Общие вопросы

1.1. Структура технического задания на разработку программного обеспечения

**Классический подход:**

* Введение (цели и сфера применения).
* Обзор (аналогичные решения, целевая аудитория).
* Требования: функциональные и нефункциональные.
* Интерфейсы и интеграции с другими системами.
* Условия приемки проекта.

**Гибкий подход (Agile):**

* Пользовательские истории с четкими критериями выполнения.
* Эпики и задачи в инструментах управления (Jira, Notion).
* Определение минимально работоспособного продукта (MVP).

**Стандартизированный подход (ГОСТ 34):**

* Основные положения.
* Цели и задачи проекта.
* Технические условия.
* План разработки.

1.2. Распространенные ошибки в ТЗ и способы их устранения

* Неточные формулировки → Применять критерии SMART.
* Отсутствие расстановки приоритетов → Использовать метод MoSCoW (Must/Should/Could/Won’t).
* Пренебрежение нефункциональными требованиями → Учитывать безопасность, производительность и нагрузку.

1.3. Различия между ТЗ, брифом и SRS

* ТЗ — формальный документ с техническими условиями (согласно ГОСТ).
* Бриф — краткое описание концепции без детализации.
* SRS (IEEE 830) — документ, ориентированный на требования к ПО, без указания бюджета.

1.4. Стандарты для технических заданий

* ГОСТ 34 — применяется для государственных заказов в РФ.
* IEEE 830 — международный стандарт для SRS.
* ISO/IEC/IEEE 29148 — актуальный стандарт по управлению требованиями.

Формулировка требований

2.1. Примеры функциональных требований

1. Регистрация: Возможность регистрации через email или социальные сети.
2. Поиск: Фильтрация результатов по цене и рейтингу.
3. Корзина: Сохранение товаров при выходе из системы.
4. Оплата: Подключение платежных систем (PayPal, Сбербанк).
5. Администрирование: Блокировка пользователей администратором.

2.2. Примеры нефункциональных требований

* Производительность: Время отклика менее 1 секунды при нагрузке 1000 запросов в секунду.
* Безопасность: Использование HTTPS и двухфакторной аутентификации для администраторов.
* Масштабируемость: Поддержка до 10 000 одновременных пользователей.

2.3. Пользовательские сценарии

**Оформление заказа:**

* Участник: Покупатель.
* Действия: Выбор товара → Добавление в корзину → Оплата → Подтверждение.

**Сброс пароля:**

* Участник: Пользователь.
* Действия: Запрос → Получение ссылки на email → Ввод нового пароля.

2.4. Способы исключения неоднозначностей

* Шаблон: «Система обязана [действие] при [условии]».
* Глоссарий: Определение терминов (например, «Клиент» — пользователь, совершающий платежи).
* Конкретные примеры: Вместо «быстрый поиск» → «Поиск завершается менее чем за 0,5 секунды».

Детализация и спецификации

3.1. Описание архитектуры

* Диаграммы:
  + UML (Use Case, Sequence, Class).
  + BPMN (для бизнес-процессов, например, доставки).
* Технологии:
  + Фронтенд: React.js.
  + Бэкенд: Python/Django.
  + База данных: ER-диаграмма в Draw.io.

3.2. Требования к интерфейсу

* UI: Макеты в Figma (разрешения, цветовая палитра, шрифты).
* UX: Длительность анимаций не более 300 мс, микроинтеракции.
* Доступность: Соответствие стандарту WCAG 2.1 (контрастность, поддержка screen readers).

3.3. Безопасность

* Данные: Шифрование по алгоритму AES-256.
* Аутентификация: OAuth 2.0 и JWT.
* Защита: Web Application Firewall (WAF), регулярные проверки безопасности.

3.4. Интеграции

* Платежи: API ЮKassa (REST/JSON).
* Доставка: API СДЭК (SOAP).
* Аналитика: Google Analytics через Google Tag Manager.

Оценка и планирование

4.1. Оценка трудозатрат

* Декомпозиция: Разделение на задачи с оценкой времени.
* Story Points: Использование шкалы Фибоначчи в Scrum.
* COCOMO: Для крупных и сложных проектов.

4.2. Этапы и временные рамки

* Waterfall:
  + Анализ: 2 недели.
  + Разработка: 8 недель.
* Agile: Итерации по 2 недели.

4.3. Критерии приемки

* Пример: «Поиск функционирует при 100+ одновременных запросах».
* Тестирование: Покрытие кода — 90%.
* Документация: Руководство пользователя в формате PDF.

Дополнительные аспекты

5.1. Документация

* Для пользователей: PDF-инструкции и видеоматериалы.
* Техническая: Swagger, файл Readme.md.
* API: Коллекция запросов в Postman.

5.2. Тестирование

* Юнит-тесты: Jest/Pytest.
* Нагрузочное тестирование: JMeter (1000 запросов в секунду).
* Ручное тестирование: Чек-листы на 50+ сценариев.

5.3. Поддержка

* Гарантийный период: 3 месяца на исправление ошибок.
* SLA: Время ответа — до 24 часов.
* Обновления: Квартальные выпуски исправлений.

Примеры и шаблоны

6.1. ТЗ для мобильного приложения

**Проект:** «FitTracker» (трекер тренировок).  
**Требования:**

* Функции:
  + Интеграция с Apple Health.
  + Настройка индивидуальных тренировок.
* Технологии: Swift (iOS), Kotlin (Android).

6.2. Шаблон ТЗ в Markdown

markdown

Copy

Download

**# Техническое задание**

**## 1. Введение**

**\*\*Цель:\*\*** Создание CRM-системы для малого бизнеса.

**## 2. Функциональные требования**

2.1. Пользователь может оформлять заказы.

2.2. Администратор имеет доступ к статистике продаж.

**## 3. Нефункциональные требования**

3.1. Доступность системы — 99,9% (SLA).