Домашнее задание Гулидовой В.В. Модуль 1.

1. Задачи системного аналитика в проекте автоматизации ZooTicket:

Этап 0. Если на проекте есть бизнес аналитик, то необходимо получить бизнес требования, если бизнес аналитик отсутствует, то необходимо собрать бизнес требования;

Этап 1. Управление требованиями:

- Выявление и сбор требований: Проведение интервью с заказчиком, администрацией зоопарка и службой поддержки. Анализ существующих процессов продажи билетов и работы с посетителями. Исследование интеграций со сторонними системами (турникеты, платежные шлюзы). Для сбора мнения конечных пользователей (посетителей) можно использовать опросы или анализ данных из соцсетей и отзывов.
- Анализ и структурирование: Анализ собранных данных на предмет противоречий, полноты и реализуемости. Перевод пожеланий stakeholders в конкретные функциональные и нефункциональные требования. Приоритизация (например, по MoSCoW).
- Документирование: Формализация требований в виде:
    - Пользовательских историй (User Stories) и сценариев использования (Use Cases) для функций (например, "Как Посетитель, я хочу купить билет онлайн, чтобы не стоять в очереди в кассу").
    - Бизнес-правил (например, "Детский билет предоставляется посетителям до 12 лет").
    - Требований к интеграциям (формат данных для обмена с турникетом).
    - Спецификации требований к ПО (SRS).

Этап 2. Моделирование и проектирование.

1) Процессное моделирование: Создание BPMN-диаграмм для ключевых бизнес-процессов (например, "Процесс покупки билета", "Процесс возврата билета", "Процесс контроля доступа через турникет").
2) Проектирование архитектуры и данных: Разработка UML-диаграмм (например, диаграмма вариантов использования, диаграмма последовательностей для сложных сценариев, концептуальная модель данных).
3) Прототипирование интерфейсов: Создание wireframes или макетов ключевых экранов (главная страница, процесс оплаты, личный кабинет администратора). Задание логики и структуры для UI/UX-дизайнера.

Этап 3. Коммуникация в процессе разработки ПО.

- Передача требований команде разработки: Участие в планировании спринтов, постановка задач разработчикам (frontend, backend) и тестировщикам (QA), оформление задач в Jira/Confluence.

- Ответы на вопросы: Постоянная поддержка команды разработки в ходе спринта, разъяснение требований и разрешение неоднозначностей.

- Валидация и приемка: Участие в приемо-сдаточном тестировании (UAT), проверка реализованного функционала на соответствие изначальным требованиям. Помощь QA в составлении чек-листов, особенно для сложных бизнес-сценариев.

2. Взаимодействие CA с участниками проекта:

- Заказчик / Product Owner -Определение бизнес-целей, видения продукта, приоритезация функциональности, согласование ключевых требований и критериев приемки.
- Пользователи (Администраторы, Кассиры, Поддержка) - Сбор информации о текущих процессах, проблемах и потребностях. Валидация предлагаемых решений и прототипов.
- Конечные пользователи (Посетители) - Косвенное взаимодействие через опросы и анализ данных для понимания их ожиданий от онлайн-покупки.
- Команда разработки (Backend, Frontend, QA) - Передача и разъяснение требований, ответы на вопросы в ходе спринта, совместное проектирование технических решений, уточнение критериев приемки.
- Tech Lead / Архитектор - Согласование технической реализуемости требований, проектирование архитектуры системы, обсуждение API и интеграций.
- UI/UX-дизайнер - Передача требований к пользовательским интерфейсам, согласование wireframes и макетов на соответствие бизнес-логике.
- Project Manager / Scrum Master - Участие в планировании спринтов, оценка сроков по задачам, информирование о рисках и блокерах, связанных с требованиями.
- Внешние подрядчики (интеграции) - согласование форматов данных и протоколов взаимодействия для интеграции с системами оплаты, турникетами и т.д.

3. Жизненный цикл проекта: Обоснование выбора

Рекомендуемый жизненный цикл: Agile (гибридная модель на основе Scrum с элементами Kanban)

Обоснование выбора:

- Неопределенность и изменчивость требований: В начале проекта сложно предугадать все нюансы, которые захотят увидеть заказчик и пользователи. В процессе работы обязательно появятся новые идеи или потребуется корректировка уже согласованных решений. Agile идеально подходит для таких условий, так как позволяет гибко вносить изменения в бэклог продукта в конце каждого спринта.

- Необходимость быстрого получения ценности: Заказчик (зоопарк) заинтересован в том, чтобы начать продавать билеты онлайн как можно скорее, даже с базовым функционалом. Итеративная разработка короткими спринтами (2-3 недели) позволяет показывать работающий продукт после каждого спринта, получать обратную связь и постепенно наращивать функциональность (например, сначала простая покупка билета, затем личный кабинет, затем интеграция с турникетами).

- Важность постоянной коммуникации: Проект затрагивает несколько разных групп пользователей. Agile-подходы, такие как ежедневные стендапы и регулярные демонстрации, обеспечивают постоянную синхронизацию между командой разработки, аналитиком и заказчиком, минимизируя риски недопонимания.

- Снижение рисков: Короткие циклы разработки позволяют быстро обнаружить, если команда движется в неправильном направлении, и скорректировать курс. Это значительно менее болезненно и затратно, чем осознать ошибку в конце большого Waterfall-проекта.

Почему не Waterfall (каскадная модель)?
Waterfall предполагает, что все требования известны, четко зафиксированы и не будут меняться до конца проекта. Для проекта автоматизации, где много скрытых нюансов и ожиданий от разных стейкхолдеров, это почти невыполнимое условие.
Любое изменение на поздних этапах (например, во время тестирования) в Waterfall ведет к значительным переделкам и сдвигам сроков.
Заказчик увидит работающий продукт только в самом конце, что является высоким риском несоответствия его ожиданиям.