AGODA Project Test Plan

1. Test Scenarios
2. Test Cases Creation
3. Testing Coverage:
    a. Manual API Testing (Using Postman)
    b. REST API Automation (Using rest-assured)
    c. Performance Testing (Using Gatling)
    d. Security testing (OWASP Security)
4. Environment Creation
5. CI/CD Integration

1.Test Cases Scenarios {Smoke/Sanity/Regression/Performance/Security}:

- Validate the keys with the Min. and Max range of APIs (e.g. maximum and minimum length)
- Verify Keys verification. If we have JSON, XML APIs we should verify it's that all the keys are coming.
- Verify XML, JSON Schema validation.
- Verify the Parse the Response data
- Verify the JSON Schema validation, Verify the Field Type, Verify the Mandatory Fields
- Verify Response headers & Negative Test cases response
- Verify that how the APIs error codes handled.
- Verify the response HTTP status code.
- Verify Response payload
- Verify Chaining Request verification.
- Verify of APIs with Data parameters.
- Verify End to End CRUD flows
- Verify Database Integrity Test Cases
- Verify File Upload Test cases

NOTE: - Segregations of tags , smoke/sanity etc. is done at test case level which is attached in google sheet.

2. Test Case Creation:

S.No	API Endpoint	Location (Drive link)	Start Date	ETA	Review	Comments/Blockers
01	Fetch Inventory	Test Cases				
02	Update Inventory	Test Cases				

3.Test Coverage:

a) API Automation

Summary:

The REST Automation Framework will focus mainly on the 2 given Modules/APIs. The strategy to achieve this milestone is to use rest-assured library in a BDD framework which will have the following integrations:

Language	Java
Framework	TestNG, Cucumber (BDD Approach)
Reporting	Extent Reports
Build Tool	Maven
VCS	Gerrit
CI/CD	Jenkins
Plugins	Slack Integration

Active Challenges:

1. Framework Readiness
2. Gerrit Integration
3. Cucumber feature file creation
4. Jenkins Integration
5. Plugins Integration

b) Performance Testing:

The Load Testing will be done using Gatling Framework and Scala as programming language. Product need to give an expected number of users on Platform. Accordingly, threshold of servers will be analysed.

Action towards Load testing:

S.No.	Task	Start Date	ETA	Comments/Blockers
01	Defining individual APIs			
02	Different User Journey			
03	Simulations Scenarios			
04	Reporting Analysis			

Sources:

Name	Type	Location
Agoda-project	Gerrit	<project branch>
AWS EC2	EC2 Instance	ec2-user@<ip> ec2-user@<ip>
User Stories	JIRA	<JIRA story ID>

c) Security Testing:

The action plan is to achieve API Security Testing for Agoda project APIs via using OWASP ZAP. It is an open-source web application security scanner which can be used for Web.

API Scanning/Attack can be done via two ways:-

a. Automated Scan - Click on quick start & enter Api name on which we want to attack, zap will scan the api's via spider and active scan start attacking on the explored api's.

b. Manual Explore

For manual explore/scan first off all we need to configure proxy in a desired browser i.e Firefox → Settings → Network → Http proxy (Localhost) & port 8080 [port number should be same as in zap]

Prerequisite

- **Setup**

Download ZAP - https://www.zaproxy.org/download/

Zap Overview - OWASP ZAP – Getting Started

Active Challenges/Needs to work

- Jenkins Integration [Add on's in existing ones]
- Form Authentication with zap owasp
- Discuss Report with dev in detailed view

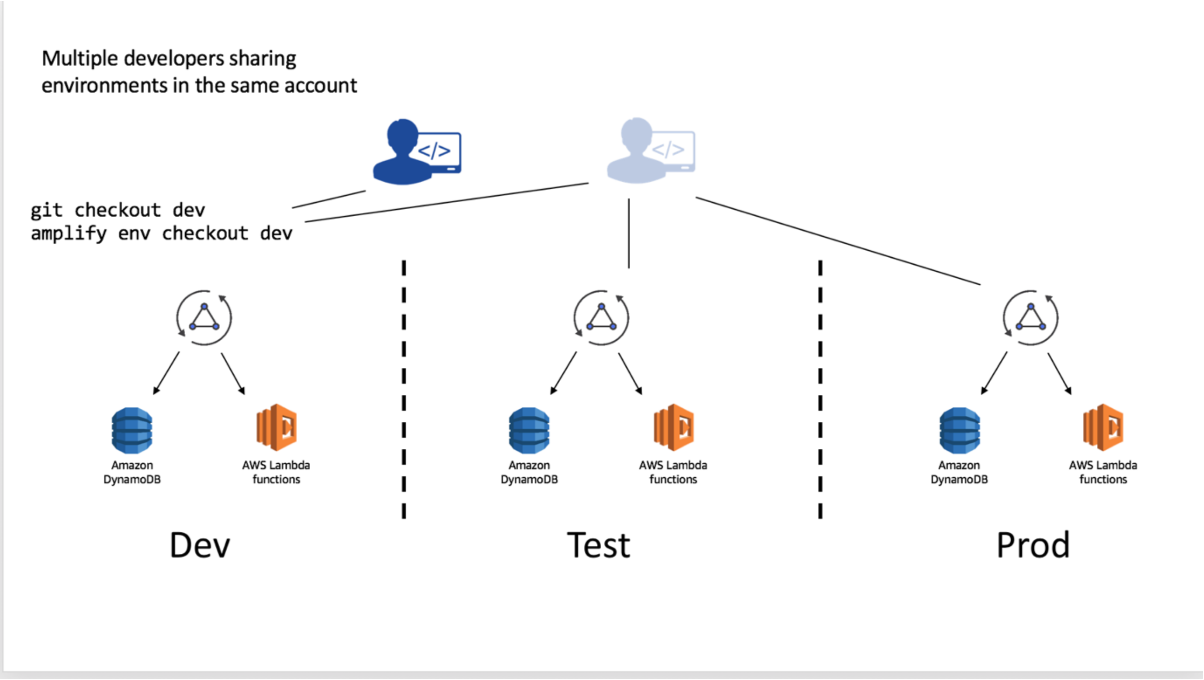
4. Test Environment Setup:

There will be 3 different environments: Test, Pre-prod, Prod

According to me, all environment has same components as follows:

1. Creation of Test Bed
2. Setup AWS (EC2 Clusters), Tomcat servers
3. Setup Docker
4. Setup Kubernetes console
5. Setup DB replicas
6. Third party integration, if any (Amazon SQS, Kafka, Redis, etc)
7. Integrate Jenkins Pipeline

Test Components Setup:



Multiple developers sharing environments in the same account

```
git checkout dev
amplify env checkout dev
```

5.CI/CD implementation strategy:

- What-test-suite at each feature branch, required pass before PR merge
  Solution: **Unit and Smoke Test Suite**. Webhooks will be integrated with the VCS so that whenever dev commits any changes in the feature branch, both test suites job will be triggered. We can mark the commit pass if the threshold of Unit and Smoke test Suite is ~100%. Now, The PR will be approved.

- What-test-suite at default branch, triggered after every new change merged.
  Solution: **Sanity Test Suite.** If job pass percentage is more than 85% then changes will be deployed in the AWS EC2 instance.

- What-test-suite at default branch, on a schedule basis
  Solution: **Regression Test Suite.**

- What-test-suite at default branch, required pass before building release package
  Solution: **E2E Test Suite which includes Regression (API + UI), Performance, Security Test Suite**.