Date / /
Page No.

Assignment 4

Ques 1 Test case design - online payment Gateway

Task

1) Design test cases for the payment input from using:

→ Boundary value Analysis

→ Equivalence class Testing

Scenario :- payment gateway handles multiple payment modes, currency conversion and fraud detection.

Assumption (state these on your copy)

→ Amount field accepts INR equivalent values from 1 to 1,00,000.

→ card number : 16 digits, cvv : 3 digits, Expiry : mm/yy, currency list includes - INR, USD, &

→ supported payment method : Credit card (cc), Debit card (DC), UPI, wall

(A) Boundary value Analysis (BVA). selection field

→ value to test : 0 (below min), 1 (min), 2 (min+1) 999 (max-1) 10001 (above max)

Expected 0 → project

1, 2, 99999, 100000 → Accept

10001 → Project

Date / /
Page No.

cardNumber (length = 16 digits)
→ values: 15 digits (below), 16 (valid), 17 (digits) (above))
→ Expected: 15 → Reject, 16 → Accept, 17 → Reject

cvv (3 digits)
value: 12 (2digits), 112 (3 digits), 1234 (4digits)
expected: 12 → Rejected, 123 → Accept.
1234 → Reject

B Equivalence class Testing

Amount:
→ valid class: 1 — 100000 → test 5000
→ Invalid class: = 0 (eg. 0) > 100000 (e.g. 1000)

card Number
→ valid: 16 digit number → 4111222 33334444
→ Invalid: Non-numeric, length ≠ 16

currency:
→ valid class: supported currencies (INR USD, EUR - Test USD)
→ Invalid:- Non-numeric

Date / /
Page No.

(ii) Decision table for handling different payment method and possible errors

condition/Rules	R_1	R_2	R_3	R_4	R_5
Payment method = CC/DC	Y	Y	Y	N	N
payment method = UPI	N	N	Y	Y	N
Balance/limit = OK	Y	N	Y	Y	Y
Card valid	~~N~~	Y	—	-	-
currency conversion required	N	N	Y	N	N
~~Fraud~~ = Suspect	N	Y	N	N	~~N~~
Action		Process reject convert Ask/app Hold Notify Process retry manual			

Ques 2 Structural testing - Library management system

(1) Explain path testing and design a simple control flow graph for the the burrowing process

Selecting independent execution path through code to ensure every path is tes (based path testing).

Aim: cover all desion outcomes and loops at least once

Date / /
Page No.

Simple control flow graph

Start
↓
check Book Availability - NO → return "Not available" → End
↓ Yes
↓
check user Borrow limit → Exceeded → Return Limit reached
↓ within limit
↓
check user fine ? Has fine → Return "pay fine" End
↓ No fine
↓
update Inventory → IssueBook → Set DueDate → log → Transaction → End

⇒ Independent Path to test
1 Book not available (Start → Not available → end
2 Available, but user limit exceeded
3 Available, limit ok, user has fines → blocked
4 Available, limit no fines → successful issue

(ii) Describe data flow testing for variable involved in the borrowing logic

Date / /
Page No.

⇒ Important variables: BookID, User, Borrow count, Due Date, inventory count.

→ Data flow concerns & tests

Definations:-

use pairs:- Ensure inventory count defined when checking availability update after issue

Test: book availability (count > 0) then ensure count decrements

User after define :- borrow count defined on login profile ⇒ used to check limit: Test Borrow count: Borrow count Boundary (limit-1)

Ques3 Integration and system Testing - ~~Ride-Hailing~~ app

i) outline an integration testing strategy for the app's components

(ii) Define alpha and beta testing and how they would be used in the app released cycle

(iii) Recommend popular testing tools for functional and non-functional testing of the app

Date / /
Page No.

(i) Approach :- Start with core service (user Registration, Ride Booking)

→ Integrate GPS tracking next cuss drives /stubs for b/w)

→ Integrate payment last (Internal gateway)

API :- Between mobile app ⇒ backend : validate request schemes.

use service virtualization :- for external dependence (map, payment, gateway)

End-to-end smoke :- After every major merge new user sign up → book ride → GPS update

Regression suits :- for critical flows (sign-up booking)

(ii) Alpha vs Beta Testing

Alpha Testing : Internal testing by developme QA in a controlled environment used to find major defects before wider release for the app run alpha with company testers

Beta testing : release to limited external real use (early adopters) in the real environment to collect usability feedback & find environment specific for the app run public beta in one try collect crash reports ux feedback.

Date / /
Page No.

iii) Recommended testing tools
→ functional (UI & API)
→ mobile device testing
→ Non functional
→ security & fuzzing

Q4 Software maintenance - Legacy Banking Software

(i) Describe the maintenance management process you would establish to handle ongoing changes.

→ Intake & Triage
→ change Request (CR)
→ Impact Analysis
→ Estimate & Plane
→ Development & Per Review
→ Release & Rollback plan
→ Continuous Documentation
→ Regression & Accepting Testing

(ii) Explain reverse engineering and software re-engining approaches to improve the system.

→ Reverse engineering :- Analyse the existing system to extract design architecture data model & create documentation.

Date / /
Page No.

⇒ Software Re-engineering :- modify and restructure the system to improve maintainability without changing extract behaviour.
Step - reverse engineering - ~~reptac~~ refactor modules ⇒ replace legecy tech

(iii) configuration management & Documentation (role)

configuration :- use vcs with branching strategy. Tag maintain development scripts cm ensure reproducible build & controlled rollbacks

Documentations : maintain architure docs API specs, rumback & change logs, update Inline code comments & a central knowledge base, good docs, reduce onboarding time & prevent regression during maintance