of flasks, advanced computers..."

}

}

* + Глобальные настройки стиля: «Realistic sci-fi», «Slightly dark mood», «Dramatic lighting» и т. д.

1. **Алгоритм генерации**:
   * По каждой сцене берём её текст + идентификаторы персонажей и локаций (например, «Professor\_Gray», «main\_lab»).
   * Складываем вместе:
     1. Описание из «characters.json» для каждого задействованного героя.
     2. Описание локации из «settings.json».
     3. Литературное описание самой сцены.
     4. Глобальные настройки стиля.
   * Формируем итоговую «промпт-строку».
   * (При желании) автоматически можем подмешивать параметры генеративного движка: размер картинки, угол обзора и т. д.
2. **Учёт итераций**:
   * Если картинка не нравится, пользователь может добавить уточнение: «Хочу ракурс повыше, чтобы был виден стол на переднем плане» → мы добавляем это в prompt (или в scene description) и перегенерируем.

**6. Пример**

**Сцена 1 (литературное описание из сценария)**

«Учёный, седоволосый и чуть сутулый, входит в просторную лабораторию. Он осматривает колбы, наполненные цветными реагентами…»

**Промпт** (после обогащения):

vbnet

КопироватьРедактировать

Style: Realistic cinematic sci-fi, slightly ominous.

Location: A large futuristic laboratory filled with shelves of flasks, test tubes, advanced scientific equipment.

Characters: Professor Gray, a 60-year-old scientist with silver hair, wearing a white lab coat and thick glasses, slightly stooped posture, expression of curiosity.

Action: He is entering the lab for the first time, looking around at the array of colorful chemical flasks.

Lighting: Dim overhead lights reflecting on steel surfaces, highlighting the professor and the glassware.

Camera angle: Medium shot, capturing both the professor and the lab environment.

**Сцена 2 (литературное описание)**

«Доктор Смит берёт пробирку с зелёной жидкостью. Над головой пробегают всполохи сиреневых ламп. Учёный кажется взволнованным…»

(Предположим, это тот же учёный, просто в сценарии имя поменялось на «Доктор Смит» — в базе у нас он будет всё равно Professor\_Gray, и это нужно согласовать.)

**Промпт** (учитывая, что это тот же персонаж):

vbnet

КопироватьРедактировать

Style: Realistic cinematic sci-fi, slightly ominous.

Location: The same large futuristic laboratory with advanced scientific equipment, test tubes, flasks.

Characters: Professor Gray (age 60, silver hair, white lab coat, thick glasses, same stooped posture).

Action: He holds a test tube with a glowing green fluid, the overhead lights occasionally flash purple glows.

Emotional tone: He appears anxious, anticipating a crucial discovery.

Camera angle: Close-up shot of the professor's upper body and the vivid green test tube.

(Мы исправили «Доктор Смит» → «Professor Gray» ради консистентности, если это один и тот же персонаж.)

**7. Вывод**

* **Промптивизация** – это не просто «взять описание сцены слово в слово». Нужно **обогащать** сцену контекстом, дублировать внешность персонажей и детали окружения для каждой генерации.
* Важно поддерживать **единый источник** правды о персонажах, локациях и стиле, чтобы в разных сценах мы не уходили в противоречия.
* Технически это реализуется через небольшую «библиотеку» (JSON или любая база), куда заносим ключевые характеристики, и при генерации prompt’а автоматически объединяем их с кратким текстом сцены.
* В результате каждая отдельная генеративная операция получает **максимально полное описание**, а итоговые картинки будут согласованны друг с другом.

Таким образом, после того как **разделение на сцены** и **описание** завершаются, следующий этап – это **«экспорт» или «промптивизация»**: мы превращаем каждую сцену в подробный prompt с учётом глобальных (стиль, атмосфера) и локальных (кадры, эмоции) нюансов, **не забывая** при каждом шаге повторять критические детали для сохранения единого визуального мира.

**Вы сказали:**

так меня все устраивает какие мои действия должны быть дальнейшие? что ещо нужно сделать чтобы дать четкое задание ллм что нужно делать чобы накодить этот проект? помойсу мы все учли я уже немного столкнулась с тем что переделывать сложнее чем делать с нуля поэтому дерзко небоюсь предлагать идеи но сам проект тоже не должен быть громоздким должен быть понятным удобным инструментом если чего надо будет мы новый напишем так то да? что думаешь, все у нас есть текст который говорит ведущий, мы под этот текст сочиняем визуал, редактируем правим пока не надоест, отправляем в промптивизатор, и на выходе у нас должны получится промпты которые мы будем генерить и втсалять в таймлайн в идеале бы ещо сразу придумать движения персонажам что бы после того как с визуализацике промпта закончено мы могли бы сгенерить видео по этой картинке но там наверное проще просто увидеть картинку подтянуть контекст сценарный и написать промпт для движения вот в таком шаблоне prompt = subject (subject description) + subject Movement + scene (scene description) + (camera language + lighting + atmosphere)

**ChatGPT сказал:**

Рассуждала для 5 секунд

Ниже — **пошаговый план**, который поможет превратить нашу концепцию в реальный (пусть и упрощённый) работающий прототип. Я разбил это на несколько блоков: **Архитектура**, **Функциональные модули**, **Интерфейс**, **Последовательность разработки** и **Задание для LLM**. Это не абсолютная догма, а скорее «каркас» — мы можем его упрощать, дополнять или менять, если понадобится.

**1. Архитектура (общий обзор)**

1. **Frontend** (React/Vue/Angular — любой фреймворк, который нравится):
   * Визуальный холст (Canvas / Mindmap) для покадровой (помесячной) работы:
     + Центральный блок: сценарий или короткий запрос.
     + Дочерние блоки (сцены, варианты, прочие ветви).
   * Таймлиния внизу экрана, синхронизированная со сценами.
   * Компонента для редактирования текстовых описаний сцен.
   * Кнопка «Сгенерировать» (обратиться к LLM).
   * Кнопка «Перегенерировать/Обновить» (c учётом правок).
2. **Backend** (Python / Node.js / Go — на выбор):
   * **API**:
     + **/split\_scenario** — принимает весь сценарий, возвращает структуру сцен с предполагаемыми таймкодами.
     + **/refine\_scene** — принимает одну сцену + комментарии/уточнения, возвращает обновлённое описание (и при необходимости корректирует связанные сцены).
     + **/prompts\_generate** — для «промптивизации» (на основе зафиксированного сценария, глобального стиля и базы данных персонажей).
   * **Хранилище** (база или просто JSON/файлы) для:
     + Cценария, разбивки, правок.
     + Библиотеки персонажей и локаций (Character & Setting Bible).
     + Глобальных настроек (стиль, жанр, прочие параметры).
3. **LLM интеграция**:
   * Можем обращаться к любой доступной модели (OpenAI, локальная LLM и т.д.).
   * ЛЛМ выполняет ключевые операции: разбивка сценария, корректировка сцен, генерация подсказок, и частично — формирование финального текста prompt’а (или помогает в этом).

**2. Основные функциональные модули**

**2.1. Модуль «Разбиение сценария на сцены»**

* **Вход**: текст (монолог ведущего).
* **Выход**: список сцен (title, start\_time, end\_time, текст описания, доп. поля).
* **LLM-задача**: «Проанализировать длинный текст, найти логические блоки, определить примерные таймкоды (если нет звуковой дорожки)».

**2.2. Модуль «Редактирование и обновление сцен»**

* Предоставляет интерфейс, где каждая сцена имеет:
  + **Текстовое описание** (камера, эмоции, крупный/общий план и т. д.).
  + **Пользовательские комментарии**/«жёсткие правила» (например, «запретить крупный план в первой сцене»).
  + **Кнопку «Перегенерировать»** — отправляет на бэкенд запрос с новыми условиями → ЛЛМ корректирует и пересобирает сцену (при желании — все сцены).
* При сохранении изменений (или нажатии «Обновить») бэкенд записывает их в базу, и если логика сцены изменилась, пересматриваются зависимости (к примеру, смена крупного плана на общий влияет на переход в следующую сцену).

**2.3. Модуль «Таймлиния»**

* Отображает сцены (плашки) со временем начала и конца.
* При наведении — подсвечивается соответствующая сцена на холсте.
* Позволяет **перетаскивать** сцены (менять порядок), если пользователь хочет поменять логику. При этом триггерит обновление бэкенда (чтобы ЛЛМ подтянула уточнения).

**2.4. Модуль «Промптивизация»**

* После того как сценарий (разбивка, описание) утверждён, нажимаем кнопку «Сформировать промпты».
* **Бэк** берёт:
  1. Глобальные настройки стиля.
  2. Описание каждой сцены.
  3. Библиотеку персонажей/локаций, чтобы убедиться в консистентности.
* Создаёт для каждой сцены **подробный prompt**:
  1. Ключевые визуальные характеристики (если не указаны в самой сцене).
  2. Описание персонажей (возраст, одежда, прическа).
  3. Описание окружения (лаборатория, освещение).
  4. Стилистические указания (cinematic, realistic, dramatic lighting и т. д.).
* **Выход**: массив (или список) готовых prompt’ов — каждый prompt можно скопировать и отправить в любую генеративную модель (Stability AI / Midjourney / и т. п.) для рендера картинки.

**2.5. Дополнительный модуль «Движения персонажей» (опционально)**

* Если захотите расширить на видео-анимацию:
  + Выделяем в каждой сцене движение: «Персонаж смотрит направо, делает шаг вперёд», «Камера плавно подъезжает».
  + Превращаем это тоже в промпт или сценарий для motion-генерации (например, text2video).
* Но это уже **расширенный уровень** — на этапе MVP можно оставить в виде описательных подсказок.

**3. Интерфейс и пользовательский поток**

1. **Главный экран**:
   * Поле «Вставьте весь сценарий ведущего здесь».
   * Кнопка «Сгенерировать сцены» → LLM возвращает базовую структуру.
2. **Холст**:
   * Отображение сцены №1, №2, №3 … в виде блоков вокруг центрального.
   * Внизу — таймлиния (сцена №1 = 00:00–00:50, сцена №2 = 00:50–01:30 и т. д.).
   * Клик по блоку → Панель редактирования.
3. **Редактирование** (в боковой панели или всплывающем окне):
   * Текст описания + кнопка «Уточнить»:
     + Можно руками исправить текст («Я хочу общий план, а не крупный»).
     + Можно словами сказать: «Сделай ракурс дальше, меньше тумана» и т. п. → нажать «Перегенерировать».
   * LLM подхватывает изменения, возвращает обновлённую версию.
4. **Когда всё устраивает**:
   * Нажимаем «Утвердить сценарий».
5. **Промптивизация**:
   * Кнопка «Сформировать промпты» → получаем список текстовых prompt’ов (по 1 на каждую сцену).
   * Возможность посмотреть/отредактировать каждый prompt вручную, если надо.
6. **Экспорт**:
   * Копируем prompt'ы или экспортируем в JSON/YAML — дальше используем для генеративной модели.

**4. Последовательность разработки (примерный план)**

1. **MVP-сервер (backend)**:
   * Поднять простой web-сервер (на Flask/FastAPI, либо Node/Express) с двумя-тремя методами:
     + /split\_scenario (POST) — принимает текст, возвращает список сцен.
     + /refine\_scene (POST) — принимает сцену + правки, возвращает обновлённую сцену.
     + /prompts\_generate (POST) — принимает итоговый сценарий, возвращает готовые prompts.
   * Интегрировать LLM через API (OpenAI, локальную модель).
2. **MVP-клиент (frontend)**:
   * Одностраничное приложение (SPA) с формой: «Вставить текст → нажать “Генерировать”».
   * Отобразить полученные сцены в виде списка (или простого «карточного» вида для начала).
   * Сделать элементарное редактирование: «Нажми на сцену → всплывёт окно → можно поменять текст + кнопка “Перегенерировать”».
   * Сформировать prompts (пока можно просто отобразить их в текстовом формате).
3. **Расширение UI** (визуальный холст + таймлиния):
   * Подключить библиотеку для диаграмм (mindmap / canvas). Например, gojs, d3.js, jsPlumb — вариантов много.
   * Добавить блоки для сцен, соединённые «ветвями» вокруг центрального блока (исходный текст).
   * Реализовать таймлинии (там проще всего сделать «скролл-бар» с плашками сцен).
4. **Библиотека персонажей/сеттингов** (Character & Setting Bible):
   * Создать простую JSON-структуру, где хранится:
     + "Professor\_Gray": { "hairColor": "silver", "age": 60, ... }
     + "main\_lab": { "style": "modern", ... }
   * При промптивизации считывать эти данные и добавлять к каждой сцене.
5. **Тестировать и дорабатывать логику**:
   * Проверять, корректно ли LLM разбивает сценарий.
   * Настраивать prompt-инструкции, чтобы LLM выдавала именно нужный уровень детализации (крупные/средние планы и т. п.).

**5. Задание для LLM (пример формулировки)**

Чтобы LLM понимала, что мы хотим (для **/split\_scenario**):

«Вот транскрипт выступления ведущего (10 минут). Пожалуйста, раздели на логические блоки-сцены, предполагая, что у нас нет аудио. Придумай примерные таймкоды, исходя из длины текста. Каждой сцене укажи:

1. заголовок, 2) возможный отрезок по времени (start\_sec, end\_sec), 3) краткое описание того, что происходит в кадре (ракурс, действия, настроение). Не используй сразу крупные планы, если мы не уточнили. Логика: от общего к частному.»

Для **/refine\_scene**:

«Входные данные: текст сцены, правки пользователя, общая стилистика (например, “более мрачный, зловещий антураж”). Учти: пользователь запретил крупные планы до финала. Нужно обновить описание сцены и, если это влияет на следующую сцену, подкорректировать и её.»

Для **/prompts\_generate**:

«Вот список утверждённых сцен. Вот база персонажей (белый халат, очки и т.п.). Сформируй для каждой сцены подробный prompt для генеративной модели, включи описание внешности персонажа, окружения, атмосферу, ракурс, детали освещения. Упомяни все важные объекты, даже если они описаны только в предыдущей сцене, чтобы генеративная модель не потеряла контекст.»

**6. Итог**

1. У нас есть **ясная структура**: от загрузки текста → автогенерации сцен → редактирования/коррекций → итогового экспорта в виде prompt’ов.
2. Проект **не слишком громоздкий**: MVP можно собрать на минимальном UI (списке сцен) и постепенно улучшать до более наглядной «канвы».
3. Если надо, мы потом **легко сделаем новый** проект или форкнем функционал — так что смело можно начинать с относительно компактного решения.

В результате получится **удобный инструмент**, где:

* Пользователь видит **короткий путь**: «загрузить сценарий → получить draft → внести правки → нажать “промпты” → сгенерить картинки → (в будущем) можно добавить генерацию видео».
* ЛЛМ при этом выступает как гибкий ассистент, но **все ключевые решения** (как именно выглядит сцена) остаются за человеком.

Надеюсь, такой план поможет **сформулировать чёткое ТЗ** и начать реализацию!

**Вы сказали:**

Посмотри на то что мы обсуждали попытайся представить как визуально должен выглядеть такой сайт, Составь промпт на генерацию дизайна canvas\_stories

**ChatGPT сказал:**

Вот подробный промпт на генерацию дизайна интерфейса для сайта **Canvas\_Stories**, учитывающий всё, что мы обсудили:

**Prompt:**

**Create a clean, intuitive, and minimalistic web interface design for a web application called "Canvas\_Stories".**

**Key features and visual layout:**

* **Main Canvas Area:**  
  A central canvas or workspace where users input their scenario/script into a central, prominent "initial node" block. Around this central block, smaller nodes representing "Scenes" branch out visually, interconnected by sleek, soft, thin lines or subtle curves. Each node clearly shows the title of the scene, brief description, and estimated timestamps.
* **Timeline at Bottom:**  
  Below the main canvas, implement a clear horizontal timeline, visually divided into segments corresponding to scenes. Each scene-segment should be labeled with a small readable timestamp, and hovering over any segment should highlight the associated scene-node in the canvas.
* **Editing Panel:**  
  A collapsible side-panel (right-hand side) designed for editing and refining individual scene nodes. Include fields for detailed text input, a prompt editor area, a section for user notes and comments, and buttons clearly labeled "Regenerate Scene" and "Save Changes".
* **Prompt Generation Panel:**  
  After finalizing scenes, users access a simple "Prompt Generation" interface (modal or collapsible panel), listing generated detailed image prompts for each scene. This section should display prompts as easily selectable, copyable text blocks, each clearly separated, possibly accompanied by a small thumbnail placeholder.
* **Color Scheme:**  
  Use a modern, tech-friendly palette: predominantly white or very light background, accented with soft shades of blue, teal, or lavender for nodes and interactive elements. Text and lines in dark gray or black for readability.
* **Typography:**  
  Select a minimalistic, clean font (e.g., Inter, Roboto, or similar) ensuring good readability. Headers should be bold and distinct; body text clear and not too dense.
* **Interactive Elements (Buttons and Sliders):**  
  Buttons with gentle rounded corners and subtle shadow effects to indicate interactivity. Consider sliders or draggable elements for adjusting timeline segments.
* **Consistency & Clarity:**  
  Maintain visual consistency throughout. Clearly distinguish different types of interactive nodes (initial input node vs generated scene nodes) by size or subtle color differentiation.

**Inspiration/Reference styles:**

* Modern productivity apps (e.g., Notion, Miro, Figma)
* Minimalistic mindmap/diagramming tools
* Soft, unobtrusive UI animations/transitions

**Layout specifics (suggested):**

* Top bar: minimalistic logo "Canvas\_Stories" (sans-serif, modern style, subtle branding). Possibly a simple menu (Home | Projects | Help).
* Main workspace in the center: canvas, draggable and zoomable.
* Bottom of screen: Interactive timeline stretching horizontally, clearly segmentable.
* Right sidebar: collapsible editing tools and prompt-generation controls.

**Emotional style:**  
Creative, professional, innovative yet approachable.

Такой промпт поможет нейросети или дизайнеру ясно понять задачу и создать подходящий интерфейс для сайта **Canvas\_Stories**.

**Вы сказали:**

Да только в темной теме нужно

**ChatGPT сказал:**

Вот обновлённый промпт, полностью адаптированный под **тёмную тему**:

**Prompt:**

**Create a modern, minimalistic, and intuitive UI design concept in a sleek dark mode for a web application called "Canvas\_Stories".**

**Key visual features:**

* **Main Canvas Area (Central Workspace)**  
  Central node ("initial node") prominently placed at the center.  
  Surrounding nodes ("scenes") branching outward from the center, interconnected by thin, elegant lines.  
  Nodes should appear slightly elevated with subtle drop shadows or glow effects for depth and interactivity.
* **Interactive Timeline (Bottom Panel)**  
  Clearly defined horizontal timeline at the bottom.  
  Scene segments separated visually, with clear, readable timestamps (e.g., 00:00–01:20).  
  Hovering over a segment highlights corresponding nodes on the canvas through subtle glow or border effects.
* **Right-Side Editing Panel (Collapsible)**  
  Intuitive editing tools, text fields, and interactive buttons ("Regenerate Scene", "Save Changes").  
  Sections clearly labeled: Scene Description, User Notes, Prompt Editor.  
  Buttons softly rounded, clearly clickable, with gentle hover animations.
* **Prompt Generation Panel**  
  Separate modal or integrated panel displaying generated image prompts as easy-to-copy text blocks, each with a thumbnail placeholder area, clearly separated and labeled.

**Dark Mode Palette:**

* **Background:** deep charcoal or matte black (#121212 – #181818)
* **Nodes:** shades of dark slate gray or muted navy (#1E1E2E – #2E3440), slightly brighter than background, clearly distinguishable
* **Accent Colors:** electric blue, teal, violet, or cyan highlights for interactivity (#7B61FF, #4BC1D2, #00C9A7)
* **Text:** Soft white (#F0F0F3) or very light gray for high readability
* **Connecting lines:** subtle gradients or muted shades slightly brighter than background nodes (#2F3241)

**Typography:**

* Modern sans-serif font, such as Inter, Roboto, or Montserrat
* Clear distinction between headings (bold) and body text (regular)

**Interactive Elements & Animations:**

* Gentle glow or border highlights for hover states
* Slight shadow or elevation for active elements
* Soft transitions for opening panels or modals

**Layout:**

* **Top Bar:** Minimalistic, thin logo "Canvas\_Stories", subtle navigation links ("Home | Projects | Help"), unobtrusive
* **Central Canvas:** Draggable, zoomable
* **Bottom Timeline:** Draggable segments, clearly labeled
* **Right Sidebar:** Easily collapsible for maximum workspace visibility

**Mood & Style:**

* Innovative, professional, sleek, creative
* Clear visual hierarchy
* Comfortably readable in low-light environments

**Inspirational References (for style & aesthetics):**

* Figma (dark theme)
* Adobe Creative Cloud dark interfaces
* Miro or Milanote (dark mode)

Такой промпт идеально подходит для создания **тёмной темы** и поможет получить современный, понятный и красивый дизайн интерфейса **Canvas\_Stories**.

**Вы сказали:**

у нас появилась первая визуализация это рисунок нужно дать коментарии по поводу того что тут не так

**ChatGPT сказал:**

Комментарий и рекомендации по первому варианту визуализации:

**👍 Что хорошо:**

* Чистый минимализм и аккуратность в дизайне.
* Цветовая гамма приятная и подходит для тёмной темы.
* Ясная логика и понятная структура интерфейса.

**⚠️ Что стоит доработать:**

**1. Центральная канва и узлы:**

* Сейчас узлы очень абстрактны: стоит добавить чуть больше информации прямо в узлы (например, краткое название или номер сцены, краткий текстовый сниппет).
* Возможно, стоит чуть увеличить центральный узел («Initial Node») и выделить его более явно, поскольку это основной элемент работы.
* Узлы-сцены желательно сделать менее однотипными: добавить либо иконки, либо текстовые метки для быстрого ориентирования пользователя.

**2. Таймлайн:**

* Таймлайн сейчас выглядит недостаточно информативным. Таймкоды не очень чётко считываются: нужно сделать шрифт чуть крупнее или контрастнее.
* Текущая активная сцена на таймлайне должна выделяться сильнее, например, яркой подсветкой или рамкой.
* Отметки времени выглядят сейчас условно и несколько непонятно (просто две цифры без обозначений). Желательно явно показать начало и конец каждой сцены (например, 00:00 - 00:45).

**3. Панель редактирования (боковая):**

* Кнопка «Regenerate Scene» выделена хорошо, но кнопка «Save Changes» кажется неактивной и теряется на фоне. Стоит её тоже сделать более активной или выделить при наведении.
* Поле «Prompt Editor» слишком общее: нужно явно указать, что это текст для генерации изображений и, возможно, добавить пример заполнения («Example: A scientist in a futuristic lab...»).
* Поле «User Notes» хорошо бы снабдить пояснением или подсказкой («Добавьте уточнения или комментарии к сцене здесь»).

**4. Общие моменты:**

* Не хватает явно выраженных инструментов управления канвой (например, zoom, pan или reset canvas).
* Нет индикации того, что можно добавлять новые сцены вручную прямо на канву (кнопка "+ Scene" или иконка).
* Верхнее меню выглядит минималистично и приятно, но стоит чуть увеличить расстояние между ссылками, чтобы избежать случайных нажатий.

**🎯 Резюме рекомендаций:**

* Сделать узлы более информативными и разными.
* Улучшить читаемость таймлайна (более четкие таймкоды).
* Сделать боковую панель более понятной и чуть активнее выделять кнопки.
* Добавить подсказки и пояснения для удобства пользователя.
* Предусмотреть инструменты навигации и управления канвой.

В остальном концепция выглядит уже интересно и перспективно!

**Вы сказали:**

посмотри пожжалуйста макет нашей канвы стори и почитай коментарии и скажи что тут ещо можно добавить?

**ChatGPT сказал:**

Рассуждала для пара секунд

Ниже несколько идей, что ещё можно добавить или улучшить в макет **Canvas\_Stories**, с учётом общей концепции, тёмной темы и возможностей, которые вы уже обсуждали:

**1. Подсказки (tooltips) и микрообучение**

* **Tooltips при наведении**:  
  Когда пользователь наводит курсор на кнопку («Regenerate Scene», «Save Changes», «Add Node» и т.д.), можно отображать краткие пояснения/подсказки, чтобы упростить знакомство с интерфейсом.
* **Короткое гайд-тур-всплывающее окно** при первом запуске:  
  Может объяснять основные части интерфейса (центральная канва, панель редактирования, таймлиния) и, например, инструменты зума.

**2. Кнопки управления холстом**

* **Zoom In/Out** или «Reset View»:  
  Чтобы пользователь мог быстро менять масштаб канвы и возвращаться к первоначальному виду.
* **Pan Mode** или иконка «рука»:  
  Для перетаскивания канвы.

Это добавит удобства в работе с большой mind map.

**3. Быстрое переключение между сценариями / проектами**

* Если предполагается, что пользователь может работать с несколькими сценариями, пригодится **небольшая навигационная панель** (слева или сверху), где будет список проектов/сценариев.
* Либо в верхнем меню – пункт «Project List», где можно выбрать другой проект или вернуться в «Dashboard».

**4. Более явные статусы узлов (сцен)**

1. **Статус «На доработке»** или «Draft»: если сцена была только что создана и не до конца вычитана.
2. **Статус «Утверждена»** / «Approved»: когда пользователь удовлетворён описанием сцены и сохранённым промптом.

Можно отобразить такие статусы цветными метками или маленькими иконками в узлах, чтобы сразу видеть, где требуется внимание.

**5. Отдельная вкладка «Персонажи и окружение»**

* Если у вас есть «библиотека» персонажей и локаций (Character & Setting Bible), удобно сделать **отдельную вкладку** или боковую панель, где пользователь может:
  + Посмотреть список всех персонажей и их характеристики.
  + Добавить/редактировать описание, внешность, цвет волос, костюмы и т.д.
  + То же самое для локаций: «Лаборатория», «Кабинет», «Улица» и т.п.
* Это позволит поддерживать консистентность при «промптивизации» сцен.

**6. Показать связь между таймкодовыми кусками и узлами**

* **Визуальная связка**: когда пользователь кликает на сцену на холсте, сразу подсвечивается соответствующий участок на таймлинии (и наоборот).
* **Возможность перетаскивать сцену на таймлинии**: менять её порядок или протяжённость во времени. В результате узел на канве меняет позицию или порядок в структуре.

**7. История изменений (версионность)**

* Если есть возможность, сделайте **простую историю**: при нажатии правки (Regenerate Scene) можно откатиться к предыдущей версии описания, если новое не понравилось.
* Это особенно полезно для итеративной работы.

**8. Прямое добавление комментариев к узлам**

* Иногда нужно оставить короткий комментарий или метку (например, «Сцену стоит уточнить – нет костюмов для статистов»).
* Реализовать это можно в виде маленькой иконки «Комментарий» на самом узле, открывающей всплывающее окошко или боковую панель с заметками, чтобы сразу видеть, что в этой сцене ещё есть нерешённые вопросы.

**9. «Глобальные» стилистические настройки**

* Кнопка/панель **«Global Style Settings»**:
  + Позволяет задать общий художественный стиль (реалистичный, мультяшный, sci-fi, фэнтези), ключевые цвета, атмосферу.
  + При желании можно переключаться между несколькими сохранёнными наборами стиля.

Эти настройки могут влиять на автоматическое «разбиение» и «промптивизацию», а также подсказывать ЛЛМ нужный тон для всей сцены.

**10. Предпросмотр результатов (если есть план прямой генерации)**

* Если в будущем появится прямое подключение к генеративному сервису (Midjourney API, Stable Diffusion web-UI и т.п.), вы можете добавить **мини-превью**:
  + Нажать «Generate Image» → получить маленький вариант сгенерированного изображения в интерфейсе.
  + Полезно для быстрых правок без постоянной ручной копипасты.

**Итог**

Все эти элементы делают интерфейс более удобным и понятным. Главное — **не перегрузить** UI. Проект можно развивать итеративно, постепенно добавляя новые функции:

1. Сначала базовый холст + таймлиния + редактирование сцен.
2. Потом — подсказки, статусы, история версий, библиотека персонажей.
3. И уже на продвинутых этапах — глобальные стили, предпросмотры генеративных изображений и т.д.

Таким образом, вы получите **мощный, но при этом достаточно гибкий** инструмент для работы над сценарием, визуальной разбивкой и последующей генерацией промптов.

так я тут провела небольшой опрос по нашему проекту и нам отсыпали немного технической инфоормации посмотри пожалуйста напиши что хорошо что плохо с чем согласен а что можно сделать лучше? Анализ проекта "Canvas\_Stories" и рекомендации 1. Сильные стороны текущего описания: Чётко определена основная цель: визуализация сценариев через интерактивную канву с использованием локальной LLM. Прописаны ключевые функции: разбиение текста на сцены, генерация подсказок для визуализации, настройка стиля, экспорт в промпты. Учтена тёмная тема интерфейса, что улучшает пользовательский опыт. Предусмотрена интеграция с генеративными моделями (например, для создания изображений). 2. Пробелы и вопросы, требующие уточнения: 2.1. Взаимодействие с интерфейсом: Перетаскивание узлов: Как пользователь будет менять структуру канвы? Нужны ли инструменты для ручного перемещения узлов, изменения связей? Сохранение правок: Как часто автосохранение? Есть ли история изменений (undo/redo)? Навигация по крупным проектам: Как работать с канвой, если сцен больше 50? Нужны ли фильтры или поиск по узлам? 2.2. Локальная LLM: Требования к оборудованию: Какие минимальные характеристики GPU/CPU/RAM нужны для работы? Как это повлияет на пользователей без мощного железа? Обновление моделей: Как пользователь будет обновлять LLM? Будет ли поддержка новых версий моделей (например, Llama 3)? Обработка больших текстов: Как система справляется с текстами длиннее 10 тыс. токенов? Есть ли оптимизации (например, разбиение на части)? 2.3. Управление стилями и консистентностью: Библиотека персонажей: Как создаётся и редактируется? Есть ли шаблоны для быстрого добавления (например, "учёный", "детектив")? Автоматизация: Может ли LLM сама предлагать характеристики персонажей на основе контекста, или всё вводится вручную? Конфликты стилей: Что делать, если пользователь вручную меняет описание персонажа в одной сцене, но не во всех? 2.4. Экспорт промптов: Форматы экспорта: Только текстовые файлы, или поддержка JSON/CSV для интеграции с другими инструментами? Кастомизация промптов: Можно ли добавлять параметры для конкретных генеративных моделей (например, --ar 16:9 для Midjourney)? Пакетная генерация: Можно ли экспортировать промпты для всех сцен сразу, или только по одной? 2.5. Безопасность и приватность: Шифрование данных: Как защищаются пользовательские проекты? Есть ли возможность использования HTTPS и шифрования баз данных? Локальное хранение: Где хранятся проекты (на устройстве пользователя или в облаке)? Как обеспечить резервное копирование? 2.6. Интеграция с внешними сервисами: Видео-редакторы: Поддержка ли форматов XML для Premiere Pro или DaVinci Resolve? Облачные хранилища: Интеграция с Google Drive, Dropbox и т.д. для синхронизации проектов. 3. Рекомендации по доработке: 3.1. Дополнить интерфейс: Добавить инструменты для ручного управления: Перетаскивание узлов. Изменение толщины/цвета связей. Группировка сцен по категориям (например, "Диалоги", "Экшн"). Реализовать фильтры и поиск по ключевым словам или тегам. Ввести режимы просмотра: радиальная схема, древовидная структура, таймлайн. 3.2. Улучшить работу с LLM: Добавить прогресс-бар для длительных операций (например, генерация 50 сцен). Реализовать кеширование запросов, чтобы повторная генерация сцен с небольшими правками не требовала полного пересчёта. Написать документацию по настройке локальной LLM для разных ОС (Windows, Linux, macOS). 3.3. Расширить функционал библиотеки: Создать шаблоны персонажей и локаций (например, "Лаборатория в стиле киберпанк", "Средневековый замок"). Добавить автодополнение характеристик: при вводе "седой учёный" LLM автоматически предлагает детали (очки, халат, возраст). Реализовать проверку консистентности: система предупреждает, если в разных сценах персонаж описывается по-разному. 3.4. Усилить безопасность: Внедрить авторизацию (например, через JWT-токены). Добавить шифрование проектов на стороне клиента (например, с помощью AES-256). Реализовать экспорт проектов в зашифрованные архивы с паролем. 3.5. Оптимизировать экспорт: Поддержка формата CSV для импорта в Excel или Google Sheets. Добавить пресеты для генеративных моделей (Stable Diffusion, Midjourney, DALL-E). Реализовать пакетную генерацию изображений через API (например, параллельные запросы к Stable Diffusion). 4. Потенциальные проблемы и их решение: Проблема Решение Низкая производительность локальной LLM на слабых ПК Добавить опцию использования облачной LLM (как fallback). Конфликты версий моделей (например, GGUF vs GPTQ) Создать конвертер форматов встроенный в интерфейс. Сложность поддержания консистентности в больших проектах Ввести систему тегов и автоматических напоминаний (например, "Персонаж X изменён в сцене 5 — обновить другие сцены?"). Перегруженность интерфейса Добавить режим "Минимум" (скрыть таймлайн и панели, оставить только канву). 5. Итоговые шаги: Создать MVP: базовая канва + разбиение текста на сцены + экспорт в текстовые промпты. Провести юзабилити-тестирование с фокус-группой (авторы, режиссёры). Доработать интеграцию с LLM: оптимизация скорости, обработка ошибок. Реализовать расширенные функции: библиотека персонажей, пакетная генерация, интеграция с видео-редакторами. Запустить документацию и поддержку: видео-гайды, FAQ, форма для обратной связи. Проект имеет большой потенциал, но требует тщательной проработки технических деталей и фокуса на пользовательском опыте. Удачи в реализации! 🚀 так а ты можешь сам ответить на эти вопросы? Хорошо, давайте разберёмся с поставленными вопросами и попробуем дать на них развёрнутые ответы, основываясь на обсуждённых ранее идеях и технических возможностях проекта. ### \*\*1. Взаимодействие с интерфейсом\*\* \*\*Перетаскивание узлов:\*\* Пользователь сможет перемещать узлы на канве с помощью drag-and-drop. Для этого можно использовать библиотеки, такие как React-DnD или Konva.js, которые поддерживают интерактивное управление элементами. Связи между узлами будут автоматически перерисовываться при перемещении. \*\*Сохранение правок:\*\* Автосохранение будет происходить каждые 30 секунд или при закрытии вкладки. История изменений (undo/redo) реализуется через хранение стейта в Redux или аналогичном менеджере состояний, позволяя откатываться на несколько шагов назад. \*\*Навигация по крупным проектам:\*\* Добавим фильтры по тегам (например, "Диалоги", "Экшн") и поиск по ключевым словам. Для проектов с большим количеством сцен можно реализовать бесконечный скролл или группировку узлов по категориям. --- ### \*\*2. Локальная LLM\*\* \*\*Требования к оборудованию:\*\* Минимальные требования: - CPU: 8 ядер (Intel i7 или аналогичный). - RAM: 16 ГБ (32 ГБ для моделей >7B параметров). - GPU: NVIDIA RTX 3060 (12 ГБ VRAM) для работы с квантованными моделями. Для пользователей без мощного железа можно предложить облачный вариант с API. \*\*Обновление моделей:\*\* Пользователи смогут загружать новые модели в формате GGUF через интерфейс. Система будет проверять совместимость и предлагать конвертацию при необходимости. Поддержка новых версий (например, Llama 3) будет добавлена через обновления приложения. \*\*Обработка больших текстов:\*\* Тексты длиннее 10 тыс. токенов будут разбиваться на чанки. LLM обработает каждый чанк отдельно, а результаты объединятся в единую структуру. Для оптимизации можно использовать алгоритмы сжатия контекста (например, sliding window). --- ### \*\*3. Управление стилями и консистентностью\*\* \*\*Библиотека персонажей:\*\* Пользователь сможет создавать шаблоны через JSON-редактор или графический интерфейс. Пример шаблона:

json

{

"character": "Профессор Грей",

"age": 60,

"appearance": "седые волосы, очки, белый халат",

"traits": "любопытный, слегка суетливый"

}

LLM будет автоматически предлагать характеристики на основе контекста, но пользователь может их корректировать. \*\*Конфликты стилей:\*\* Система будет проверять описания персонажей во всех сценах и выдавать предупреждения при несоответствиях. Например: > «В сцене 5 профессор Грей описан как "брюнет". Это противоречит глобальному шаблону. Исправить?» --- ### \*\*4. Экспорт промптов\*\* \*\*Форматы экспорта:\*\* - Текстовые файлы (TXT, MD). - JSON для интеграции с API генеративных моделей. - CSV для анализа в таблицах. \*\*Кастомизация промптов:\*\* Пользователь сможет добавлять параметры через шаблоны:

{scene\_description}, {character\_details}, --ar 16:9, --v 6

\*\*Пакетная генерация:\*\* Экспорт всех промптов сразу в ZIP-архив с структурированными папками по сценам. --- ### \*\*5. Безопасность и приватность\*\* \*\*Шифрование данных:\*\* Проекты будут шифроваться на стороне клиента с использованием AES-256. Пароль для дешифровки задаёт пользователь. \*\*Локальное хранение:\*\* Данные сохраняются в IndexedDB или локальных файлах (через Electron для desktop-версии). Резервные копии можно экспортировать в зашифрованные файлы. --- ### \*\*6. Интеграция с внешними сервисами\*\* \*\*Видео-редакторы:\*\* Экспорт в XML-форматы, совместимые с Premiere Pro и DaVinci Resolve. Пример структуры:

xml

<clip name="Сцена 1" start="00:00:00" end="00:01:30">

<prompt>Учёный в лаборатории...</prompt>

</clip>

\*\*Облачные хранилища:\*\* Интеграция с Google Drive через REST API. Пользователи смогут синхронизировать проекты в фоновом режиме. --- ### \*\*7. Потенциальные проблемы и решения\*\* \*\*Низкая производительность LLM:\*\* - Режим "Lite": использование моделей с 3B параметрами. - Оффлайн-кеширование: результаты генерации сохраняются для повторного использования. \*\*Конфликты версий моделей:\*\* Встроенный конвертер моделей (например, через llama.cpp) позволит преобразовывать форматы (GGUF → GPTQ). --- ### \*\*8. Итоговый план разработки\*\* 1. \*\*Этап 1 (2 недели):\*\* - Базовая канва с drag-and-drop. - Интеграция локальной LLM через LM Studio. - Экспорт промптов в TXT. 2. \*\*Этап 2 (3 недели):\*\* - Библиотека персонажей. - История изменений. - Поддержка облачных LLM (OpenAI API как fallback). 3. \*\*Этап 3 (4 недели):\*\* - Интеграция с Stable Diffusion. - Экспорт в XML/CSV. - Тестирование с фокус-группой. 4. \*\*Этап 4 (1 неделя):\*\* - Документация и релиз. --- Этот план обеспечит поэтапную реализацию проекта с учётом всех обсуждённых требований и потенциальных сложностей. Ответы на ключевые вопросы проекта "Canvas\_Stories" 1. Взаимодействие с интерфейсом Перетаскивание узлов: Реализовано через библиотеку Konva.js. Пользователь может перемещать узлы, а связи автоматически перестраиваются. Для группировки сцен добавлены инструменты выделения и контекстное меню. Сохранение правок: Автосохранение каждые 30 секунд. История изменений (undo/redo) хранит до 50 последних действий. Навигация: Фильтры по тегам (например, "Диалоги") и поиск по ключевым словам. Для больших проектов — группировка узлов и свёртывание ветвей. 2. Локальная LLM Требования: Минимальные: CPU 8 ядер (Intel i7), RAM 16 ГБ, GPU RTX 3060. Для слабых ПК — облачный режим с API. Обновление моделей: Пользователи загружают модели в формате GGUF через интерфейс. Конвертер поддерживает Llama 3, Mistral и другие. Большие тексты: Текст разбивается на чанки по 2048 токенов. Используется алгоритм sliding window для сохранения контекста. 3. Управление стилями Библиотека персонажей: Графический редактор с шаблонами. Пример шаблона: json Copy { "name": "Учёный", "details": "очки, халат, седые волосы" } Конфликты стилей: Система проверяет несоответствия и предлагает исправления. Например: «В сцене 5 учёный без очков. Обновить все сцены? [Да/Нет]». 4. Экспорт промптов Форматы: TXT, JSON, CSV. Для видео-редакторов — XML с таймкодами: xml Copy <scene start="00:01:30" prompt="Учёный в лаборатории..."/> Run HTML Кастомизация: Пользователь добавляет параметры в шаблон: python Copy "{описание}, cinematic, 8k --ar 16:9" Пакетная генерация: Экспорт всех промптов в ZIP. Для Stable Diffusion — параллельные запросы через API. 5. Безопасность Шифрование: AES-256 на стороне клиента. Ключ хранится локально. Локальное хранение: Данные сохраняются в IndexedDB. Резервные копии — в зашифрованных ZIP-архивах. 6. Интеграция Видео-редакторы: Экспорт в XML для Premiere Pro и DaVinci Resolve. Облако: Синхронизация с Google Drive через OAuth 2.0. 7. Проблемы и решения Проблема Решение Низкая производительность Режим "Lite" с моделями 3B параметров. Конфликты версий моделей Встроенный конвертер GGUF → GPTQ. Перегруженность интерфейса Режим "Минимум" (скрывает таймлайн). 8. План разработки MVP (2 недели): Канва + генерация сцен + экспорт в TXT. Расширение (3 недели): Библиотека персонажей + интеграция с Stable Diffusion. Финальная доработка (4 недели): XML-экспорт + облачная синхронизация + документация.

**ChatGPT сказал:**

\*\*Причинно рассмотренные аспекты проекта "Canvas\_Stories"\*\* для пара секунд

Ниже даю общий анализ того, что хорошо проработано в этом техническом описании «Canvas\_Stories», а что можно было бы ещё улучшить или пересмотреть:

**Плюсы и сильные стороны**

1. **Системность и полнота описания**
   * Упомянуты все основные аспекты проекта: интерфейс (канва), работа с локальной LLM, хранение, безопасность, экспорты в разные форматы.
   * Разделение на блоки (интерфейс, LLM, стили, экспорт и т. д.) упрощает понимание проекта.
2. **Учёт практических сценариев**
   * Функции вроде перетаскивания узлов, автосохранения с историей изменений, фильтров, разбивки большого текста на чанки — это то, что действительно понадобится в реальном использовании.
   * Поддержка разных моделей (GGUF, GPTQ) и возможные проблемы с совместимостью — очень важная техническая деталь.
3. **Меры безопасности**
   * Шифрование данных (AES-256), хранение локально и экспорт зашифрованных бэкапов.
   * Указана перспектива облачного API для тех, у кого слабое «железо».
4. **Гибкий экспорт**
   * TXT, JSON, CSV, XML, ZIP — покрывает широкий спектр случаев: от простого текстового вывода до интеграции с видеоредакторами (DaVinci, Premiere).
   * Пакетная генерация промптов с параллельными запросами к SD – тоже удобно для продвинутых пользователей.
5. **Представлен план разработки**
   * Разбитый на этапы (MVP, расширение, финальная доработка).
   * Указывает примерные сроки (2 недели / 3 недели / 4 недели), даёт представление о дорожной карте.

**Возможные улучшения и уточнения**

1. **Управление очень большими проектами**
   * Вопрос о том, **как** именно будут отображаться, например, 100+ сцен на одной канве, остаётся довольно открытым. В рекомендациях есть упоминание группировки и сворачивания ветвей, но иногда требуется полноценная «иерархическая» структура (двухуровневая, трёхуровневая).
   * Можно предложить **альтернативные режимы** отображения: радиальный/матричный/древовидный. Так пользователь при желании может переключиться на «древовидный список» и быстро находить нужную сцену.
2. **Версионность и коллаборация**
   * Упоминается undo/redo, но если проект растёт, может понадобиться «ветвление» версий (как в Git).
   * Если планируется использовать проект несколькими людьми, нужен «live-collab» или хотя бы система «check-in/check-out». Пока это не описано, но может стать важным при реальной работе в команде.
3. **Пользовательские роли и доступ**
   * Безопасность описана, но нет явного указания, будут ли **роли** (админ, редактор, просмотр). Если проект ориентирован на студии, где может быть несколько участников, нужно учитывать разграничение прав.
4. **Гибкость таймкодов**
   * Упомянута автоматическая разбивка (start/end), но в реальности пользователи могут захотеть вручную редактировать таймкоды (например, переписать 00:01:30 → 00:01:45).
   * В планах разработки стоит упомянуть возможность ручного управления таймкодами, а также синхронизации, если в будущем появится аудио-дорожка.
5. **Технические детали локальной LLM**
   * Минимальные требования (CPU 8 ядер, RAM 16ГБ, GPU RTX 3060) — разумно, но если модель 13B+ параметров, иногда 3060 может не хватать для комфортной скорости. Стоит чётче оговорить, что производительность будет зависеть от степени квантования (4bit, 8bit и т. п.).
   * «Использование облачного режима» — хорошо, но желательно расписать, **как** технически это будет выглядеть (отдельный ключ, план оплаты, если это OpenAI, и т. д.).
6. **Автоматическая проверка на противоречия**
   * Идея крутая («В сцене 5 учёный без очков, а глобально он с очками»), но реализация может быть сложнее, чем кажется. Тут нужно как минимум **краткое описание алгоритма** или принципов, по которым система ищет несоответствия.
   * Могут быть моменты, где персонаж снимает очки по сюжету, и это не ошибка. Значит, нужен UI, который позволит «подтвердить» намеренное различие или «исправить» ошибку.
7. **Мульти-язычность интерфейса и контента**
   * Возможно, нужно учитывать языки интерфейса и поддержку разных языков сценария. Если проект планируется интернациональным, стоит оговорить возможность локализации.
8. **Дополнение к генерации**
   * Упомянута генерация изображений и экспорт к видео-редакторам. Если в планах есть и text-to-video (например, через Gen-2, RunwayML), стоит хотя бы намекнуть, как это может выглядеть (понятно, это пока вещь будущего).
9. **Ограничения по длине контекста**
   * Разбиение большого текста на чанки — отличная идея, но как именно мы «склеиваем» итоги? Понадобятся эвристики для определённых случаев (когда предложение «разорвано» посередине чанка). Возможно, нужно указать, что используется «sliding window» или «smart chunking» (на уровне параграфов, сценарных блоков).
10. **Расширенная аналитика**

* Может понадобиться блок для **аналитики**: сколько у нас сцен, сколько персонажей, насколько проект «консистентен». Это не критично для MVP, но для крупных проектов может быть полезно.

**Итоговая оценка**

* **В целом**: техническое описание выглядит весьма проработанным и реалистичным. Хорошо, что учтена локальная LLM, вопросы совместимости форматов, а также продуманы детали безопасности и экспорта.
* **Что можно усилить**: проработать крайние кейсы (огромное число сцен, совместная работа, тонкая проверка консистентности персонажей), расширить функционал интерфейса (разные режимы просмотра, интеграция undo/redo на уровне ветвления проектов) и уточнить детали облачного режима.

Если финальным пользователям нужен лёгкий, понятный инструмент для коротких роликов, текущий подход достаточно убедителен. Если амбиции масштабируются (полнометражные сценарии, коллективная работа), тогда стоит более подробно описать коллаборацию, версионность и удобство навигации по огромным сценариям.

\*\*Техническое задание на разработку интерфейса "Canvas\_Stories"\*\*

---

### \*\*1. Общие требования\*\*

- \*\*Цель\*\*: Создать интуитивный и функциональный интерфейс для визуализации сценариев, генерации идей и управления сценами.

- \*\*Стиль\*\*: Минимализм, тёмная тема (основной фон: #121212, акценты: #7B61FF, #4BC1D2), читаемые шрифты (Inter, Roboto).

- \*\*Адаптивность\*\*: Поддержка десктопов (от 1280px) и планшетов (768px+).

---

### \*\*2. Основные компоненты интерфейса\*\*

#### \*\*2.1. Центральная канва\*\*

- \*\*Функционал\*\*:

- Ввод текста/запроса в центральный блок (размер: 400x200px, фон: #1E1E2E, текст: #F0F0F3).

- Ветвящиеся ноды (сцены/идеи) с возможностью перетаскивания и изменения связей.

- Анимация появления/исчезновения нод (плавное масштабирование + fade-in).

- Zoom/Pan канвы (кнопки +/- в правом нижнем углу, drag-to-pan).

- \*\*Визуал\*\*:

- Ноды: округлые блоки (радиус: 8px) с заголовком, кратким описанием и иконкой типа (текст/видео/графика).

- Связи между нодами: тонкие линии с градиентом (#7B61FF → #4BC1D2).

#### \*\*2.2. Таймлайн\*\*

- \*\*Расположение\*\*: Фиксированная панель внизу экрана (высота: 80px).

- \*\*Элементы\*\*:

- Горизонтальная шкала с делениями (шаг: 10 секунд).

- Сегменты сцен (цвет соответствует типу: ведущий — синий, визуал — фиолетовый).

- Ховер-эффекты: подсветка сегмента и связанной ноды.

- Drag-and-drop для изменения длительности сцен.

#### \*\*2.3. Боковая панель редактирования\*\*

- \*\*Функционал\*\*:

- Редактор текста сцены (поля: заголовок, описание, таймкоды).

- Кнопки действий: «Перегенерировать», «Сохранить», «Удалить».

- Секция «Комментарии» для заметок пользователя.

- Выбор стиля сцены (реализм, фэнтези, научпоп).

- \*\*Визуал\*\*:

- Панель открывается/закрывается по клику на ноду (ширина: 320px).

- Акцентные кнопки: «Перегенерировать» — #7B61FF, «Удалить» — #FF6161.

#### \*\*2.4. Панель промптов\*\*

- \*\*Расположение\*\*: Модальное окно или правая панель.

- \*\*Элементы\*\*:

- Список сгенерированных промптов с кнопкой «Копировать».

- Предпросмотр миниатюр (заглушки для будущих изображений).

- Экспорт в TXT/JSON (кнопка «Скачать всё»).

---

### \*\*3. Интерактивность\*\*

- \*\*Хоткеи\*\*:

- `Ctrl + Z` / `Ctrl + Y` — отмена/повтор действий.

- `Ctrl + S` — автосохранение.

- \*\*Анимации\*\*:

- Плавное появление боковых панелей (transition: 0.3s ease).

- Микровзаимодействия при наведении (подсветка, тени).

---

### \*\*4. Технические требования\*\*

- \*\*Библиотеки\*\*:

- Рендер канвы: Konva.js или D3.js.

- Управление состоянием: Redux/Zustand.

- Стили: Tailwind CSS + SCSS.

- \*\*Совместимость\*\*:

- Браузеры: Chrome, Firefox, Safari (последние 2 версии).

---

### \*\*5. Макеты и примеры\*\*

- \*\*Ссылка на Figma\*\*: [Прототип в Figma](https://figma.com/file/...)

(включает все состояния: hover, active, редактирование).

---

\*\*Итог\*\*: Интерфейс должен обеспечивать плавную работу с 50+ нодами, мгновенный отклик на действия пользователя и чёткую визуальную иерархию. Готово к интеграции с LLM и генеративными моделями.