1. Мы не просто кликаем — мы думаем

Эта презентация будет особенно полезна менеджерам, аналитикам, а возможно — и разработчикам.
Сегодня мы немного приоткроем завесу: чем частично занимается тестировщик и как устроена его работа изнутри.
Многие думают, что тестировщик — это тот, кто просто тыкает по кнопкам и ждёт, что что-то сломается.
Но на самом деле — он аналитик, исследователь и даже детектив.
Мы изучаем продукт, продумываем, как его проверить, и стараемся заглянуть на шаг вперёд — чтобы найти проблему до того, как её увидит пользователь.
Сегодня программы становятся сложнее: сайты, приложения, системы — всё связано, всё должно работать чётко.
И именно хорошо продуманное тестирование помогает сделать продукт надёжным, удобным и безопасным.
Плюс — мы экономим время и деньги: лучше найти ошибку до релиза, чем исправлять её потом.

2. Основы тест-дизайна: зачем и как?

Тест-дизайн — это не импровизация, а систематический подход. Он основан на выборе правильных сценариев и наборов данных, которые максимально эффективно выявляют ошибки и минимизируют риски при запуске.
И чтобы делать это правильно, мы не принимаем всё на веру.
Мы задаём вопросы. Уточняем у аналитиков и бизнеса: «Что здесь имеется в виду?», «А что будет в этом случае?», «Это точно должно работать так?»
Потому что качественный тест-дизайн невозможен без понимания.
Чем глубже мы разбираемся в требованиях, чем больше уточняем детали — тем точнее можем спроектировать проверку и найти то, что может пойти не так.

3. Пример из жизни: покупка билета в кино

Представим себе ситуацию: покупка билета на фильм с возрастным ограничением 16+. Этот простой, на первый взгляд, процесс содержит множество важных деталей для тестирования. Пользователь имеет возможность купить билет как онлайн, так и в кассе. Система при этом должна учитывать возрастные ограничения и подтверждение данных, обеспечивая, чтобы зрители подходящего возраста имели доступ, а несовершеннолетние — нет. Особое внимание уделяется точному подтверждению возраста, правильному выбору места, а также информированию пользователя через корректные сообщения об ошибках. Здесь могут возникнуть типичные

проблемы — обход ограничений, ошибки интерфейса, несогласованность данных, которые тестировщик обязан предусмотреть и проверить.

4. Требования к системе - это основа для тест-дизайна

Основа любого тест-дизайна — четко сформулированные требования. В данном случае, система должна гарантировать, что покупка билета без сопровождения доступна только лицам старше 16 лет. Если покупатель моложе, билет можно приобрести только в кассе при условии присутствия родителя, который предъявляет документ. При онлайн-покупке требуется подтверждение возраста через паспорт или дату рождения, причём в случае несоответствия система должна корректно отобразить сообщение "Фильм доступен с 16 лет". Это обеспечивает пользователю прозрачную обратную связь и предотвращает ошибки. Именно требования служат основой для построения сценариев тестирования.

5. Техника 1: Граничные значения

Одним из ключевых подходов является тестирование граничных значений. Возраст 15 лет — это крайний порог, при котором доступ запрещён, а 16 и 17 — уже разрешён. Такая проверка позволяет выявить типичные ошибки программирования, например, когда условие "возраст больше 16" исключает ровно 16-летних пользователей.

6. Техника 2: Эквивалентное разбиение

Эквивалентное разбиение помогает сгруппировать пользователей по категориям по возрасту и поведению системы. Например, система разделяет пользователей на три группы: 16 лет и старше — допускаются к покупке; младше 16 — не допускаются; младше 16 без сопровождения — также запрещено. Для эффективного тестирования достаточно проверить представителей каждой группы. Такая техника снижает количество тестов без снижения их качества и позволяет охватить все варианты поведения системы, предотвращая избыточное тестирование и вероятность пропуска критичных сценариев.

7. Техника 3 — Таблица решений

Таблица решений — одна из самых эффективных техник, когда необходимо проверить все возможные комбинации факторов. В случае с покупкой билетов это возраст пользователя, наличие родителя, способ покупки и подтверждение

документов. Игнорирование какой-либо комбинации может привести к пропуску ошибки. Например, ребёнок с родителем может купить билет в кассе, но не онлайн без документа. Таблица решений обеспечивает полный охват всех сценариев, делая тестирование системным и надежным, что значительно повышает качество конечного продукта.

8. Вывод: мы не просто кликаем

Тестирование — это сложный, продуманный процесс, включающий анализ разнообразных сценариев. Тестировщики применяют схемы, таблицы, логические методы, что кардинально отличает их деятельность от простого повторения действий. Это системный анализ, направленный на выявление наиболее уязвимых мест в продукте. Благодаря сочетанию технических знаний и творческого подхода, тестировщики предотвращают ошибки ещё до выхода продукта в свет, обеспечивая качество не только для разработчиков, но и для конечных пользователей.

9. Интерактив: упражнение с тестированием лифта

Чтобы лучше понять методики тест-дизайна, полезно рассмотреть пример с тестированием лифта. Основные аспекты — проверка кнопок выбора этажей, контроль нагрузки и реакции системы на различные состояния. Это отличный способ применить на практике теорию, учитывая различные технические требования и особенности взаимодействия пользователей с системой.

10. Как думает тестировщик: системный подход

Тестировщик применяет методики граничных значений и эквивалентного разбиения также и в задачах тестирования лифта. Он систематически анализирует возможные сценарии и выбирает тест-кейсы, которые позволяют обнаружить ошибки заранее. Такой подход гарантирует тщательную проверку и минимизирует риски.

11. Техника 1 для лифта: граничные значения

Пример применения граничных значений в тестировании лифта включает проверку кнопок выбора этажей, где важно убедиться в корректности работы для минимального и максимального этажей. Также критична проверка нагрузки лифта — при превышении допустимого веса система должна предупредить или заблокировать движение.

12. Техника 2 для лифта: эквивалентное разбиение

Лифт можно разделить на три основных режима работы:
🔹 Обычный режим — лифт перевозит людей по вызовам.
🔹 Аварийный режим — сработала сигнализация, отключение электричества, застревание.
🔹 Режим обслуживания — техническое ТО, обновление ПО, проверка систем.
Каждый режим — это отдельная логика поведения системы.
Поэтому их нужно тестировать по отдельности, чтобы убедиться, что:

- В обычной работе всё работает чётко.
- При аварии срабатывают датчики, связь и безопасность.
- В режиме обслуживания нельзя случайно запустить движение, но можно проводить диагностику.

Такой подход помогает выявить ошибки при переключении между состояниями — например, если лифт "забывает", что он в аварии, или позволяет подняться пассажиру во время ремонта.
Как и при разбиении пользователей на группы (например, дети / взрослые в кинотеатре), деление на режимы упрощает тестирование, но при этом сохраняет полный охват всех важных ситуаций.

13. Таблица решений: условия закрытия дверей лифта

Закрытие дверей лифта зависит от нескольких факторов — датчиков препятствий и нажатия кнопки закрытия. Система должна закрыть двери только при отсутствии препятствий и своевременном нажатии кнопки. В противном случае дверь остаётся открытой, предотвращая аварийные ситуации.

14. Итог: тестировщик мыслит глубже

В заключение стоит отметить: тестировщики — это не просто исполнители, которые прокликивают сценарии.
Мы — аналитики, которые вникают в процессы, задаём вопросы, ищем слабые места и предугадываем риски.
Именно поэтому мы не боимся уточнять у аналитиков и бизнеса: «А как должно работать здесь?», «Что считается ошибкой?», «А если пользователь сделает вот так?»
Потому что чем подробнее и яснее у нас требования — тем точнее, полнее и эффективнее мы можем проверить систему.
Глубокое понимание = глубокое тестирование.
Именно такой подход позволяет находить проблемы до релиза и делать продукт надёжным не случайно, а осознанно.