# Homework: Software Quality Assurance Introduction

## Think Testing: Gas Station

A woman goes to a gas station and fills fuel in her car. She pays and tries to start the car, but **the car fails to start**. List possible **reasons** that you can think of.

|  |  |
| --- | --- |
| **Problem #1** | **Car is broken/mechanical problem** |
| **Problem #2** | Key is lost |
| **Problem #3** | Woman put wrong fuel |
| **Problem #4** | Key is broken |
| **Problem #5** | Woman got in the wrong car |
| **Problem #6** | Car was stolen |

## Think Testing: Tooth Brushing

It is time to teach your 6-year-old child to brush its teeth alone. It needs a clear **step-by-step instruction**, so list the steps and be as detailed as you can.

|  |  |
| --- | --- |
| **Step #1** | **Turn on the water** |
| **Step #2** | **Take the toothpaste** |
| **Step #3** | Open the toothpaste |
| **Step #4** | Take the toothbrush |
| **Step #5** | Put a little paste on to the brush |
| **Step #6** | Open your mouth |
| **Step #7** | Put the brush in your mouth |
| **Step #8** | Start moving the brush and cleaning the teeth from left/right side |
| **Step #9** | Move the brush to the other side |
| **Step #10** | Take the brush out of your mouth |
| **Step #11** | Clean the brush |
| **Step #12** | Clean your mouth and teeth |
| **Step #13** | Wash your hands |
| **Step #14** | Turn off the water |

## Think Testing: 5 Kg Bag

How would you **test a 5 kg capacity grocery shopping paper bag**? Describe the tests that you could perform.

|  |  |
| --- | --- |
| **Test #1** | **Put 1kg products in the bag and check if it will not tear** |
| **Test #2** | **Put 2kg products in the bag and check if it will not tear** |
| **Test #3** | **Put 5kg products in the bag and check if it will not tear** |
| **Test #4** | **Put 5.1kg products in the bag and check if it will tear** |

## Login Form UX Problems

As QA engineers, you should be trained to **find issues in the UI and UX** of the apps under test. Given an e-commerce web site "**My Wonderful Shop**", find the **problems** in the below **login form**:

Graphical user interface

Description automatically generated

|  |  |
| --- | --- |
| **Problem #1** | Wrong website name, it is "**My Wonderful Shop**", but link is “your-wonderful-shop” |
| **Problem #2** | Wrong link name, it is log in form, but link is “add-to-basket” |
| **Problem #3** | Password is before username |
| **Problem #4** | Not aligned buttons |
| **Problem #5** | Not needed Log out button, as it is log in form |

## Weather Forecast Bug

Your software company works on a project, which displays the **weather forecast** for the next few days. Developers take the weather forecast from **external source**, where the forecast data is given in computer-readable format. The received **weather forecast data** holds weather information for the next 5 days. For each day the following data is obtained: weather icon (sunny / cloudy / light rain / heavy rain / snow / fog / etc.), the min and max temperature, precipitation percentage and wind strength.

During the testing you, as a QA engineer, find out that the **temperatures** come from the external source in **degrees Fahrenheit**, but are displayed in **degrees Celsius without a conversion**. This obviously produces wrong results, such as extremely hot temperatures (like 78 °C). Describe the problem in detail in the table below:

|  |  |
| --- | --- |
| **Mistake** | The developer made the following mistake: did not create a correct functionality that converts Fahrenheit to Celsius |
| **Bug (location)** | The bug in the code should be in the module / function, responsible for: wrong function for showing the correct temperature |
| **Failure (symptoms)** | When the buggy code goes in production, it fails as follows: wrong temperature shown |

## Age Checking Machine

As a QA engineer you often will be assigned to **check if certain business logic is correct**. Your company develops a special machine (hardware + software), designed to be put at the entrance of a **bar**, where people come for a drink. The machine should **check the age** of people when they try to enter and tell them, if they are **allowed to walk in**.

You are assigned to **test the machine**, which reads the personal ID card, extracts the **age of the person** from it (integer number), and displays appropriate **message** to each visitor, **based on the age**. The machine **opens the door** when the person is allowed to enter the bar.

By design the machine should implement the **following logic**:

|  |
| --- |
| **Check visitor’s age and print appropriate message** |
| 1. If **age > 0**, and **age < 18**, then **print** "*You are too young to visit our bar*". The door stays closed. 2. If **age > 18**, then **print** "*Welcome to our bar. Enjoy!*" and the door opens. 3. **Otherwise**, **print** "*Invalid age. Please try again*". |

Do you find **any issues** in the above logic?

|  |
| --- |
| The logic should include checking including 0 as age and leave the doors closed.  The next thing which is missed is to include 18 as age and open the door if the customer is 18. |