Тест-план проекта Astrofy (https://github.com/TheLonelyAstronaut/Astrofy)

1. **Объект тестирования**

   Astrofy – интернет-платформа по продаже электроники и различных видов техники, включая в себя серверную часть, кроссплатформенное мобильное приложение и панель администратора.

   Мобильное приложение реализовано с помощью фреймворка React Native, который позволяет писать кроссплатформенные приложения на разные операционные системы с возможностью инджекции нативного кода каждой платформы. Приложение включает в себя бизнес-логику и декларативное описание интерфейса (TypeScript), и систему оплаты (Kotlin/Swift).

   Сервер реализован с помощью фреймворка Apollo Server, который дает возможность использовать так называемый язык запросов GraphQL, который служит заменой традиционному REST. Включает в себя такие сервисы, как Amazon S3 и Heroku Postgres.

   Панель администратора реализована с помощью фреймворка с ReactJS и Material UI с инджектом типов данных из TypeScript прямиком в JavaScript благодаря специально-разработанной библиотеке трансформации для TypeScript и WebPack, что позволяет гибко и быстро проектировать интерфейсы.

   Система должна обладать быстротой и гибкостью для моментального развертывания на любых устройствах и возможности быстрого изменения существующих реализаций в процессе дальнейшей разработки.

   Функциональные требования не были выполнены полностью в связи с добавлением дополнительного функционала, неописанного в SRS. Например, из-за реализации панели администратора было затрачено меньше ресурсов на мобильное приложение и возможность загрузки фото пользователя не была реализована на стороне клиента, хотя такая возможность на стороне сервера есть.

   Нефункциональные требования выполнены частично, реализовано все, кроме покрытия кода unit тестированием в связи с расширением функционала, который отсутствует в SRS и требует значительного количества ресурсов.

2. **Риски**

   В связи с использованием сторонних сервисов (Amazon, Heroku, Firebase), работоспособность системы не инкапсулируется в границах работоспособности хоста, и качество предоставляемых услуг зависит по большей части от внешних сервисов, потому что вся информация о товарах платформы хранится именно на них.

3. **Аспекты тестирования**

   Для данной системы будет проведен ряд тестов, которые будут включать в себя тестирование следующих частей приложения: авторизация, работа с товарами, процесс оплаты.

4. **Подходы к тестированию**

   Так как основная логика работы с товарами находится на сервере, то в качестве тестирования сервера будут использоваться клиенты и среда тестирования GraphQL, которую можно использовать после запуска сервера. Тестирование клиентов будет проходить в ручном режиме.

5. **Представление результатов**

   Результаты будут представлены в виде таблицы, содержащей в себе описание теста и его результат. Тест можно считать пройденным только в случае полного совпадения реального результата с ожидаемым, иначе тест считать проваленным. Будут протестированы вход, регистрация, процесс оплаты, загрузка товаров и их поиск.

6. **Выводы**

   Вышеописанные требования позволяют протестировать работу системы и убедиться в ее стабильности, что поможет разработке дальнейших планов по развитию системы и ее выводу на рынок, так как при сбое того или иного компонента системы можно сделать очередную итерацию разработки и тестирования данного кейса в соответствии с уже одобренными компонентами системы, которые прошли тесты и используются в модуле, содержащем ошибку.