Hochschule Fulda
University of Applied Sciences

Test-Oriented Software Development

Unit Testing he Registration Module

# of the Fulda-Stadt System

Lloyd M. Dzokoto

Matrikel-Nr: 246985

Winter Semester 2016, Fulda

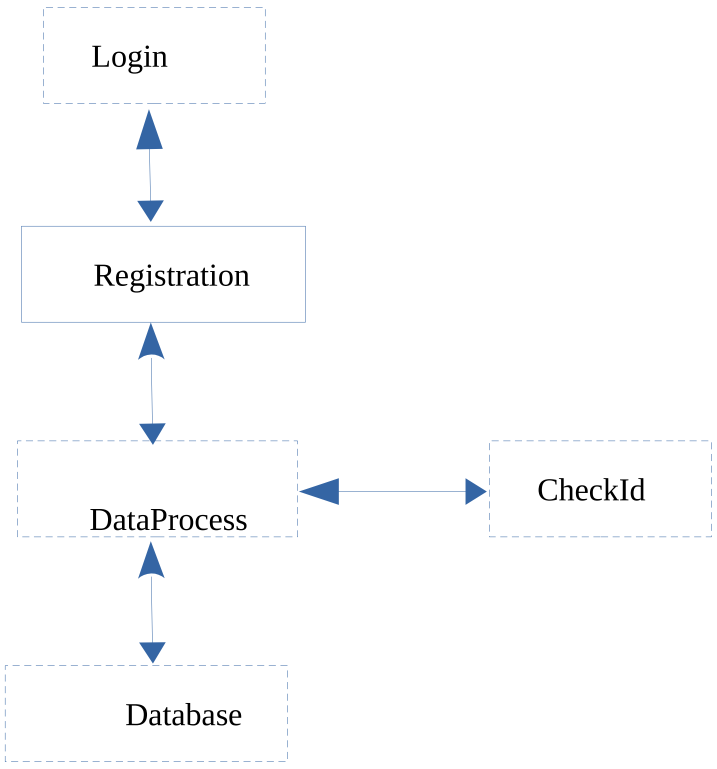
Table of Contents:

- **Introduction**
- **System Architecture**
- **Unit Requirements**
- **Test Strategy**
- **Peer Review**
- **Lab Report**

Introduction

The objective of this test strategy for the Fulda-Stadt System (FSS) software, is to define the test scope, test levels with its associated test types ,pass /fail criteria, and risk analysis.

System Architecture:



Unit Requirements:

The Registration Module of the Fulda-Stadt System would be considered for a Unit Testing. The primary objective of the module is to allow new registrants to register.

Requirement Analysis:

Unit Users (Actors): Registrants (a user who wants to use the module).

After a successful login of a registrant, the System shall:

- load the registration form.
- provide a text-field for the registrant to enter a valid full name.
- provide a text-field for the registrant to enter a valid Address code.
- provide a text-field for the registrant to enter a valid Passport Identification.
- provide a submit button for the registrant to submit his/her data.
- provide a cancel button for the registrant to discontinue registration.

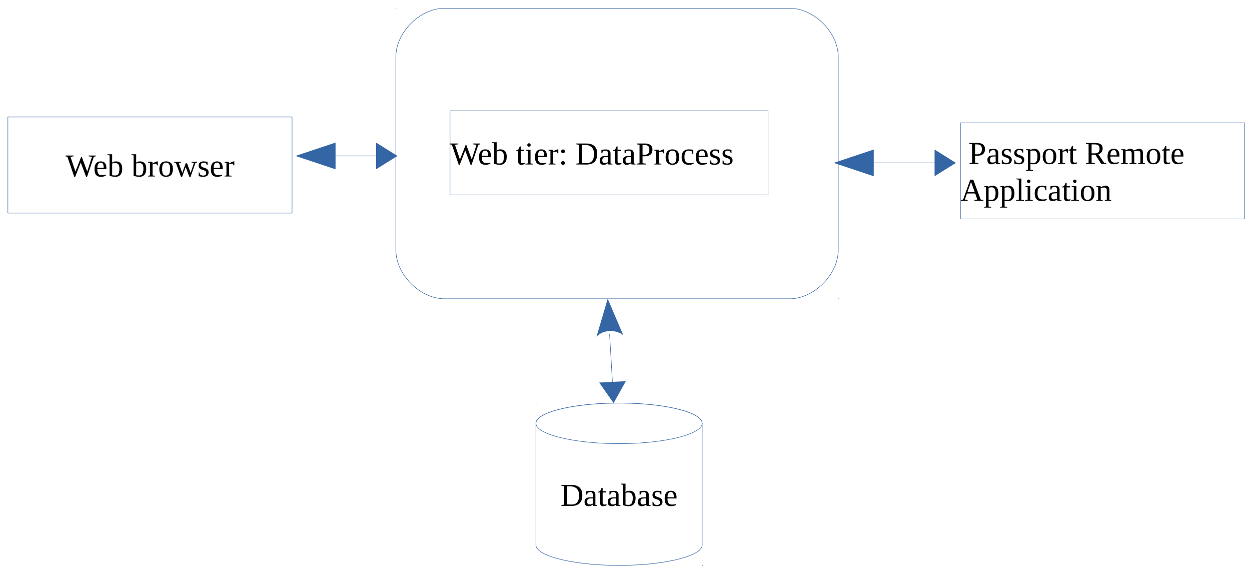
After Cancel button is clicked,the System shall:

- close registration process.

After Submit button is clicked,the System shall:

- submit the form data for further processing.

Functional Specification:



Unit System Requirements:

• JavaScript

• HTML5

• CSS3

• Google Chrome, Mozilla Firefox, Safari (latest and second-to-latest versions)

Unit Design Details:

The Registration Module would be achieved with the JavaScript function:

```
function checkForm()
{
    //check for valid fullname and empty string
   //check for valid addresscode for Fulda
   //check for right format for passport identification
}
```

Test Strategy:

The objective of this document is to design a testing strategy for the Registration Module of Fulda-Stadt System (FSS) .
The test will execute and verify the test scripts, quality criteria, and define   all high and medium severity defects per the entrance criteria, prioritize lower severity defects for future fixing for all test levels according to the V-Model.

Scope:

- Automated JavaScript Unit Test
- Static Review Test

Deliverables:

- Static Review
- Test Report

To be Tested:

- Component Test

Not to be Tested:

- Integration Test
- System Test
- User Acceptance Test

Test Types:

- Functional Testing
- Non-functional Testing

Test Levels:

Component Test:

Objective :The Registration Module would be tested in isolation to find defects and confirm the module is function as specified.

Integration Test:

Objective : A Component Integration Test would be performed to detect defects in the interactions between the Registration and DataProcess Modules.

Assumption: All modules are assumed to be developed and tested.

System Test:

Objective: A System Testing would be performed to determine whether the individual modules put together meets the specification requirements.

Assumption: All modules are assumed to be developed and tested.

User Acceptance Test:

Objective: A User Acceptance Test to confirm the system meets the expectations of user.

	Component Test	*Integration Test*	*System Test*	*User Acceptance Test*
Tester(s)	1. Development Team	1. Development Team 2. Independent Testers	1. Independent Test Team 2. End Users	Application Users Consultants
Test Basis	Component requirements and code.	Software detailed design	System requirements, functional specifications and risk analysis report	1.User and System requirements. 2.User expectations

Test Object(s)	checkForm method	Registration Module & ProcessData Module ProcessData Module & Data Storage	User manuals Database	Business processes on integrated system
Test Environment	Latest version of Google Chrome running on Ubuntu 16.0.4 or above	Networked workstations running Ubuntu 16.0.4	Production-like environment	Production environment
Test Strategy	Test Driven Development	1.Top- Down 2.Incremental	Test Cases	Alpha Testing and Beta Testing
Test Design Method	Black box technique Syntax Test and Equivalence Class Partition	White box technique Statement Testing and Coverage	White box technique: Decision Testing and Coverage:	White box technique.
Test End Criteria	75% test cases coverage	80% of branch coverage	100% passed test cases	Complete Alpha and Beta testing
Method Justification	reduce test cases using representative value	Complete statement coverage is required.	Testing combinations of logical expressions for each textfield.	Receive feedbacks from users Independent testers.
Test Tool	Test framework (JavaScript Unit Test)	stubs, drivers and debuggers	Commercial and open-source testing tools	Commercial and open-source testing tools

Non-Functional Testing

	Component Test	*Integration Test*	*System Test*	*User Acceptance Test*
Functionality	Can a new user register?			
Reliability	Is the form submitted with valid data?			
Security		Is data communication secured?	Is the system safe from cyber attacks?	Is system secured from hackers?
Usability		Are error messages generated in red colors?	Is the system usuable?	Are end users able to use the system without much difficulty?
Maintainability	Is code well commented?			Can new functionalities be added?
Portability	Is Module able to run on different Operating System		Is the system working well on other Operating systems?	
Efficiency		Does it require complicated hardware and software settings	Is the system machine-resource efficient?	Is the system running slow on user's machine
Robustness	How to handle invalid data			Are user's given prompts for invalid data.
Compatibility	Can module be used with different	Whether Modules are able to		

	browsers?	communicate.		
Performance	How long does it take to load page?	Can the system sustain 50 users at peak hours.	Can the system sustain 100 users at peak hours.	
Reliability	Is the Module producing the right outcome.	Are the Modules working together to produce the right outcome?		Is registrations successfully

Risk Analysis:

Risk Id #	*Test Level*	*Risk Scenario*	*Probability*	*Impact*	*Mitigation*
R01	Component Test	Developer is reluctant to perform unit testing	high	high	Entire Development team is responsible for unit testing.
R02		Unit Test is browser and platform specific dependent	medium	high	Makes changes to module to be platform and browser independent
R03		A non-experienced tester is asked to develop test cases	medium	high	Provide test case design guideline documents
R04	Integration Test	Communication challenges between Development Team and Test	high	high	Team Leads must have a meeting with their members to resolve this.

		Team			
R05		Late modifications are made to a module without testing	medium	medium	Perform a regression test on module before integration test.
R06	System Test	There aren’t enough software licenses for setting up the test environment. Process of acquiring a license takes about two weeks.	medium	high	Create test environment with available license and begin testing.
R07		Inappropriate test design method for test case development	medium	high	New test cases would be designed using appropriate method
R08	User Acceptance Test	User insists on making changes to requirement document before using software	medium	high	User should sign off the current development and sign a new contract to modify requirement document
R09		User has not got	medium	medium	Contract

		the required resources to test system			independent testers

Peer Review Checklist - Instruction

Test Object: Registration Module, Lloyd M. Dzokoto, Hochschule Fulda, 28.03.2017

Goal:

- Improve quality
- Cost reduction by early defect detection

Review schedule: 21.11.2016, 18:00 – 19:00, G111

Moderator: Galindo Bello Manases Jesus

List of reviewers:

Role	Person	Scribe	Time (h) spent for preparation	remarks
System Architect(Maintainability, Design, code quality etc)	Ramanpreet Kaur	Ramanpreet Kaur	45	Code is quite hard to understand without enough comments.
Business Analyst – Required Functionality	Touhidur Rahman		30	Code could add email functionality

(Verification and validation)				
Critical Paths, Code completeness and functioning.	Aleksandr Anfilov		30	It is more efficient to have separate functions to handle each textfield.
Java Programming Expert	Intesar Haider		30	Standard coding guidelines were observed.

Kick-off Meeting: 28.03.2017, 11:00-13:00, Hochschule Fulda Linux Labor

Before the review starts:

	Yes	No	
Code runs without compiler warnings?	X		
Reviewers are well prepared?	X		
Reference documents available?	X		Fuctional Design Document
Scribe is named?	X		

After the review:

	Yes	No	
Is the list of review findings available?	X		
Time spent for preparation filled in above?	10mins		
Result agreed by reviewers?	X		

Result: Accepted with changes

Priority Levels: 1- High, 2-Medium, 3- Low

List of findings:

No.	Location	Raised by	Priority	fixed	remarks
1.	Function checkForm	Ramanpreet Kaur	2	X	Author agrees to fix this
2.	Function checkForm	Touhidur Rahman	3		Manager would find out from Customer if it is needed.
3.	Function checkForm	Aleksandr Anfilov	2	X	Author agrees to fix this.
4.	Function checkForm	Intesar Haider	3		

Lab Report

Exercise number: 1	Date: 20.03.2017
Title of the exercise: Unit Test	
Description: A Unit Test was performed on the Registration Module of the Fulda-Stadt System. This involved the fullname, AddressCode, and PassportId. One of the main challenges I faced was finding a standard format for the fullname, and PassportId check. The role of the checkForm function checked that the required fullname, AddressCode, PassportId textfields contained data that conformed to the predefined syntax. The function checkForm failed to submit the form when any of the required fields' data violated its predefined syntax. On the other hand, the function checkForm submitted the form data for further processing in the absence of any predefined syntax violation.	
Results: The test cases were designed to cover all critical aspects related to the required data. Majority of the test cases failed. These defects where fixed and the unit re-tested. The re-testing passed all the test cases. This gives a high confidence of the unit after executing these test cases.	
What did you learn? Testing is not a one-time activity but a continuous activity. Static Reviews are very useful because without executing the code, defects concerning programming logic, and good programming practices can be ensured to make it possible for future development. It is very important to ensure quality criteria of a software in addition to the functional testing. Testing shows the presence of defects and not the absence of them.	