# Objectives

* + Automate as many test cases as possible to improve the quality and timeliness of deployments from the IT perspective
  + Reduce cost by automating test cases
  + Reduce time to implement allowing Business to enhance the system quickly to allow for new product concept categorizing as needed
  + Improve quality on high priority changes and testing

# Goals

# Constraints

# Assumptions

# High Level Requirements

# Detailed Requirements

# Standards

## Guideline Test Techniques

### specification attributes checklist

#### Complete

##### Is anything missing or forgotten?

##### Is it thorought?

##### Does it include everything necessary to make it stand alone?

#### Accurate

##### is the proposed solution correct?

##### Does the properly define the goal?

##### Are there any errors?

#### Clear

##### is the description exanct and not vague?

##### is the a single interpretations?

##### is is easy to read and understand?

#### Consistent

##### is the description of the feature written so that it doesn't conflict with itself or other items in the specification?

#### Relevant

##### is the statement necessary to specify the feature?

##### is it extra information that should be left out?

##### is the feature traceable to an original customer need?

#### Feasible

##### can the feature be implemented with the available personned tool, and resources within the specified budget and schedule?

#### Code free

##### does the specification stick with defining the product and not the underlying software design, architecture, and code?

#### testable

##### can the feature be test?

##### is enough information provided that a tester could create tests to verify its operation?

### specification terminology checklist

#### always, every, all, none. Never

##### make sure it is

##### think as a tester of cases that voilate them

#### certainly, therefore, clearly, obviously, evidently

##### clarify

##### these words ten to persuade you into accepting something as a give

##### don't fall into the trap

#### some, sometimes, often, usually, ordinarily, customerily, most, mostly

##### these workds are too vague

##### it's impossible to test a feature that operation sometimes

#### etc., and so forth, and so on, such as

##### when specs finish with words like these they are not testable

#### good, fast, cheap, efficient, small, stable

##### these are unquantifiable

##### they are not testable

#### handled, processed, rejected, skipped, eliminated

##### these terms can hide large amounts of functionality that need to be specified

#### if then but missing else

##### what will happen if the if does not happen

### test to pass then test to fail

#### fail ideas

##### code to capture old defects

##### run in low memory

##### run against upper and lower limits

##### test using hot keys and special characters

##### test int by usine long or double

##### test str short by using short long

##### default, empty, blank, numm, zero, and nont

##### known incorrect value

##### turn off power

##### shut down in the middle of a process

##### comment out connections in the source code, recompile, then test

##### ask to see the metrics that state the edit is used the most

# Formal Reviews

## essential elements

### identify problems

#### find problem with the software

#### find missing items

#### data declarations

#### computation errors

#### comparision errors

#### control flow errors

#### subroutine parameter errors

#### input and output errors

### other

#### will the software work with all languages

#### is the software intended to be compiles by other compilers

#### any conflicts between software and all possible hardware and operating systems

#### any warnings being seen

#### perform manual reality checks

### follow rules

#### provide an agenda

### prepare

#### provide and agenda with code to be reviewed and roles defined

### write a report

#### make all findings available to the entire team

# Standards

## should goto statements be used

## how can web service calls be located for everyone to leverage

## All TC projects should only have a main, remaining methods need to be located in another package to be leveraged by all

# Principal attributes of tools and automation

## Speed

## Efficiency

## Accuracy and precision

## Resource reduction

## Simulation and emulation

## relentlessness

# Configuration Testing

# Compatibility Testing

# Software Security Testing

# Feasibility

# Approach for automating governance

## Run similar to the access database

## Or create a RunTestCase java file for each test case that creates a log with the test case and date on it

## Or create one new RunTestCase that repeats the entire loop for each test case

## Or setup RunTestCase with a parameter then loop on a text file of test cases to run

# Approach for converting test case process from access to java

## Create technical requirements for each test case

## Maintain one DFD

## Update FEDE BOM Source

## Create public variables for controlling whether testing is focus

## Install code to avoid launching user interface

## Rewrite code to facilitate this process

### Currently the MS Access app wants the TC entered and the environment then wants the user to execute each step so it can record all logs to establish a baseline

### The JOptionPanel can be used to gather the test case number and test for the existence of a baseline

### Java can do this if the user first tries to pick from a list of actions but has the opportunity to enter a new action and for each of these cycles, the cancel button will be renamed to completed, a text file is written with the action and an access database is launched or Alteryx workflow is launch to populate the logs for the baseline. The user pick list will launch an Alteryx flow to create a text file to read into the pick list or JOptionPanel.

### If the baseline exists, the JOptionPanel can read each step found in the baseline prompting the user to click Yes when done. It can also display test with color for the health of each step as well as pass or fail.

I do not know what to provide for status.

Do you know the answer to any of the following questions?

* What should be status consist of?
* What are the detailed tasks we are being asked to do?
* What high level tasks are those detailed tasks mapped to?
* How are the high level tasks related to the goals and objectives for this activity?
* Do the assumptions include no documentation?

Here are my assumption:

Status

* Design - Fails
  + Test cases written are not coded to call other dependent test cases
* Code - Fails
  + All variables are not descriptive
* Test Case Execution - Passes
  + The test cases did execute from Eclipse without error
* Test Case Results Analysis - Fails
  + As data changes on the test cases, the test cases fail based on the below
    - When Engineering Commodity is blank, the test case runs
      * The Engineering Commodity should be a dependent test case
    - When there are no records left, the test case does not go to the next project or add a new record
* Documentation Fails
  + Code has no documentation of CDSID’s that have touched the code
  + SharePoint has no documention on the research for any of these test cases
* Assuming we supposed to report based on the spreadsheet everyone is working to has
  + 10 test cases showing completed under
    - Feasibility of Automation = Web Service Automation
    - Function = Edit Part
    - Category not equal to PPM
  + 3 test case source code has been promoted

Edit Part RunTestCases

package com.ford.fede.testcases.testing;

import java.io.IOException;

import java.io.InputStream;

import java.util.Arrays;

import java.util.List;

import java.util.Properties;

import org.apache.log4j.Logger;

import com.ford.fede.testcases.helper.UserHelper;

import com.ford.fede.testcases.util.CommonConstants;

import com.ford.fede.testcases.util.TestResultsExcelWrite;

import com.ford.pd.bom.security.outbound.layer.SecurityReturnCode;

public class RunTestCases {

private static Logger logger = Logger.getLogger(RunTestCases.class);

protected static String user\_id = null;

public static void main(final String[] args) throws Exception {

final Properties properties = getProperties();

final String mode = properties.getProperty("FEDEBOM.mode");

final String env = properties.getProperty("FEDEBOM.env");

final String wslCookie = properties.getProperty("FEDEBOM.wslCookie");

final String retrieveEndpointurl = properties.getProperty(mode + "." + env + ".endpointUrl");

final String userId = System.getProperty("user.name").toUpperCase();

logger.info("env"+env);

boolean isValidUser = false;

List<String> roles;

String selectedRole;

TestResultsExcelWrite resultsWriter = new TestResultsExcelWrite();

if (validateProperties(mode, env, wslCookie, retrieveEndpointurl)) {

isValidUser = UserHelper.login(mode, env, retrieveEndpointurl, wslCookie, userId);

if (!isValidUser) {

logger.error(SecurityReturnCode.UserDoesNotExistInDB);

} else {

roles = UserHelper.retrieveValidEmpolymentTypes(mode, env, retrieveEndpointurl, wslCookie, userId);

selectedRole = UserHelper.verifySelectedRole(roles);

if (!selectedRole.equals(CommonConstants.NO\_ROLE\_FOUND)) {

logger.info("Running Test Cases: " + Arrays.toString(args));

TC15936 tc15936 = new TC15936();

tc15936.absoluteCostChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC15937 tc15937 = new TC15937();

tc15937.deltaCostChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC15938 tc15938 = new TC15938();

tc15938.absoluteWeightChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC15939 tc15939 = new TC15939();

tc15939.deltaWeightChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC15940 tc15940 = new TC15940();

tc15940.absoluteToolingCostChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC15941 tc15941 = new TC15941();

tc15941.deltaToolingCostChange(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId, roles);

TC16428 tc16428 = new TC16428();

tc16428.roundtripCAD(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16429 tc16429 = new TC16429();

tc16429.roundtripCost(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16430 tc16430 = new TC16430();

tc16430.roundtripManufaturing(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16431 tc16431 = new TC16431();

tc16431.roundtripMaterialAndCoating(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16432 tc16432 = new TC16432();

tc16432.roundtripPartClassification(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16433 tc16433 = new TC16433();

tc16433.roundtripPreProductionTooling(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16434 tc16434 = new TC16434();

tc16434.roundtripProductionTooling(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16436 tc16436 = new TC16436();

tc16436.roundtripSourcing(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16439 tc16439 = new TC16439();

tc16439.roundtripWeight(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16761 tc16761 = new TC16761();

tc16761.matrialAndCoatingAttributesValidValues(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16762 tc16762 = new TC16762();

tc16762.manufacturingAttributesValidValues(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC16763 tc16763 = new TC16763();

tc16763.CostAttributesValidValues(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,roles);

TC18999 tc18999 = new TC18999();

tc18999.historyOfCADAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19000 tc19000 = new TC19000();

tc19000.historyOfCostAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19001 tc19001 = new TC19001();

tc19001.historyOfManufacturingAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19002 tc19002 = new TC19002();

tc19002.historyOfMaterialAndCoatingAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19003 tc19003 = new TC19003();

tc19003.historyOfPartClassificationAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19006 tc19006 = new TC19006();

tc19006.historyOfServiceAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19007 tc19007 = new TC19007();

tc19007.historyOfSourcingAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19008 tc19008 = new TC19008();

tc19008.historyOfTorqueAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19010 tc19010 = new TC19010();

tc19010.historyOfWeightAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

TC19011 tc19011 = new TC19011();

tc19011.historyOfWERSAttributes(mode, env, retrieveEndpointurl, wslCookie, resultsWriter, userId,selectedRole);

logger.info("Running Test Cases Completed");

} else {

logger.error(CommonConstants.NO\_ROLE\_FOUND);

}

}

}

else {

if (mode == null || mode.isEmpty()) {

logger.error("Mode cannot be empty in application.properties");

}

if (env == null || env.isEmpty()) {

logger.error("Env cannot be empty in application.properties");

}

if (wslCookie == null || wslCookie.isEmpty()) {

logger.error("Cookie cannot be empty in application.properties");

}

if (retrieveEndpointurl == null || retrieveEndpointurl.isEmpty()) {

logger.error("End point url cannot be empty in application.properties");

}

}

}

private static boolean validateProperties(final String mode, final String env, final String wslCookie,

final String retrieveEndpointurl) {

return (mode != null && !mode.isEmpty()) && (env != null && !env.isEmpty())

&& (wslCookie != null && !wslCookie.isEmpty())

&& (retrieveEndpointurl != null && !retrieveEndpointurl.isEmpty());

}

private static Properties getProperties() {

final Properties properties = new Properties();

InputStream input = null;

try {

input = Thread.currentThread().getContextClassLoader().getResourceAsStream("application.properties");

properties.load(input);

logger.info("application.properties loaded");

} catch (final IOException e) {

logger.error("Error: reading application.properties failed.");

logger.error(e.getMessage());

} finally {

if (input != null)

try {

input.close();

} catch (final IOException e) {

logger.error(e.getMessage());

}

}

return properties;

}

}