Software Testing Report of

Typing tutor

Version <1.0>

Group Name: 4

Instructor:	**Dr. Dang Duc Hanh**
Course:	**Software Engineering 2016**
Date:	**04/12/2016**

Nguyễn Trường Giang	**K59CA**
Phạm Nguyễn Hoàng	**K59CA**
Nguyễn Văn Quang	**K57CA**
Lương Ngọc Huyền	**K59CA**
Nguyễn Văn Tiến	**K59CA**

Table of Contents

Contents

1. Introduction

1.1 Purpose

The purpose of this report is to illustrate a detailed description of Software Testing Plan to assess all the functionalities of the Typing Tutor Website System. This document is intended for both the stakeholders and the developers of the system.

1.2 References

The report was completed following the book and documentation recommended and provided by Dr. Dang Duc Hanh including:

- The requirements documentation of Typing Tutor.
- The Design Model Documentation of Typing Tutor.
- The textbook Software Engineering 9th Edition written by Sommerville, Chapter 8
- The series of lectures and slides of Software Engineering Course 2016 provided by Dr. Dang Duc Hanh, specifically lecture 8.

2. Software Testing

2.1 Items under Testing

The Typing Tutor system will consist of two basic areas of testing. The first item under test is how the information will be retrieved and displayed to the user based on their initial request. The second item assessed is how the information is stored within the system database tables.

2.2 Scope

The scope of this document is to test the Typing Tutor system by each component, component integration, and full system functionality. However, due to the lack of available project time, this Typing Tutor will only include unit-testing procedures. The unit-testing will be performed by employing SeleniumHQ version to generate the Java unit-testing cases.

2.3 Approach

2.3.1 Unit Testing

This process involves the testing of particular system components. These components are isolated from other portions and tested for their input, output, and module procedures.

2.3.2 Integration Testing

Integration testing procedures incorporate system components and how they perform together functionally between one another. System parts are built together forming new interfaces and these are tested to determine.

2.3.3 System Testing

This involves the complete integration of all system components and how they perform as a whole unit. This type of testing validates the entire system as a functional entity.

3.0 TEST DESIGN

3.1 Approach

3.1.1 Unit Test

3.1.1.1 “Login valid user” test case

Login as specific user with their Facebook account including their email and their password. System should display the current Home page and their account accessible at the right corner of website.

3.1.1.2 “Login Invalid user” test case
Login as specific user with the email and incorrect password. System should display screen with authentication-failed message.

3.1.1.3 “Navigate to Home Page” test case
The user clicks on Home button from the left of navigation bar and system should display the Home page correctly to the user.

3.1.1.4 “Navigate to Profile Page” test case
The user clicks on the link “Welcome user’s name” from the right of navigation bar and system should display properly the Profile of this user to respond the user’s request.

3.1.1.5 “Learn Basic Lesson” test case
The user click on the basic lesson that is displayed in the basic lesson list of the Basic Lesson Page, then the system should return and display correctly the lesson that the user wants to learn.

3.1.1.6 “Practice Lesson” test case
The user click on the practice lesson that is displayed in the practice lesson list of the Practice Page, then the system should return and display correctly the lesson that the user wants to practice.

3.1.1.7 “Play Game” test case
The user click on the name of game from the navigation bar, then the system should return and display correctly the Game page that the user requests to play.

4. TEST CASES

4.1 “Login valid user” test case

1.1 request.getParameter() does not result in a null value
1.1.1 quality parameter= “userName”
1.1.2 Expected input: login value =”password”
1.2 request.getParameter() does not result in a null value
1.2.1 quality parameter=”password”
1.2.2 password value=”password”
Execute: validate(“hello”, “password”)

Expected output: Display main screen (Home Page) with successful login notification from Facebook of the user.

4.2 “Login Invalid user” test case

1.1.0 request.getParameter() does not result in a null value
1.1.1 parameter value= “userName”
1.1.2 Expected Input: login value =”wrongusername”
1.2 request.getParameter() does not result in a null value
1.2.1 parameter value=”password”
1.2.2 Expected Input: password value=”wrongpasswd”
Execute: validate(“wrongusername”, “wrongpasswd”)
Expected output: Display screen with authentication-failed message from the Facebook Oath Login.

4.3 “Navigate to Home Page” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Home.php"
1.1.2 page value = "/WEB-INF/Home.php"
Execute:request.getRequestDispatcher("/WEBINF/Home.php").forward(request, response)
Expected output: The system return the Home page as requested.

4.4 “Navigate to Profile Page” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Profile.php"
1.1.2 page value = "/WEB-INF/Profile.php "
Execute: request.getRequestDispatcher("/WEB-INF/Profile.php ").forward(request, response)
Expected output: The system will return the Profile web Page of the user as requested.

4.5 “Learn Basic Lesson” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Lesson.php/detail/1"
1.1.2 page value = "/WEB-INF/Lesson.php/detail/1"
Execute: request.getRequestDispatcher("/WEB-INF/Lesson.php/detail/1").forward(request, response)
Expected output: The system will return the Basic Lesson Page in accordance with the id of lesson requested.

4.5 “Practice Lesson” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Practice.php/paragraph/100”
1.1.2 page value = "/WEB-INF/Practice.php/paragraph/100”
Execute: request.getRequestDispatcher("/WEB-INF/Practice.php/paragraph/100").forward(request, response)
Expected output: The system will return the Practice Lesson Page in accordance with the length of lesson requested.

4.6 “Bubbles Game” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Bubbles.php”
1.1.2 page value = "/WEB-INF/Bubbles.php”
Execute: request.getRequestDispatcher("/WEB-INF/ Bubbles.php").forward(request, response)
Expected output: The system will return the Bubble Game Page in accordance as requested.

4.7 “Random Game” test case

1.1.0 id.equals() does not result in a null value
1.1.1 parameter value = "/Random.php”
1.1.2 page value = "/WEB-INF/Random.php”
Execute: request.getRequestDispatcher("/WEB-INF/ Random.php").forward(request, response)
Expected output: The system will return the Random Game Page in accordance as requested.

Index