Agile Mind –3ndIncrement Report

## Team Members :

## Kendall Bingham, Vidya Sridhar

## Design of Services

#### Service Description :

The following features are implemented :

* Short Term Memory oriented Quizzes : This game, is again, focused at the audience whose memory is in bad shape. Short term and long term memory issues is a battle struggled by many of the older age group. This game has a series of questions that has basic questions asked as a series of passages, whose details are to be remembered and recollected by an individual. After a certain amount of time, in our case, a minute, the passage’s corresponding questions will be asked.
* This consists of first reading the comprehensive passages and then filling out the multiple choice questions soon after, after a stipulated period of time.

#### Class Diagram

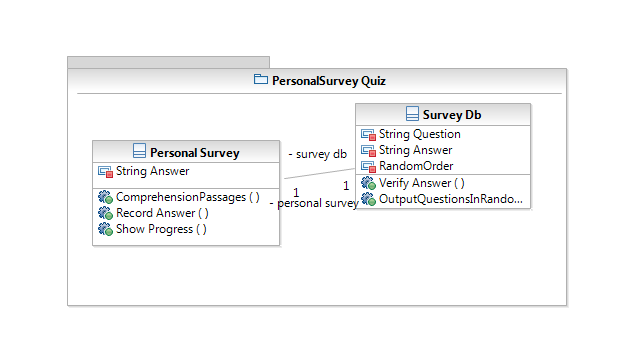


Fig 1 :Personal Quiz Class Diagram

#### Sequence Diagram

Personal Quiz :

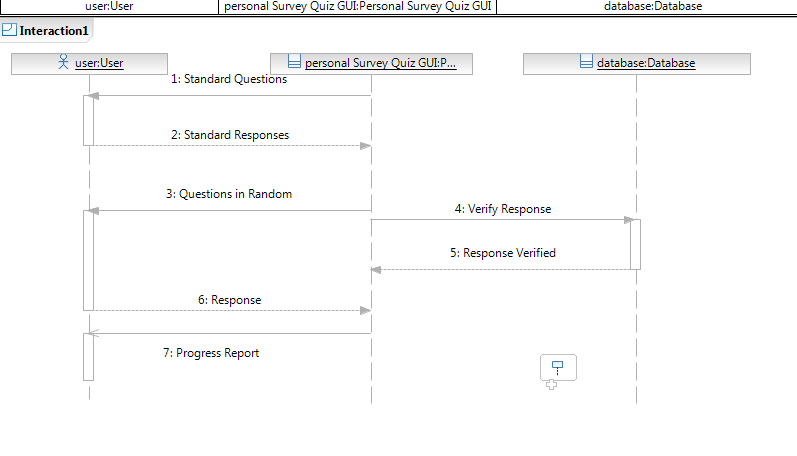
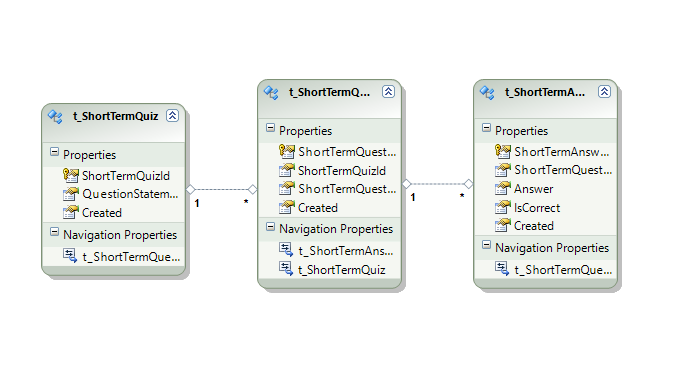
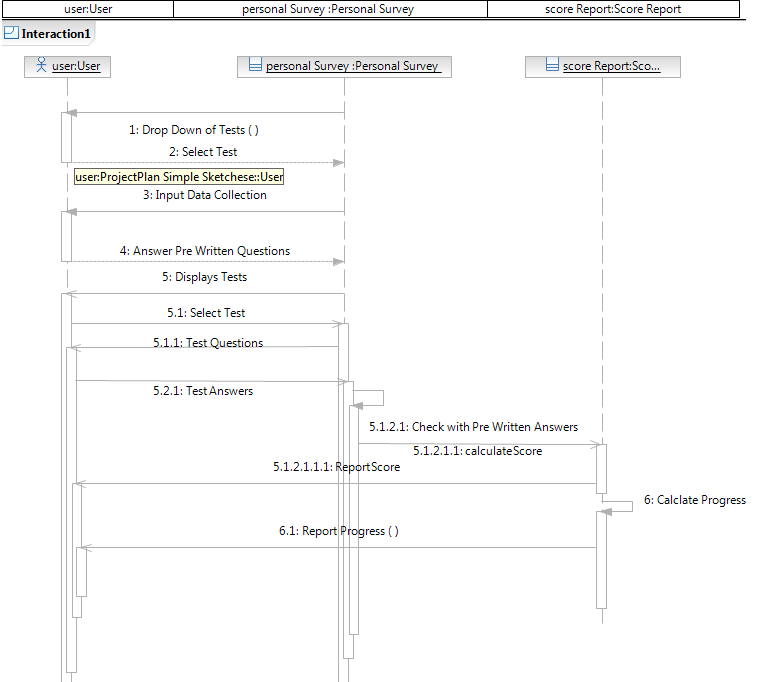


Fig 2 :Personal Quiz Sequence Diagram





#### Design of Unit Test Cases [Inclusive of Increment 1]

Design of the test cases should be designed for the following :

Check if Login credentials are returned

Check If the user’s login data is indeed written into the database

Check if the right information is output corresponding to input user’s credentials in the database

Check if Login is successful depending on user name, its existence.

Check if Login is successful depending on whether password and user id is correct.

**Increment 2**  
Check if the user profile saves all the input data

Check if profile quiz is accessible

Check if profile quiz gives correct output

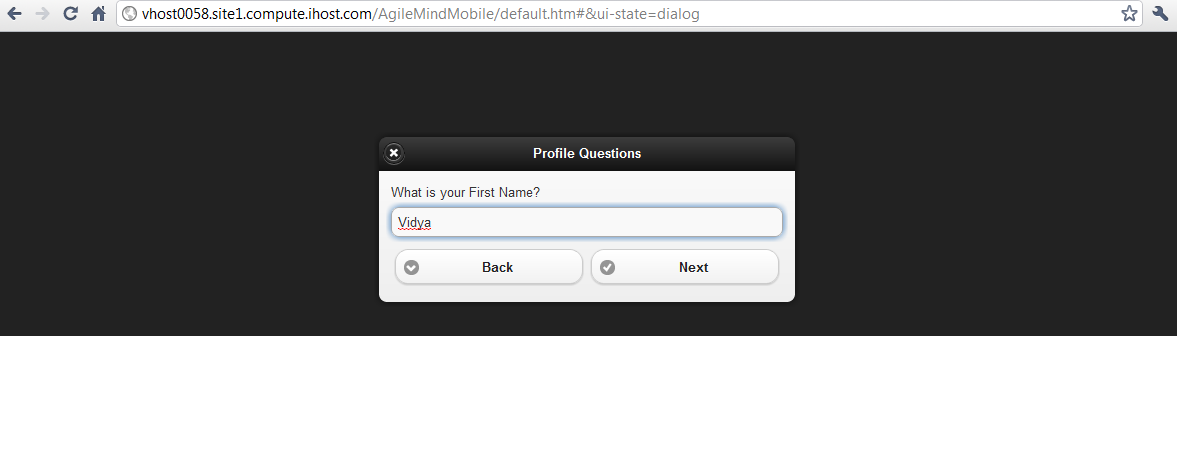
# Implementation

#### Service Implementation

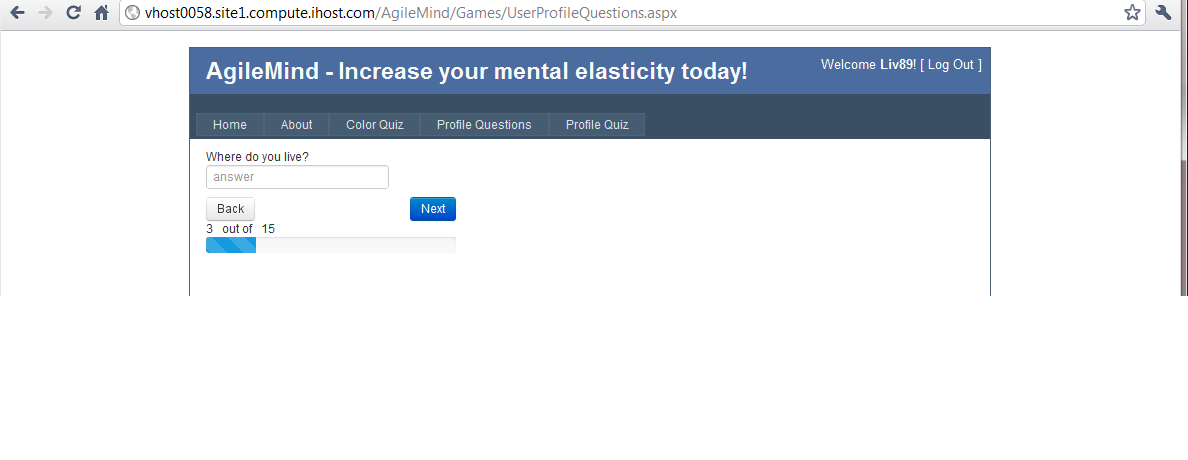
Services include both the mobile and full web based ones.

#### User Interface Implementation

Mobile :



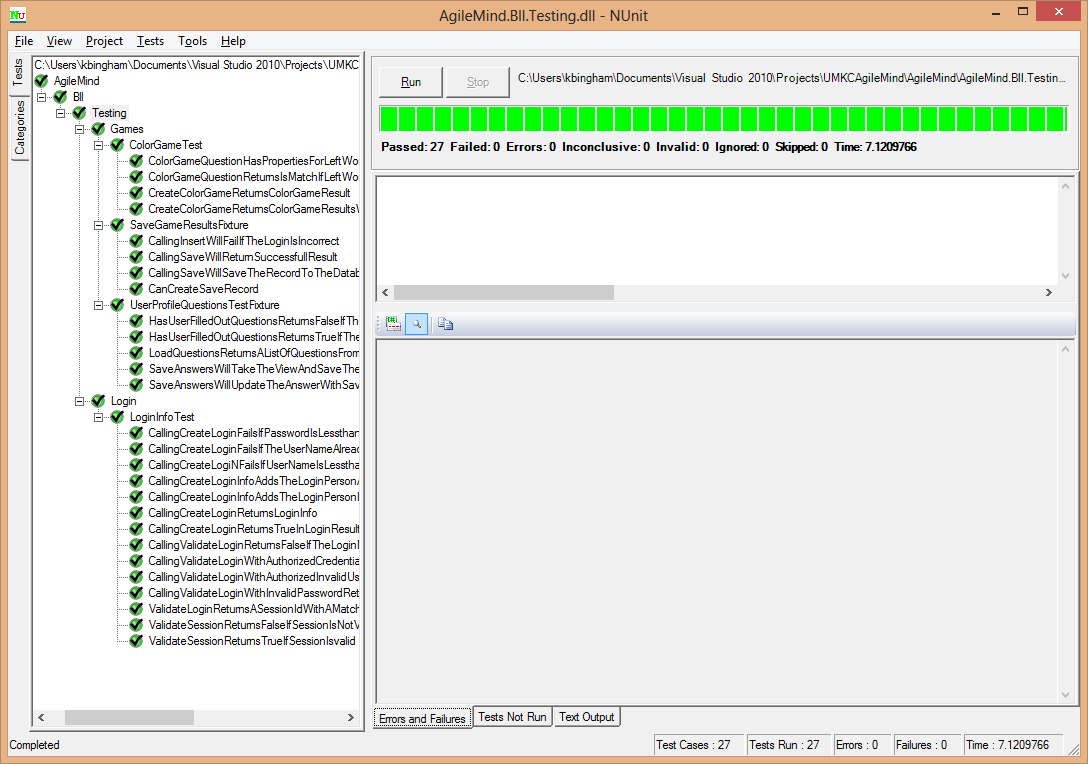
Web:



#### Test Case Implementation

Test Cases Implemented are :

Increment 2



Previous Tests :

**CallingCreateLoginReturnsLoginInfo()**

**CallingCreateLoginInfoAddsTheLoginPersonIntoTheDatabase()**

**CallingCreateLoginInfoAddsTheLoginPersonAndReturnsItInLoginInfoObject()**

**CallingCreateLoginReturnsTrueInLoginResult()**

**CallingCreateLoginFailsIfTheUserNameAlreadyExists()**

**CallingCreateLogiNFailsIfUserNameIsLessthan5Long()**

**CallingValidateLoginWithAuthorizedCredentialsReturnsTrueAndTheLoginInformation()**

**CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()**

**CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()**

**CallingValidateLoginWithInvalidPasswordReturnsFalse()**

**CallingValidateLoginReturnsFalseIfTheLoginNameAndPasswordAreCorrectButActiveIsFalse()**

# Testing

All previously ran test codes running fine.

**Test Codes : 11 Test Case Units**

#region -- using declarations --

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using NUnit.Framework;

using AgileMind.BLL.Login;

using System.Data;

using System.Data.Entity;

#endregion

namespace AgileMind.Bll.Testing.Login

{

[TestFixture()]

public class LoginInfoTest

{

/\*-- Testing --\*/

/\*-- CreateLogin Testing --\*/

#region -- **CallingCreateLoginReturnsLoginInfo()** Method --

[Test()]

public void CallingCreateLoginReturnsLoginInfo()

{

String loginName = "TestName";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginInfo = LoginResult.CreateLogin(loginName, password, email);

Assert.IsNotNull(loginInfo);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingCreateLoginInfoAddsTheLoginPersonIntoTheDatabase()** Method --

[Test()]

public void CallingCreateLoginInfoAddsTheLoginPersonIntoTheDatabase()

{

String loginName = "TestName";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginInfo = LoginResult.CreateLogin(loginName, password, email);

AgileMind.DAL.Data.AgileMindEntities agileMind = new DAL.Data.AgileMindEntities();

List<AgileMind.DAL.Data.Login> loginList = agileMind.Logins\_CheckLogin(loginName, password).ToList();

Assert.AreEqual(1, loginList.Count, "After insertion of new login there should be one login");

Assert.AreEqual(loginName, loginList[0].LoginName, "Login Name should be set equal");

Assert.AreEqual(password, loginList[0].Password, "Password should be equal");

Assert.AreEqual(email, loginList[0].EmailAddress);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingCreateLoginInfoAddsTheLoginPersonAndReturnsItInLoginInfoObject()** Method --

[Test()]

public void CallingCreateLoginInfoAddsTheLoginPersonAndReturnsItInLoginInfoObject()

{

String loginName = "TestName";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginInfo = LoginResult.CreateLogin(loginName, password, email);

Assert.IsNotNull(loginInfo.LoginInfo);

Assert.AreEqual(loginName, loginInfo.LoginInfo.LoginName);

Assert.AreEqual(email, loginInfo.LoginInfo.EmailAddress);

DeleteLoginsFromDB(loginName);

}

#endregion

#region **-- CallingCreateLoginReturnsTrueInLoginResult()** Method --

[Test()]

public void CallingCreateLoginReturnsTrueInLoginResult()

{

String loginName = "TestName";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

Assert.IsTrue(loginResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

#region **-- CallingCreateLoginFailsIfTheUserNameAlreadyExists()**

Method --

[Test()]

public void CallingCreateLoginFailsIfTheUserNameAlreadyExists()

{

String loginName = "TestName";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

LoginResult duplicateResult = LoginResult.CreateLogin(loginName, password, email);

Assert.IsFalse(duplicateResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingCreateLogiNFailsIfUserNameIsLessthan5Long()** Method --

[Test()]

public void CallingCreateLogiNFailsIfUserNameIsLessthan5Long()

{

String loginName = "Test";

String password = "Password";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

Assert.IsFalse(loginResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingCreateLoginFailsIfPasswordIsLessthan6Long()** Method --

[Test()]

public void CallingCreateLoginFailsIfPasswordIsLessthan6Long()

{

String loginName = "TestAccount";

String password = "Passw";

String email = "Email";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

Assert.IsFalse(loginResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

/\*-- Validate Login testing --\*/

#region -- **CallingValidateLoginWithAuthorizedCredentialsReturnsTrueAndTheLoginInformation()** Method --

[Test()]

public void CallingValidateLoginWithAuthorizedCredentialsReturnsTrueAndTheLoginInformation()

{

String loginName = "TestAccount";

String password = "Password";

String email = "Email@test.com";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

LoginResult validateResult = LoginResult.ValidateLogin(loginName, password);

Assert.IsTrue(validateResult.Success);

Assert.IsNotNull(validateResult.LoginInfo);

Assert.AreEqual(loginName, validateResult.LoginInfo.LoginName);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()** Method --

[Test()]

public void CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()

{

String loginName = "TestAccount";

String password = "Password";

String email = "Email@test.com";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

LoginResult validateResult = LoginResult.ValidateLogin("IncorrectAccount", password);

Assert.IsFalse(validateResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- CallingValidateLoginWithInvalidPasswordReturnsFalse() Method --

[Test()]

public void **CallingValidateLoginWithInvalidPasswordReturnsFalse()**

{

String loginName = "TestAccount";

String password = "Password";

String email = "Email@test.com";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

LoginResult validateResult = LoginResult.ValidateLogin(loginName, "nope");

Assert.IsFalse(validateResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

#region -- **CallingValidateLoginReturnsFalseIfTheLoginNameAndPasswordAreCorrectButActiveIsFalse()** Method --

[Test()]

public void CallingValidateLoginReturnsFalseIfTheLoginNameAndPasswordAreCorrectButActiveIsFalse()

{

String loginName = "TestAccount";

String password = "Password";

String email = "Email@test.com";

DeleteLoginsFromDB(loginName);

LoginResult loginResult = LoginResult.CreateLogin(loginName, password, email);

AgileMind.DAL.Data.AgileMindEntities agileMindDB = new DAL.Data.AgileMindEntities();

List<AgileMind.DAL.Data.Login> loginList = (from data in agileMindDB.Logins where data.LoginName == loginName select data).ToList();

foreach (AgileMind.DAL.Data.Login login in loginList)

{

login.Active = false;

}

agileMindDB.SaveChanges();

LoginResult validateResult = LoginResult.ValidateLogin(loginName, password);

Assert.IsFalse(validateResult.Success);

DeleteLoginsFromDB(loginName);

}

#endregion

/\*-- Helper Methods --\*/

#region -- DeleteLoginsFromDB() Method --

private void DeleteLoginsFromDB(String LoginName)

{

AgileMind.DAL.Data.AgileMindEntities agileMindDB = new DAL.Data.AgileMindEntities();

List<AgileMind.DAL.Data.Login> loginList = (from p in agileMindDB.Logins where p.LoginName == LoginName select p).ToList();

foreach (AgileMind.DAL.Data.Login loginToDelete in loginList)

{

agileMindDB.DeleteObject(loginToDelete);

}

agileMindDB.SaveChanges();

}

#endregion

}

}

# Deployment

#### Cloud site URL

#### Website :

<http://vhost0058.site1.compute.ihost.com/AgileMind/Games/ShortTerm.aspx>

#### Mobile :

***http://vhost0058.site1.compute.ihost.com/AgileMindMobile/default.htm***

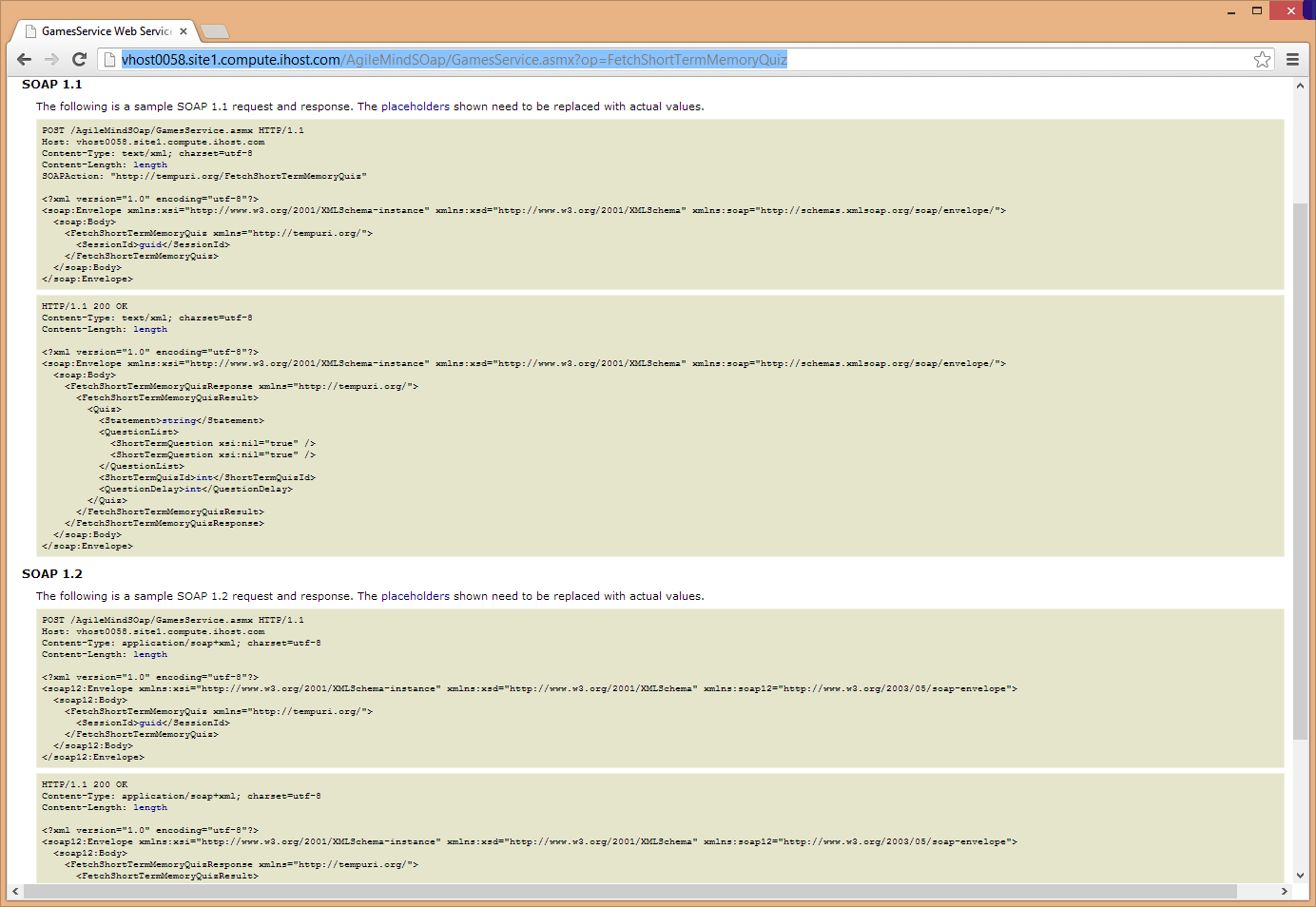
#### GIThub URL

***https://github.com/kbinghamibs/UMKC5551\_Project***

# Report

**Screen Shots**

**Web services :**

****

**Link :** http:\\vhost0058.site1.compute.ihost.com/AgileMindSOap/GamesService.asmx?wsdl

Unit Test case screenshots

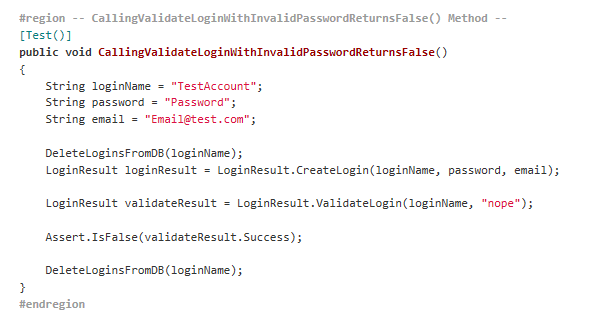
****

Fig 1 : Check if Password is valid

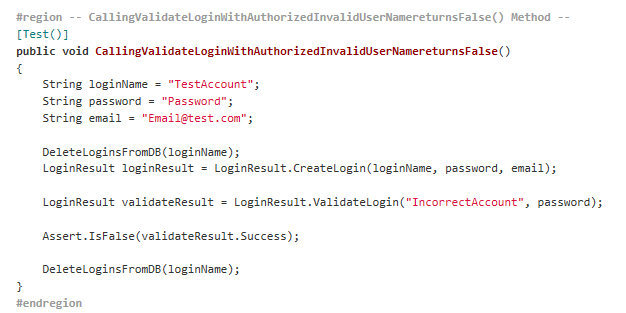


Fig 2 : check if User Name is correct

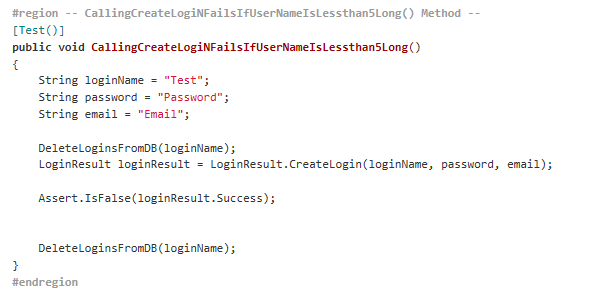
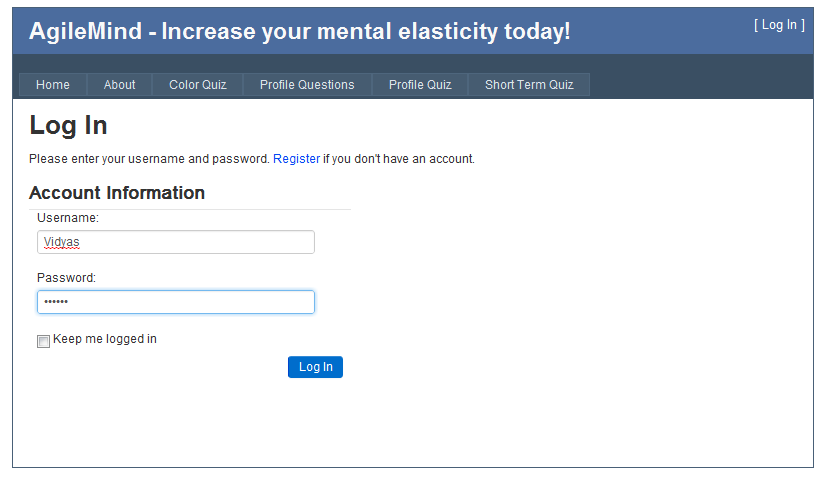
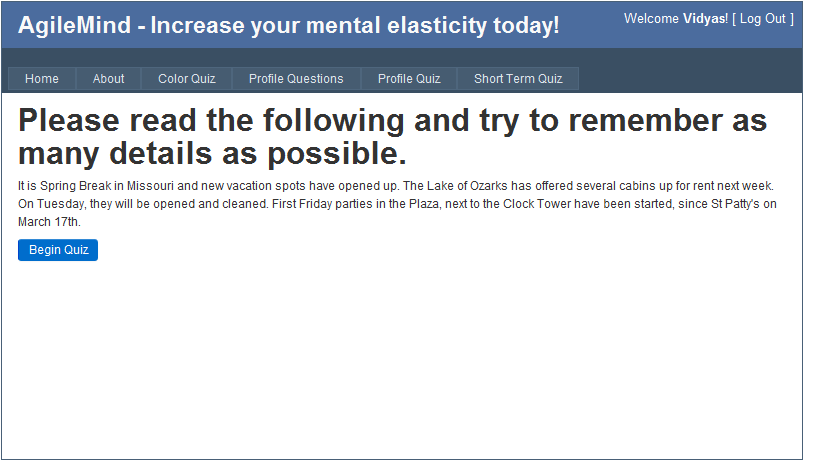


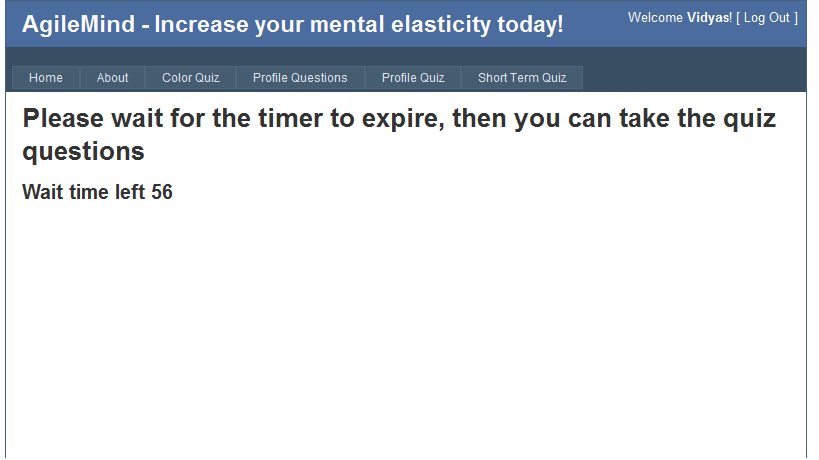
Fig 3: check if Username conforms to length requirement

User Interface :

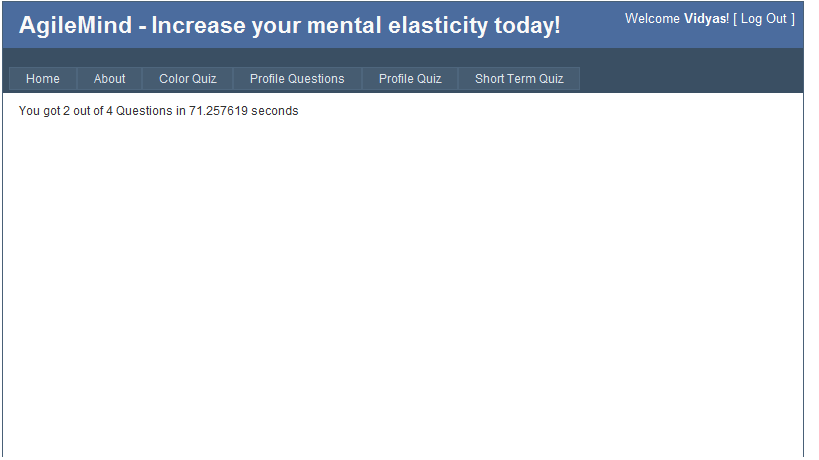
Web :



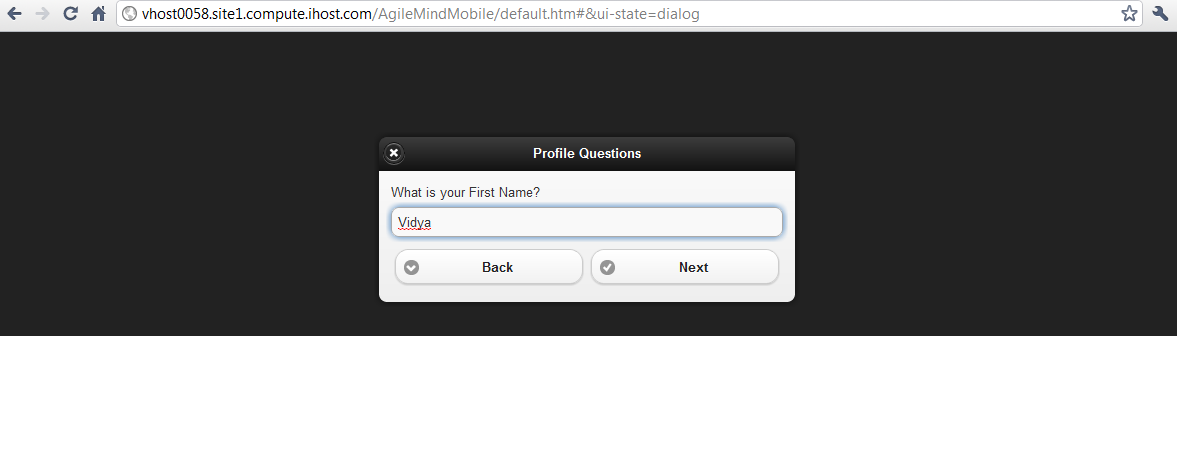


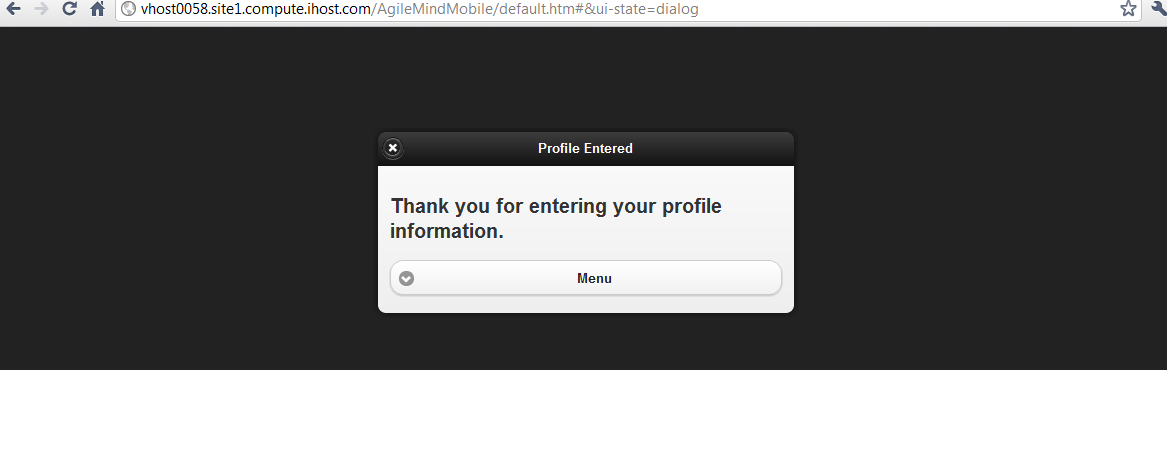






Mobile :





**Explanation on Design, implementation and testing**

The purpose of this service was to quiz the user depending on the passages he read n minutes ago.. After one comprehensive passage is read and quizzed about, the subsequent passage is required to be read, soon after which his progress may be noted.

Both a web and mobile service are created.

# Project Management

**Implementation Status Report**

**Total services :**

Login

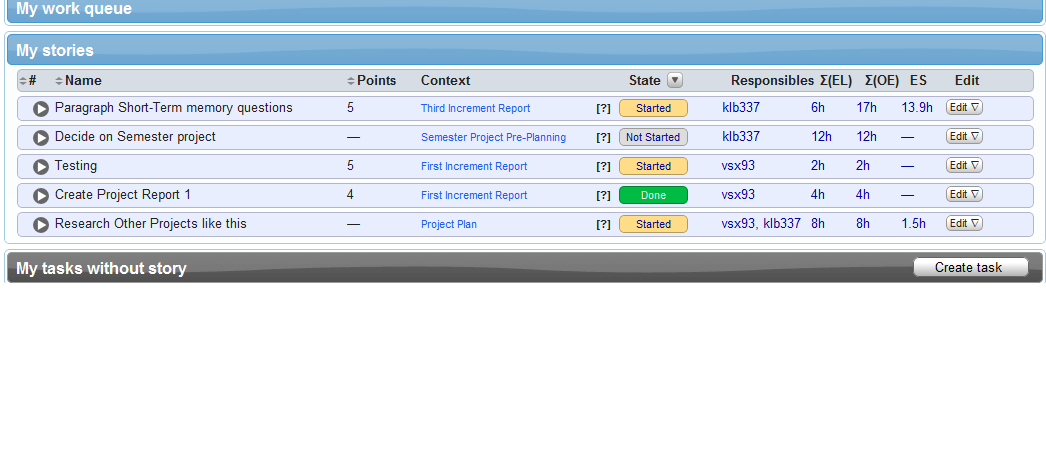
Color Match

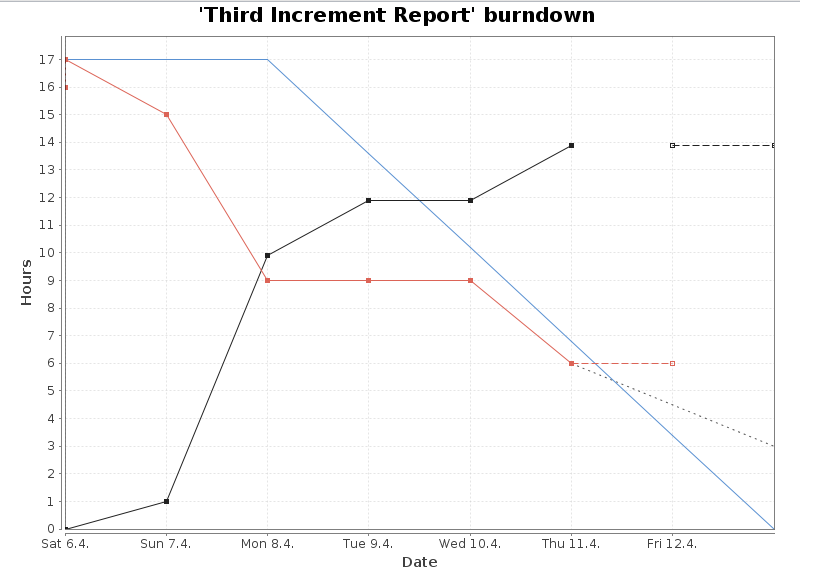
User Profile

Personal Survey Quiz

Comprehension Passage

**Screenshots**





**Issues/ Concerns :**

Syncing of the profile and user quiz in terms of accessing the mobile site’s quiz after entering personal info initially posed a problem.