# Exercise: Test Techniques I

Problems for exercises and homework for the "QA Fundamentals an Manual Testing" course @ Software University.

## 1. Equivalence Partitioning / Boundary Value Analysis – Age Checker

Now that you are familiar with the Equivalence Partitioning / Boundary Value Analysis Techniques, let's recall **The Age Checker App** from the QA Basics course, that works as follows:

• If age is less than 13, returns "child"

• If age between 13 (inclusively) and 20 (exclusively), returns "teenager"

• If age between 20 (inclusively) and 65 (exclusively), returns "adult"

• If age is equal or bigger than 65, returns "elder"

• If the age is negative or above 150, returns "error"

**Your task is:**

**Equivalence Partitioning:** Divide the possible input values of the "age" into different equivalence classes or partitions. Remember to include both valid and invalid partitions.

**Boundary Value Analysis:** Identify the boundary values of the defined partitions and come up with test cases that include these boundary values. Ensure you consider "edge cases" - values just outside of valid ranges.

**Note:** Keep in mind that testing should cover not only expected or valid inputs but also unexpected or invalid ones. Consider all possible scenarios that might be encountered in a real-world situation.

## 2. Equivalence Partitioning / Boundary Value Analysis

**This is an exercises from the ISTQB textbook "Fundamentals of Software Testing ISTQB Certification"**

**Scenario:** If you take the train before 9:30 am or in the afternoon after 4:00 pm until 7:30 pm ('the rush hour'), you must pay full fare. A saver ticket is available for trains between 9:30 am and 4:00 pm and after 7:30 pm.

What are the partitions and boundary values to test the train times for ticket types? Which are valid partitions and which are invalid partitions? What are the boundary values? (A table may be helpful to organize your partitions and boundaries.) Derive test cases for the partitions and boundaries.

Are there any questions you have about this 'requirement'? Is anything unclear?

## 3. Simplest Decision Table

**Scenario:** Your local fitness center offers Yoga classes. The classes are open to both members and non-members. However, the booking process is different based on these factors:

If the participant is a member, they can book the class online.

If the participant is a non-member, they need to call the center to book their spot.

Based on these rules, your task is to create a decision table. This table should reflect all the possible scenarios and the corresponding results.

**Consider these conditions:**

Participant wants to book Yoga class (True/False)

---

Participant is a member (True/False)

**And this action:**

Booking method (Online/Phone Call)

# 4. Decision Table

**This is an exercises from the ISTQB textbook "Fundamentals of Software Testing ISTQB Certification"**

**Scenario:** If you hold an 'over 60s' railcard, you get a 34% discount on whatever ticket you buy. If you are traveling with a child (under 16), you can get a 50% discount on any ticket if you hold a family railcard. Otherwise, you get a 10% discount. You can only have one type of railcard.

Produce a decision table showing all the combinations of fare types and resulting discounts and derive test cases from the decision table.

# 5. State Transition

**This is an exercises from the ISTQB textbook "Fundamentals of Software Testing ISTQB Certification"**

**Scenario**: A website shopping basket starts out as empty. As purchases are selected, they are added to the shopping basket. Items can also be removed from the shopping basket. When the customer decides to check out, a summary of the items and the total cost are shown to him to say whether it is OK. If the contents and price are OK, the customer leaves the summary display and goes to the payment system. Otherwise, they go back to shopping (so you can remove items if you want).

**a.** Produce a state diagram showing the different states and transitions. Define a test in terms of the sequence of states to cover all transitions.

**b.** Produce a state table. Give an example test for an invalid transition.

# 6. Statement and Decision Testing

**This is an exercises from the ISTQB textbook "Fundamentals of Software Testing ISTQB Certification"**

**Scenario**: A vending machine dispenses either hot or cold drinks. If you choose a hot drink (e.g., tea or coffee), it asks if you want milk (and adds milk if required), then it asks if you want sugar (and adds sugar if required), then your drink is dispensed.

**a.** Draw a control flow diagram for this example. (Hint: regard the selection of the type of drink as one statement.)

**b.** Given the following tests, what is the statement coverage achieved? What is the decision coverage achieved?

**Test 1**: Cold drink

**Test 2**: Hot drink with milk and sugar

**c.** What additional tests would be needed to achieve 100% statement coverage? What additional tests would be needed to achieve 100% decision coverage?

# 7. Pairwise Testing

Assume you have a function of a software application for testing. The **function contains** the following fields (input values, that can be accepted by the fields, also are given below):

1. List box that contains 10 elements (input values – 0, 1, 2, 3, 4, 5, 6, 7, 8, 9);

2. Radio button (input values – On or Off);

4. Checkbox (input values - Checked or Unchecked);

5. OK button (input values - Does not accept any value, only redirects to the next page).

**Your task is:**

1. Calculate how many would be the possitive test cases, if you have to cover every single possibility?

2. Using Pairwise testing tool of your choosing, reduce the number of necessary test cases.

3. These was only positive test cases. How about negative ones? Write at least 3 negative test cases.

# 8. Use Case Testing / Online Movie Ticket Booking System

**This is an exercises from the ISTQB textbook "Fundamentals of Software Testing ISTQB Certification"**

**Use Case**: Book a movie ticket

**Actor:** User

**Precondition:** The user has a registered account and is logged in; The movie showtime is available.

**Steps:**

The user searches for a movie;

Selects the desired movie from the search results;

Chooses the preferred showtime;

Selects the desired seat;

Proceeds to checkout;

Selects a payment method and provides payment information;

Confirms the booking;

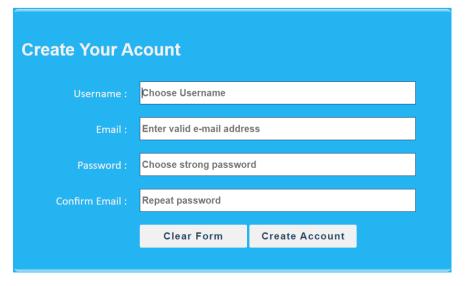The system processes the booking and sends a booking confirmation to the user.

**Postcondition:** The selected seat is marked as booked for the chosen showtime. The user receives a booking confirmation email. The booking appears in the user's booking history.

**Task:** Derive the test cases from the 'Book a movie ticket' use case.

# 9. Registration Form

Remember this registration form from QA Basics Form?

http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com/registration-form-bugs/

### Create Your Acount

| | |
|---|---|
| Username : | Choose Username |
| Email : | Enter valid e-mail address |
| Password : | Choose strong password |
| Confirm Email : | Repeat password |

Clear Form      Create Account

---

Follow us:

Since you are familiar with many testing techniques, your task is to decide which techniques can be applied to the Login Form testing and write the test cases.

Keep in mind the following **field requirements**:

- **Username** consists of **no less than 6 letters AND numbers**.
- E-mail should be valid e.g., something **/at/** something **/dot/** something
- **Password** should be **minimum 6 characters** and **maximum 12 characters**. **Uppers case, lower case, number,** and **special character required.**
- **Confirm password** should **match password.**
- **Clear Form should clear all fields.**
- **Create account redirects** to another page and **"You are now logged in." message** should **appear.**