**SELECTING TEST STRATEGY**

**(QUESTION 1)**

* Cho một đoạn giới thiệu một dự án đang được triển khai và các tính năng của nó
* Cung cấp các thông tin bên lề như thời gian phát triển, độ lớn team dev, độ lớn team test
* Có các ràng buột hệ thống
* Yêu cầu chọn test strategy, giải thích lí do chọn và cho biết cách ứng dụng vào việc test dự án

Most Suitable Testing Strategies:

Reasons for Choosing These Strategies:

Implementation Approach:

Yêu cầu của dự án:

Có tài liệu rõ ràng không? → Nếu có, chọn Analytical hoặc Methodical.

Có tiêu chuẩn cụ thể không? → Nếu có, chọn Process-/Standard-compliant.

Tính chất của ứng dụng:

Phức tạp, nhiều trạng thái? → Chọn Model-based.

Đơn giản, cần kiểm tra nhanh? → Chọn Reactive (Dynamic).

Sự tham gia của các bên liên quan:

Khách hàng có ý kiến rõ ràng về ưu tiên không? → Chọn Directed (Consultative).

Tần suất thay đổi:

Có nhiều cập nhật không? → Chọn Regression-averse.

Thời gian và nguồn lực:

Hạn chế thời gian? → Chọn Reactive.

Đủ thời gian để phân tích? → Chọn Analytical hoặc Model-based.

**Project Requirements:**

**- Are there clear documents?**

**If yes → Choose: Analytical or Methodical** **Explanation:**

* **Analytical**: Focuses on in-depth understanding through analysis, suitable when there are detailed documents.
* **Methodical**: Step-by-step, structured approach that follows a planned sequence. Good when documentation is solid and well-organized.

**How to implement:**

* Start with a **requirement analysis** phase.
* Use **UML diagrams, user stories, and wireframes** to model requirements.
* Conduct **design reviews** and **checklists** to ensure each step aligns with documentation.

**Standards & Compliance:**

**- Are there specific standards?**

**If yes → Choose: Process-/Standard-compliant** **Explanation:**

* Emphasizes **following industry standards or internal processes** (e.g., ISO, CMMI, coding standards, agile processes).

**How to implement:**

* Adopt a **development methodology** (e.g., Scrum or Kanban).
* Use **code linters**, **CI/CD pipelines**, and **peer reviews** to maintain code quality.
* Regularly audit for compliance against the selected standard.

**Nature of the Application:**

**- Complex, many states?**

**If yes → Choose: Model-based** **Explanation:**

* Suited for systems with many interactions or states (e.g., real-time apps, workflows).
* Emphasizes **using models** to simulate and validate system behavior.

**How to implement:**

* Use **state diagrams**, **sequence diagrams**, or **finite state machines**.
* Apply **MVC architecture** to separate models from views and controllers.
* Test logic with **unit tests** focused on different states.

**- Simple, needs quick checks?**

**If yes → Choose: Reactive (Dynamic)** **Explanation:**

* Suited for **fast-changing, user-facing or frontend-heavy apps**.
* Focus is on quick feedback loops, frequent testing, and fast fixes.

**How to implement:**

* Use **hot-reloading**, **component testing** (e.g., with React, Vue).
* Integrate **CI tools** for instant feedback.
* Prioritize **end-user feedback** in short iteration cycles.

**Stakeholder Involvement:**

**- Does the customer have clear priorities?**

**If yes → Choose: Directed (Consultative)** **Explanation:**

* The strategy centers around **stakeholder priorities**, allowing frequent feedback and direction changes.
* Best for projects with a **clear product owner or decision maker**.

**How to implement:**

* Schedule **regular sprint reviews or demo sessions** with clients.
* Use **backlog grooming sessions** to reassess priorities.
* Maintain a **visible priority list** using a Kanban board or backlog system.

**Frequency of Change:**

**- Frequent updates?**

**If yes → Choose: Regression-averse** **Explanation:**

* Focuses on ensuring **existing functionality is not broken** by new changes.
* Emphasizes **regression testing and automation**.

**✅ How to implement:**

* Build a **comprehensive test suite** (unit, integration, regression).
* Automate tests in CI/CD workflows.
* Use tools like **TestNG, JUnit, Selenium**, etc., depending on the tech stack.

**Time and Resources:**

**- Limited time?**

**If yes → Choose: Reactive** **Explanation:**

* Fast-moving, iterative, **prioritizes delivering quickly**.
* Often embraces a **“fail fast”** mindset.

**How to implement:**

* Use **Agile or Lean methods**.
* Deliver **Minimum Viable Product (MVP)** first.
* Prioritize **high-value features** and use **time-boxed sprints**.

**- Enough time to analyze?**

**If yes → Choose: Analytical or Model-based** **Explanation:**

* Encourages **planning and modeling upfront**.
* More time is spent on **understanding requirements and design** before coding.

**How to implement:**

* Allocate a **dedicated discovery/design phase**.
* Do **risk analysis**, **user research**, and **prototype testing**.

Involve team in **design discussions** to ensure shared understanding.

**CODE REVIEW AND BUG IDENTICATION**

**(QUESTION 2)**

* Cho 1 hàm có đầu vào đầu ra, khoảng 10-20 dòng code, lấy theo chuẩn code java
* Thường hàm này có mảng (có vòng for, tràn mảng, đảo các giá trị, tính trên các giá trị)
* Hãy tìm các "bug" trong hàm này, thường là 5-6 bugs, ghi số dòng xảy ra bug, giải thích ngắn gọn bug tìm thấy
* *VD:*

*| Stt | Dòng | Bug description*

*1 15 lack of variable declaration*

*2 1 wrong coding convention of function name, must be camelcase*

**THƯỜNG CÓ NHỮNG BUG SAU**

1. **GIỐNG LỀ CODE, DẤU CÁCH TRONG BIỂU THỨC**

*VD:*

*if (yob < 1990)*

*sout(...) -> INCORRECT, WRONG IDENTATION*

*if (yob < 1990)*

*sout(...) -> CORRECT*

*int a= 5; -> INCORRECT, NEED A SPACE OF SEPERATION OF EXPRESSION ELEMENT*

*int a =5; -> INCORRECT, NEED A SPACE OF SEPERATION OF EXPRESSION ELEMENT*

*int a = 5; -> CORRECT*

*int a = b+c -> INCORRECT, NEED A SPACE OF SEPERATION OF EXPRESSION ELEMENT*

1. **KHAI BÁO BIẾN KHÔNG TUÂN THEO CODING CONVENTION**
2. **BIẾN: NOUN, camel Case Notation**

*VD:*

*int A -> INCORRECT, WRONG VARIABLE NAMING CONVENTION*

*double Salary -> INCORRECT, WRONG VARIABLE NAMING CONVENTION*

*double caculateSalary -> INCORRECT, VARIABLE MUST BE NOUN*

*double basicSalary -> CORRECT*

1. **HÀM: VERB + NOUN, camel Case Notation**

*VD:*

*GetArea() -> INCORRECT, WRONG FUNCTION NAMING CONVENTION*

*getArea() -> CORRECT*

*payment() -> INCORRECT, FUNCTION MUST START WITH VERB*

*pay() -> CORRECT*

*payAnOrder() -> CORRECT*

1. **CLASS: NOUN, Pascal Case/Upper Camel Case Notation**

*VD:*

*public class homeController() {*

*} -> INCORRECT, WRONG CLASS NAMING CONVENTION*

*public class GetHomeController() {*

*} -> INCORRECT, CLASS MUST BE NOUN*

*public class HomeController() {*

*} -> CORRECT*

1. **DẤU {...}**

*VD:*

*boolean pay() {*

*} -> CORRECT*

*boolean pay()*

*{*

*} -> INCORRECT, WRONG ANGLE BRACKETS CONVENTION*

1. **THIẾU ACCESS MODIFIER/ACCESSOR (public, private, default, protected) KHI THIẾT KẾ CLASS**

*VD:*

*public class X {*

*private ... dành cho field*

*public ... dành cho hàm*

*} -> CORRECT*

*public class X {*

*int yob ...*

*public ...*

*} -> INCORRECT, LACK OF ACCESS MODIFIER, SHOULD ADD PRIVATE WHEN DECLARING FIELDS*

*public class X {*

*public double salary;*

*public ...*

*} -> INCORRECT, VIOLATION OF ENCAPSULATION PRINCIPLE*

1. **KHAI BÁO BIẾN CHƯA GÁN VALUE MÀ DÙNG TRONG BIỂU THỨC**

*VD:*

*int a;*

*sout(a) -> INCORRECT, MISSING VARIABLE DECLARATION*

*int a;*

*int b = a; -> INCORRECT, MISSING VARIABLE DECLARATION*

*int b = 10;*

*int c = b; -> CORRECT*

1. **CHÚ Ý VÒNG FOR**

*VD:*

*for(; ;) {*

*sout(...)*

*} -> INCORRECT, WRONG LOGIC*

*for(); {*

*} -> CORRECT (TRONG 1 SỐ TÌNH HUỐNG)*

*for(int a = 1; a <= 10, a++);*

*sout("hello") -> CORRECT*

*if(); -> CORRECT (TRONG 1 SỐ TÌNH HUỐNG)*

1. **SAI VỀ LOGIC XỬ LÍ (LÀM SAU CÙNG)**

**COMPONENT TEST CASE DESIGN**

**(QUESTION 3)**

* Viết các test case để đảm bảo các lệnh trong hàm được test ít nhất 1 lần
* Lệnh trong hàm: lệnh thường, lệnh rẽ nhánh, lệnh lặp
* Thường đề bài sẽ có hàm có code có nhiều lệnh if else, yêu cầu thiết kế test case
* Đảm bảo: mọi lệnh được chạy ít nhất 1 lần - statement coverage
* Đảm bảo: mọi lệnh rẽ nhánh được chạy ít nhất 1 lần - branch coverage
* *VD: Kiểm tra đủ tuổi vào quán bar, rạp phim (18+)*

*public boolean check18Plus(int age) {*

*if (age < 18) //ST1 //BR1*

*return false; //ST2*

*return true; //ST3*

*}*

Đề thi yêu cầu thiết kế các test case để 100% statement coverage, 100% branch coverage

* **KỊCH BẢN NỘP BÀI 1:**

TEST CASE #1: check18Plus(15); CHỈ LỆNH ST1, ST2 ĐƯỢC CHẠY ST1 + ST2 / 3 LỆNH = 2/3 = 66.6666%

* **KỊCH BẢN NỘP BÀI 2:**

TEST CASE #1: check18Plus(15); CHỈ LỆNH ST1, ST2 ĐƯỢC CHẠY

TEST CASE #2: check18Plus(18); CHỈ LỆNH ST1, ST3 ĐƯỢC CHẠY

100% STATEMENT COVERAGE

* **KỊCH BẢN NỘP BÀI 3:**

TEST CASE #1: check18Plus(15); BR1 ĐƯỢC CHẠY

100% BRANCH COVERAGE

* **KỊCH BẢN NỘP BÀI 4:**

TEST CASE #1: check18Plus(18); BR1 ĐƯỢC CHẠY

100% BRANCH COVERAGE

* **FORMAT NỘP BÀI:**

TEST CASE ID: 1

TEST DESCRIPTION: CHECK AGE UNDER 18

TEST STEPS: GIVEN AGE = 15

INVOKE/CALL THE METHOD check18Plus(15)

EXPECTED VALUE: FALSE

TEST CASE ID: 2

TEST DESCRIPTION: CHECK AGE OVER OR EQUAL 18

TEST STEPS: GIVEN AGE = 18

INVOKE/CALL THE METHOD check18Plus(18)

EXPECTED VALUE: TRUE

**SYSTEM TEST CASE DESIGN**

**(QUESTION 4)**

* Đề cấp cho 2 business processes với các tình huống và nhiều lựa chọn khác nhau, viết test case cho các tình huống
* *VD:*

*Think about the normal flow, alternative flow, and possible exceptions that may occur in each business process. Write at least 5 test cases for each business process, ensuring you cover:*

* ***Normal Flow****: The expected process under typical conditions.*
* ***Alternative Flow****: Variations where users take different actions but still complete the process.*
* ***Exception Flow****: Error cases where something goes wrong (e.g., invalid input, missing data, system failures).*

*For each test case, follow the standard format:*

1. ***Test Case ID:*** *TC-XX*
2. ***Test Scenario:*** *Test … with normal/alternative/exception flow*
3. ***Preconditions*** *(if applicable)****:*** *User want to …*
4. ***Test Steps:*** 
   1. *User làm gì*
   2. *User làm gì*

*…*

***5. Expected Result:*** *… success*