**[10:00] [Заказчик]**  
Добрый день, коллеги. Давайте начнём с обсуждения общих требований. Первым пунктом — регистрация и аутентификация пользователей.

* Нам нужна возможность регистрации нового пользователя с обязательной валидацией полей (логин, email, пароль).
* Пароли должны храниться в зашифрованном виде, с использованием надёжных алгоритмов хеширования и соли.
* Для аутентификации планируем использовать JWT-токены, которые будем хранить в cookies.
* В качестве дополнительной опции — возможность восстановления пароля через email.

**[10:05] [Команда]**  
Здравствуйте! По первому пункту — можем реализовать регистрацию с обязательной валидацией на клиенте и сервере. Для шифрования паролей предлагаем использовать bcrypt или Argon2.  
Также уточняем: для восстановления пароля планируется отправка ссылки с одноразовым токеном?

**[10:07] [Заказчик]**  
Именно, ссылка должна быть активной в течение ограниченного времени (например, 15–30 минут). Убедитесь, что реализована проверка срока действия токена.

**[10:10] [Команда]**  
Отлично, примем к сведению. Переходим к следующему блоку — игровой процесс. Опишите, пожалуйста, детали.

**[10:12] [Заказчик]**  
Основные моменты игрового процесса:

* Каждый раунд начинается с генерации координат из заранее составленного списка локаций Барнаула.
* Необходимо избегать повторного использования координат в рамках одной игровой сессии, чтобы не повторять раунды.
* При старте раунда подгружаются панорамные изображения через API Яндекс.Панорамы, которые соответствуют сгенерированным координатам.
* Пользователь выбирает предполагаемое местоположение на интерактивной карте, реализованной с помощью Leaflet.js.
* Система подсчитывает результат на основе расстояния между выбранной точкой и реальным местоположением, фиксируя очки, время и количество попыток.
* После завершения раунда выводится итоговая таблица результатов.

**[10:18] [Команда]**  
Поняли. Есть несколько вопросов:

* Нужно ли фиксировать отдельные попытки внутри одного раунда или учитывать только финальный выбор?
* Какой метод подсчёта очков предпочтительнее — линейная зависимость от расстояния или более сложная формула?

**[10:20] [Заказчик]**  
Отличный вопрос. Хотелось бы, чтобы фиксировались как попытки, так и финальный выбор. Формула может быть следующей: базовые очки минус штраф за расстояние (с шагом, например, 1 очко за 100 метров, с минимальным порогом). Но оставляем возможность на доработку после тестирования.

**[10:22] [Команда]**  
Хорошо, запишем это. Теперь обсудим личный кабинет пользователя.

**[10:23] [Заказчик]**  
В личном кабинете должны быть следующие возможности:

* Просмотр и редактирование профиля (имя, логин, email, аватар).
* Изменение настроек, например, для указания API-ключей или параметров игрового процесса.
* Просмотр истории игр с возможностью фильтрации и сортировки по различным параметрам (дата, очки, расстояние).
* Хранение детальной информации о каждом раунде: дата, время, набранные очки, количество попыток.

**[10:28] [Команда]**  
Хорошо, реализуем многостраничный личный кабинет с вкладками «Профиль», «Настройки» и «История игр». Возможно, добавим возможность экспорта истории в CSV или PDF — это будет полезно для аналитики.

**[10:30] [Заказчик]**  
Это интересное предложение, можем обсудить на следующем этапе. Переходим к обсуждению интерфейса и взаимодействия с пользователем.

**[10:32] [Заказчик]**  
Нам важно, чтобы:

* Пользователь всегда имел быстрый доступ к истории и настройкам из личного кабинета.
* Формы регистрации, авторизации и редактирования профиля работали динамично, без перезагрузки страниц (используем асинхронные запросы).
* Сообщения об ошибках и уведомления отображались на экране сразу и понятно.
* По возможности, предусмотреть переключение между тёмной и светлой темами.
* Интерфейс должен быть адаптивным и корректно отображаться на ПК, планшетах и смартфонах.

**[10:37] [Команда]**  
Согласны. Используем современный SPA-подход с AJAX или fetch-запросами для динамики. Темы можем реализовать через CSS-переменные с возможностью переключения. Будет проведено тестирование на основных устройствах.

**[10:40] [Заказчик]**  
Далее идёт вопрос администрирования. Что включить в этот блок?

**[10:42] [Заказчик]**  
Нам важно:

* Управление пользовательскими сессиями: автоматическое завершение сессии по истечении срока действия токена, принудительное завершение сессии при подозрении на компрометацию.
* Защита API-маршрутов через проверку токенов и ролей пользователей.
* Логирование всех ключевых событий: успешные запросы, ошибки, действия пользователей, изменения в базе данных.

**[10:45] [Команда]**  
Мы можем реализовать middleware для проверки JWT и логировать события через специализированный логгер. Возможно, стоит также реализовать админ-панель для мониторинга активности пользователей?

**[10:47] [Заказчик]**  
Идея с админ-панелью хорошая, но пока на базовом уровне — просто логирование и возможность просмотра логов через специальный API-эндпоинт.

**[10:48] [Команда]**  
Записываем. Далее обсудим обработку ошибок и уведомления.

**[10:50] [Заказчик]**  
Система должна:

* Отображать понятные сообщения об ошибках (например, неверный логин/пароль, некорректные координаты).
* Иметь механизм для восстановления работы после критических сбоев, например, перезагрузку игровых данных или повторный запрос к серверу.
* Показывать всплывающие уведомления о важных событиях, таких как завершение раунда или успешное сохранение профиля.

**[10:53] [Команда]**  
Реализуем централизованный обработчик ошибок как на сервере, так и на клиенте. Используем toast-уведомления для вывода сообщений. Также добавим автоматическую перезагрузку данных в случае временной недоступности сервера.

**[10:55] [Заказчик]**  
Хорошо. Теперь обсудим дополнительные возможности и расширения.

**[10:57] [Заказчик]**  
Мы хотим, чтобы:

* Интерфейс поддерживал многоязычность (хотя сначала достаточно двух языков, можно добавить и больше в будущем).
* В будущем предусмотреть интеграцию с другими картографическими сервисами помимо Яндекс.Панорамы.
* Позже добавить возможность создания пользователями собственных маршрутов и заданий, расширяя игровой процесс.

**[11:00] [Команда]**  
Отлично, заложим основу для локализации и модульной архитектуры, чтобы в будущем было легко добавить новые функциональности. Интерфейс построим с учётом расширяемости.

**[11:02] [Заказчик]**  
Переходим к вопросу монетизации. Какие варианты вы предлагаете?

**[11:03] [Заказчик]**  
Предлагается:

* Внедрение спонсорских блоков и партнёрских программ.
* Размещение рекламы (баннеры, видео, реклама с наградой, локальная реклама).
* Возможность платной подписки для доступа к расширенным функциям или эксклюзивному контенту.

**[11:05] [Команда]**  
Реализуем рекламный модуль, который можно будет легко отключить или настроить в зависимости от потребностей. Подписку можно реализовать через разделение ролей пользователей, интегрировав платёжные системы на следующем этапе.

**[11:07] [Заказчик]**  
Отлично, теперь обсудим нефункциональные требования.

**[11:08] [Заказчик]**  
Нефункциональные требования следующие:

* **Производительность:**
  + Быстрая загрузка панорам и отзывчивость интерфейса.
  + Асинхронная обработка запросов на сервере (например, через FastAPI и Uvicorn).
* **Надёжность и доступность:**
  + Обеспечение круглосуточной работы приложения, минимизация простоев.
  + Резервное копирование данных и возможность быстрого восстановления после сбоев.
* **Безопасность:**
  + HTTPS для шифрования трафика, хранение паролей с использованием надёжного хеширования.
  + Защита API через JWT и регулярное обновление зависимостей.
* **Удобство использования:**
  + Интуитивно понятный интерфейс, адаптивная вёрстка.
  + Поддержка основных браузеров (Chrome, Firefox, Edge, Safari).
* **Масштабируемость:**
  + Возможность горизонтального и вертикального масштабирования, использование контейнеризации (Docker) и микросервисной архитектуры.
* **Тестирование:**
  + Проведение ручного и автоматизированного тестирования функционала, API и безопасности.

**[11:15] [Команда]**  
Отлично, по производительности используем асинхронные вызовы, что повысит скорость работы приложения. Планируется реализация резервного копирования через регулярные снепшоты базы данных. Автоматизация тестирования через CI/CD поможет выявлять проблемы на раннем этапе.

**[11:18] [Заказчик]**  
Также нужно предусмотреть автоматизацию развёртывания. Важно, чтобы сборка, тестирование и деплой происходили без значительных простоев, а развертывание происходило в контейнерах Docker.

**[11:20] [Команда]**  
Это мы уже обсуждали. Настроим CI/CD pipeline с использованием GitLab CI или GitHub Actions, чтобы каждый коммит проходил через тесты, а деплой осуществлялся автоматически.

**[11:22] [Заказчик]**  
Еще один момент — совместимость с разными ОС. Приложение должно корректно работать на Unix/Linux серверах.

**[11:23] [Команда]**  
Используем PostgreSQL в связке с SQLAlchemy, а сервер развернём на базе Uvicorn с FastAPI, что соответствует требованиям к ОС. Это будет легко поддерживать в контейнерах Docker.

**[11:25] [Заказчик]**  
Есть ли предложения по мониторингу производительности и безопасности?

**[11:27] [Команда]**  
Мы можем интегрировать систему мониторинга, такую как Prometheus + Grafana, а также настроить алёрты для своевременного реагирования на проблемы. Логи будем централизовать через ELK-стек или аналогичные решения.

**[11:30] [Заказчик]**  
Отлично, спасибо за подробное обсуждение. Подводя итог, у нас сформирован подробный список функциональных и нефункциональных требований, который охватывает регистрацию, игровой процесс, личный кабинет, интерфейс, администрирование, обработку ошибок, дополнительные возможности, монетизацию и технические аспекты работы приложения.

**[11:32] [Команда]**  
Спасибо за встречу. Мы подготовим подробное техническое задание на основе сегодняшних обсуждений и будем держать вас в курсе дальнейших шагов. Если возникнут дополнительные вопросы или уточнения, мы свяжемся.

**[11:33] [Заказчик]**  
Отлично, жду документацию. До встречи!