Кейс по Бронированию спортивных мероприятий

Кейс ,№	Описание кейса	Цель тестирования	Виды тестирования и их обоснование	Уровни тестов (Unit, API, UI)	Примеры позитивных сценариев	Примеры негативных сценариев
1	**Регистрация нового пользователя:** Email/пароль + Соцсети (Google, FB).pp	1. Убедиться, что пользователи могут успешно и безопасно регистрироваться обоими способами. 2. Убедиться, что система корректно обрабатывает валидные/невалидные данные.	**Функциональное:** Проверка основной логики (создание аккаунта, валидация). **Интеграционное:** Проверка работы с Google/Facebook API. **Безопасности:** Проверка защиты паролей, OAuth. **Юзабилити:** Удобство формы, понятность ошибок.	**Unit, API, UI**	1. Успешная регистрация по Email/паролю. 2. Успешная регистрация через Google. 3. Успешная регистрация через Facebook. 4. Возможность войти после регистрации.	1. Регистрация с существующим email. 2. Невалидный формат email. 3. Несовпадающие пароли 4. Слишком простой/короткий пароль. 5. Пустые обязательные поля. 6. Отмена авторизации в соцсети. 7. Попытка регистрации с XSS

						в полях.
2	**Поиск мероприятий:** Фильтрация (спорт, дата, место), пагинация	1.Обеспечить корректную работу фильтров и пагинации 2.Проверить производительность поиска при большом объеме данных.	**Функциональное:** Проверка логики фильтрации и пагинации. **Производительности:** Скорость ответа при разных фильтрах и объеме данных. **Юзабилити:** Понятность и удобство фильтров и пагинации.	**API, UI**	1. Корректные результаты при фильтрации по одному/нескольким параметрам. 2. Работа кнопок пагинации (вперед, назад, номера страниц). 3. Отображение сообщения "ничего не найдено" при отсутствии результатов.	1. Некорректный формат даты в фильтре. 2. Выбор несуществующего вида спорта/местоположения. 3. Переход на несуществующую страницу пагинации. 4. Слишком долгий ответ поиска при сложных фильтрах (Performance). 5. Попытка SQL-инъекции в полях фильтра (Security).
3	**Бронирование билетов:** Выбор кол-ва,	**Гарантировать, что** 1.Система правильно отслеживает	**Функциональное:** Проверка логики выбора кол-ва и проверки остатка.	**Unit, API, UI**	1. Успешное бронирование N билетов при наличии	1. Попытка забронировать 0 билетов 2. Попытка забронировать

	проверка доступности, нет мест.	доступность билетов, 2. не позволяет забронировать больше, чем есть, 3. и корректно обрабатывает конкурентные запросы.	**Нагрузочное:** Проверка поведения при одновременном бронировании (особенно последних билетов). **Юзабилити:** Ясное отображение доступности и ошибок.		достаточного кол-ва. 2. Успешное бронирование последнего доступного билета.	больше билетов, чем доступно 3. Попытка забронировать билеты на полностью распроданное мероприятие. 4. Одновременная попытка двух пользователей забронировать последний билет.
4	**Оплата билетов:** Интеграция PayPal/Stripe, успех/неуспех/прерывание, возврат.	1. Обеспечить надежное взаимодействие с платежными системами, 2. корректную обработку всех статусов платежа и процесса возврата/отмены.	**Функциональное:** Проверка статусов оплаты и логики возврата. **Интеграционное:** Ключевое - взаимодействие с PayPal/Stripe API. **Безопасности:** Защита платежных данных. **E2E:** Весь флоу от бронирования до подтверждения оплаты/возврата.	**API, UI, Интеграционное**	1. Успешная оплата через PayPal/Stripe. 2. Корректное отображение статуса "Оплачено". 3. Успешный запрос и обработка возврата (если применимо). 3. Отмена неоплаченного бронирования.	1. Неуспешная оплата (недостаточно средств, ошибка карты, лимит банка и др). 2. Прерывание оплаты (пользователь закрыл вкладку). 3. Тайм-аут ответа от платежной системы. 3. Попытка повторного возврата. 4. Ошибка при возврате на стороне платежной системы.

5	**Промокоды и скидки:** Применение (% или фикс.), проверка срока действия.	1. Убедиться, что промокоды применяются корректно, 2. скидка рассчитывается правильно, 3. валидация (срок действия, лимиты) работает.	**Функциональное:** Проверка логики расчета скидки и валидации кодов. **Юзабилити:** Легкость применения кода, отображение скидки. **Безопасности:** Защита от подбора/злоупотребления кодами.	**Unit, API, U**	1. Успешная отправка отзыва и рейтинга в пределах лимитов. 2. Корректное отображение отзыва на странице мероприятия. 3. Отправка отзыва с максимально допустимой длиной.	1. Попытка отправить отзыв с <script> тегом (XSS). 2. Отправка отзыва, превышающего лимит длины. 3. Отправка пустого отзыва (если не разрешено). 4. Попытка отправить отзыв до посещения мероприятия (если есть проверка).
6	**Оставление отзыва:** Текст + рейтинг после мероприятия, XSS, лимит длины.	1. Позволить пользователям оставлять отзывы безопасно, 2. предотвращая XSS-атаки 3. и соблюдая ограничения по длине.	**Функциональное:** Сохранение и отображение отзыва/рейтинга. **Безопасности:** Проверка на XSS и другие вредоносные вставки. **Юзабилити:** Удобство формы отзыва.	**Unit, API, UI**	1. Успешная отправка отзыва и рейтинга в пределах лимитов. 2. Корректное отображение отзыва на странице мероприятия. 3. Отправка	1. Попытка отправить отзыв с <script> тегом (XSS). 2. Отправка отзыва, превышающего лимит длины. 3. Отправка пустого отзыва (если не разрешено).

					отзыва с максимально допустимой длиной.	4. Попытка отправить отзыв до посещения мероприятия (если есть проверка).
7	**Уведомления о мероприятиях:** Email/SMS напоминания, спам, шаблон.	Гарантировать своевременную и корректную доставку уведомлений с правильным содержанием и без попадания в спам.	**Функциональное:** Проверка триггеров отправки и генерации контента. **Интеграционное:** Взаимодействие с Email/SMS шлюзами. **Юзабилити:** Понятность и полезность уведомления. **E2E:** Проверка всего пути от события до получения уведомления.	**Unit, API, Интеграционное, E2E**	1. Уведомление отправлено за N часов/дней до события. 2. Email доставлен в папку "Входящие" (с использованием тестовых сервисов). 3. SMS доставлено. 4. Корректные данные мероприятия в шаблоне уведомления.	1. Уведомление не отправлено. 2. Уведомление отправлено с неверными данными. 3. Email попал в спам. 4. Уведомление отправлено пользователю, отменившему бронь. 4. Некорректный формат/верстка шаблона.
8	**Личный кабинет организатора:** CRUD мероприятий,	Убедиться, что организаторы могут управлять своими мероприятиями, а система ролей	**Функциональное:** Проверка создания, чтения, обновления, удаления мероприятий.	**API, UI**	1. Организатор успешно создает новое мероприятие.	1. Организатор пытается редактировать/удалить чужое мероприятие. 2. Обычный пользователь

	проверка ролей.	надежно предотвращает несанкционированный доступ.	**Безопасности/Контроля Доступа:** Проверка прав доступа (роли организатора). **Юзабилити:** Удобство интерфейса для организатора.		2. Организатор успешно редактирует свое мероприятие. 3. Организатор успешно удаляет свое мероприятие. 4. Изменения корректно отображаются пользователям.	пытается получить доступ к ЛК организатора. 3. Создание мероприятия с некорректными данными (прошедшая дата, отрицательная цена).
9	**Система отзывов о пользователях:** Метки от организаторов, валидация, защита.	Реализовать систему меток о пользователях так, чтобы она была защищена от злоупотреблений и клеветы, с корректной валидацией.	**Функциональное:** Проверка добавления/просмотра меток. **Безопасности:**Только авторизованные организаторы могут ставить метки. **Юзабилити:** Механизмы защиты от клеветы (если есть - оспаривание, модерация).	**API, UI**	1. Организатор ставит метку "не оплатил вовремя" пользователю, связанному с его мероприятием. 2. Просмотр меток (если доступно админу/другим организаторам).	1. Организатор пытается поставить метку пользователю, не связанному с его мероприятием. 2. Попытка добавить некорректную/вредоносную информацию в метку. 3. Обычный пользователь пытается поставить метку. 4. Отсутствие механизма оспаривания (если это

						требование).
10	**Отображение календаря:** Виды (месяц/неделя/день), навигация по датам.	Обеспечить корректное отображение событий в разных видах календаря и плавную, правильную навигацию между датами/периодами.	Функциональное: Проверка логики отображения событий и навигации. Юзабилити: Удобство использования календаря. Производительности: Скорость загрузки календаря с большим количеством событий.	**API, UI**	1. Корректное отображение событий в виде месяца/недели/дня. 2. Успешный переход на следующий/предыдущий месяц/неделю/день. 3. Выделение текущей даты. 4. Переход к конкретной дате через date picker (если есть).	1. Некорректное отображение событий (не на тех датах, дублирование). 2. Ошибка при переходе на прошлые/будущие периоды. 3. Медленная загрузка календаря (особенно вид месяца). 4. Визуальные баги, наложение элементов.

Кейс по разработке менеджера задач

Кейс, №	Описание Кейса	Цель тестирования	Виды тестирования и их обоснование	Уровни Тестов (Unit, API, UI)	Примеры позитивных сценариев	Примеры негативных сценариев
1	**Создание нового проекта:** Название, описание, дедлайн. Проверка границ полей.	1. Убедиться, что проекты создаются корректно, 2. валидация полей (название, описание, дедлайн) работает правильно, включая граничные значения и пустые поля.	**Функциональное:** Проверка основной логики создания и валидации. **Юзабилити:** Удобство формы создания, понятность ошибок. **Безопасности:** Проверка на XSS/инъекции в полях ввода.	**Unit, API, UI**	1. Создание проекта с минимально/максимально допустимыми данными (название, описание). 2. Создание проекта с дедлайном и без него. 3. Создание проекта с описанием и без него (если разрешено).	1. Попытка создать проект без обязательного поля (название). 2.Название/описание превышает лимит символов. 3. Некорректный формат дедлайна (напр., прошлая дата, если нельзя). 4. Ввод HTML/JS кода в поля (XSS).
2	**Добавление**	1. Гарантировать, что	**Функциональное:** Проверка	**API, UI**	1. Добавление	1. Добавление

	задачи в проект: Назначение ответственного/себя, дедлайн, без исп.	задачи корректно добавляются в проект, 2. назначаются исполнители (или остаются без них), 3. устанавливается дедлайн.	логики добавления, назначения, установки дедлайна. **Контроля Доступа:** Если есть роли, проверить, кто может добавлять/назначать задачи.		задачи с назначением на себя. 2. Добавление задачи с назначением на другого пользователя (если есть команда). 3. Добавление задачи без исполнителя. 4. Добавление задачи с дедлайном и без него.	задачи в несуществующий проект. 2. Назначение задачи на несуществующего пользователя. 3. Установка некорректного дедлайна (формат, прошлая дата). 4. Попытка добавить задачу без обязательных полей (напр., названия).
3	**Тайм-трекинг:** Старт/стоп/пауза/рестарт таймера. Корректность подсчета, грани.	1. Обеспечить точный подсчет времени, затраченного на задачу, 2. корректную работу кнопок управления таймером 3. и обработку	**Функциональное:** Проверка логики старт/стоп/пауза, суммирования времени. **Нагрузочное:** Проверка при одновременной работе нескольких таймеров (если возможно). **Юзабилити**: Понятность	**Unit, API, UI**	1. Запуск таймера, остановка, проверка записанного времени. 2. Запуск, пауза, рестарт, остановка, проверка суммарного	1. Запуск таймера для уже завершенной задачи. 2. Попытка запустить несколько таймеров одновременно

		граничных случаев (0, 24+ ч).	интерфейса таймера.		времени. 3. Корректный подсчет при работе таймера > 1 часа, > 24 часов (если поддерживается).	(если не разрешено). 3. Запись 0 секунд времени. 4. Отрицательное или нелогичное время после сбоев/ручных манипуляций (если возможно). 5. Несохранение времени при закрытии вкладки/сбое.
4	**Интеграция с Google Calendar:** Синхронизация дедлайнов, отказ в доступе.	1. Убедиться, что дедлайны задач корректно синхронизируются с Google Calendar, 2. и система адекватно реагирует на отказ пользователя в предоставлении доступа.	**Функциональное:** Проверка логики синхронизации (создание, обновление, удаление событий). **Интеграционное:** Взаимодействие с Google Calendar API. **Безопасности:** Проверка корректности OAuth флоу.	**API, UI, Интеграционное**	1. Успешное подключение к Google Calendar. 2. Дедлайн задачи появляется как событие в календаре. 3.Изменение/удаление дедлайна в Task Manager отражается в	1. Пользователь отзывает доступ к календарю. 2. Ошибка API Google Calendar (лимиты, недоступность). 3. Некорректное время/дата события в календаре.

					календаре. 4. Корректная работа после повторного предоставления доступа.	4. Дублирование событий в календаре. 5. Не удаление события из календаря при удалении задачи/дедлайна.
5	**Выставление счетов:** Генерация из часов/ставки, валюты, налоги.	1. Проверить корректность генерации счетов на основе затраченного времени и ставки, 2. правильность расчетов с учетом разных валют и налоговых ставок	**Функциональное:** Проверка алгоритмов расчета суммы, налогов. **Локализации:** Корректное отображение и расчеты для разных валют. **Юзабилити:** Понятность и гибкость формы генерации счета.	**Unit, API, UI**	1. Генерация счета для проекта с одной/несколькими задачами. 2. Корректный расчет итоговой суммы с учетом ставки и времени. 3. Применение налоговой ставки. 4. Генерация счета в разных валютах.	1. Генерация счета при нулевом затраченном времени или нулевой ставке. 2. Некорректный формат валюты/налоговой ставки. 3. Ошибка расчета при сложных условиях (много задач, разные ставки). 4. Отрицательная итоговая сумма (если возможно).

6	**Совместный доступ к проекту:** Приглашение, отправка инвайта, уровни доступа.	1. Гарантировать, что пользователи могут приглашать других в проект, 2. инвайты доставляются, 3. а уровни доступа (просмотр/полный) применяются корректно.	**Функциональное**: Проверка механики приглашения и назначения ролей. **Контроля Доступа/Безопасности:** Проверка, что права доступа действительно ограничивают/разрешают действия. **Интеграционное:** Отправка email-инвайтов.	**API, UI, Интеграционное**	1. Успешное приглашение пользователя с правами "только просмотр". 2. Успешное приглашение с "полным доступом". 3. Приглашенный пользователь видит проект/может редактировать (согласно правам). 4. Отзыв доступа у пользователя.	1. Приглашение пользователя с невалидным email. 2. Попытка пригласить уже добавленного пользователя. 3. Пользователь с "просмотром" пытается редактировать проект/задачи. 4. Недоставка email-инвайта. 5. Попытка дать права выше собственных.
7	**Уведомления о статусе задач:** Email (изменения), настройка частоты.	Обеспечить своевременную доставку email-уведомлений об изменениях статуса задач согласно настройкам пользователя	**Функциональное:** Проверка триггеров отправки и соответствия настройкам частоты. **Интеграционное:** Взаимодействие с сервисом отправки email.	**Unit, API, Интеграционное, E2E**	1. Уведомление приходит немедленно при изменении статуса (если выбрано). 2. Уведомления приходят раз в	1. Уведомления не приходят/приходят с задержкой. 2. Приходят не те уведомления или дублируются.

		(немедленно, раз в день, отключено).	**Юзабилити:** Понятность и полезность уведомлений.		день (дайджест). 3. Уведомления не приходят (если отключено). 4. Корректный текст уведомления.	3. Настройки частоты не работают. 4. Email попадает в спам. 5. Некорректные данные в тексте уведомления.
8	**Аналитика по проектам:** Отчеты (задачи, время, доход), фильтры (период, тип).	Убедиться, что система генерирует корректные отчеты по задачам, времени, доходу (если применимо) и правильно применяет фильтры по периодам и типам задач.	**Функциональное:** Проверка точности данных в отчетах и логики фильтрации. **Производительности:** Скорость генерации отчетов на больших объемах данных. **Юзабилити:** Понятность и наглядность отчетов и фильтров.	**Unit, API, UI**	1. Корректный отчет по завершенным задачам за неделю/месяц. 2. Корректный отчет по затраченному времени за период. 3. Правильное применение фильтра по типу задач. 4. Корректное отображение данных (суммы, количества).	1. Неверные данные в отчетах (неправильный подсчет задач/времени). 2. Фильтры работают некорректно (не тот период, не те типы). 3. Очень медленная генерация отчетов (Performance). 4. Ошибка при генерации отчета с пустыми

						данными.
9	**Группировка и сортировка задач:** По приоритету, дедлайну, статусу, комбинации.	Гарантировать, что задачи корректно группируются и сортируются по выбранным критериям (приоритет, дедлайн, статус) и их комбинациям.	**Функциональное:** Проверка логики сортировки и группировки. **Юзабилити:** Удобство и предсказуемость интерфейса сортировки/группировки.	**API, UI**	1. Сортировка задач по дедлайну (от раннего к позднему и наоборот). 2. Сортировка по приоритету. 3. Группировка по статусу. 4. Комбинированная сортировка (напр., сначала по статусу, потом по дедлайну).	1. Неправильный порядок после сортировки. 2. Задачи не группируются или группируются неверно. 3. Ошибка при применении нескольких критериев сортировки/группировки. 4. Визуальные баги при отображении сгруппированных/отсортированных списков.
10	**Поиск по задачам и проектам:** Ключевые слова, негативные варианты.	Обеспечить быстрый и релевантный поиск по задачам и проектам по ключевым словам в названии/описании, корректно	**Функциональное:** Проверка релевантности результатов поиска. **Производительности:** Скорость поиска при большом количестве	**API, UI**	1. Поиск находит нужную задачу/проект по слову из названия. 2. Поиск находит по слову из	1. Пустой поисковый запрос. 2. Запрос со спецсимволами (*, ?, /, <).

		обрабатывая некорректные запросы.	задач/проектов. **Негативное** тестирование: Обработка некорректных/пустых запросов.		описания. 3. Поиск нечувствителен к регистру. 4. Поиск находит по части слова (если поддерживается)	3. Очень длинный поисковый запрос. 4. Поиск не находит существующую задачу/проект. 5. Поиск выдает нерелевантные результаты. 6. Медленный поиск (Performance).