Test Plan

Testing Instagram App

# Test Methodology

This document briefly describes the testing plan and methodology for testing Instagram application

## Test Categories

Below are the different categories of testing that will be covered while testing the AUT

* Smoke and Sanity
* Functional Testing
* Regression Testing
* Performance Testing
* Exploratory Testing
* Integration Testing
* Security/Penetration Testing

## Test Case Creation

### Smoke And Sanity Tests

1. Open the App and check for crash
2. Open the App and check for error logs
3. Open the App and check for elements/objects

### Functional Tests

1. Try to login to app with invalid credentials
2. Try to login to app with only username
3. Try to register in the app using invalid email

### Regression Tests

1. Try to login to app with existing user credentials
2. Try to open an existing public users account using search
3. Try to open an existing private users account using search

### Integration Tests

1. Login to the app, search for an user, follow the user, like one of their post and log out from the app
2. Login to the app, upload a new image, leave a comment on the same post and logout from the app
3. Login to the app, upload a new image, delete the uploaded post and logout from the app

### Performance Tests

1. Create a load for 1000 users trying to login to the app at the same time
2. Create a load for 1000 users trying to register into the app at the same time
3. Create a load for 1000 users trying to view and interact on same post in the app at the same time

## Dependencies

Below are the information required to start testing.

* Login Credentials
* Assumptions(if any)
* Initial conditions(if any)
* Preloaded App data(if any)
* Platform specific APK/APP file(for stable performance)

## Defect Reporting

Assuming JIRA is used for defect reporting,

Defects will be raised on the development JIRA project.

Below Information is required to create JIRA ticket

* Title – Title for the ticket(prefix app version if necessary)
* Description – Please provide Version Information, Steps for replication, Actual Results, Expected Result, Screenshot/Video clips(if available) and any other notes/logs to assist dev team
* Affected Version – Specify the version affected
* Type of Ticket – Bug/New Feature/Question
* Component – App/ Functionality Component that is under escalation
* Label – To categorize or filter tickets for dashboard and monitoring purposes
* Assignee – User/Dev to whom the ticket should be assigned to
* Priority – based on how crucial the ticket is

## Tooling

Belo are the tools and technologies used for Testing

* JIRA – to flag defects and store features and requirements
* ALM/TestRail
* IntelliJ IDEA
* Appium
* Emulators(iOS and Android) and Real Devices
* Selenide
* Cucumber

## Test Management

Tests can be managed using ALM / TestRail for the long run.

# Practical Implementation

## Test Automation

Functional Tests are picked for Automation using Appium with Selenide and Cucumber

## Reporting

Cucumber reports are being used for Automation reporting. After every run this report will be created and will be deleted after a day automatically.

# CI/CD Pipeline

We can implement CICD using any of the popular tools like Team City/Jenkins/Bamboo/CircleCI

Create an YAML for the pipeline with below necessary informations

* Variables – to set URL, environment, credentials etc
* To Run Tests – Specify the Image(pbc), tasks(scripts to be executed), final-tasks, artifacts to be extracted
* Triggers – What should trigger the pipeline, it could be any other deployment/build or fixed time
* Branching Info – Branch overrides info, creation, deletion info, jira links

### Execution

Whenever the trigger activates, it will create the image, use branching info and start running the tasks provided

### Result

In case of Success, publish the artifact via mail/slack

In case of failure, publish the log and failure info via mail/slack to needed users