Agile Mind –4th Increment Report

## Team Members :

## Kendall Bingham, Vidya Sridhar

## Design of Services

#### Service Description :

The following features are implemented :

* Summary Page. Charts show the performance summary of a person so far, after they have tracked the performance, based on every test / quiz so far.
* By checking the overall progress, the user can know how much he needs to improve.
* Identify Images. A list of images are displayed out of which the user picks the necessary word corresponding to image.
* Personal quiz is also improved upon.

#### Class Diagram

[Previous Class diagrams of features have been used]

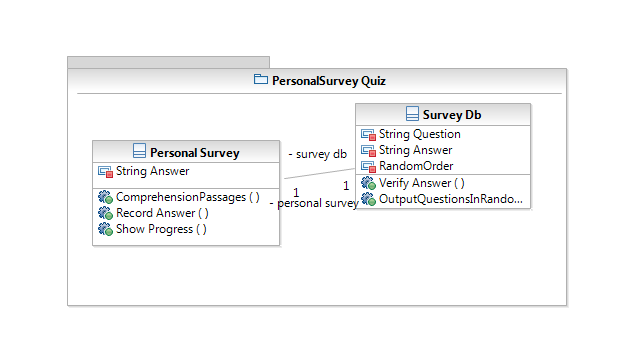


Fig 1 :Personal Quiz Class Diagram

#### Sequence Diagram

Personal Quiz :

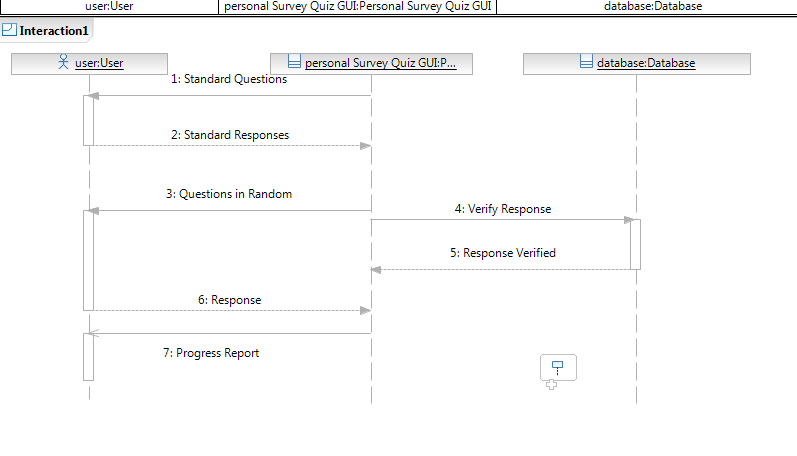
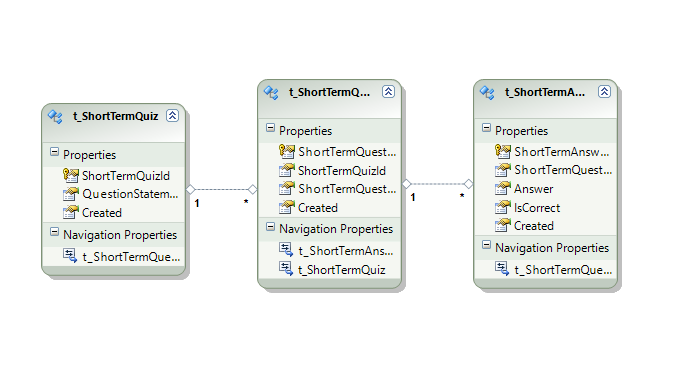
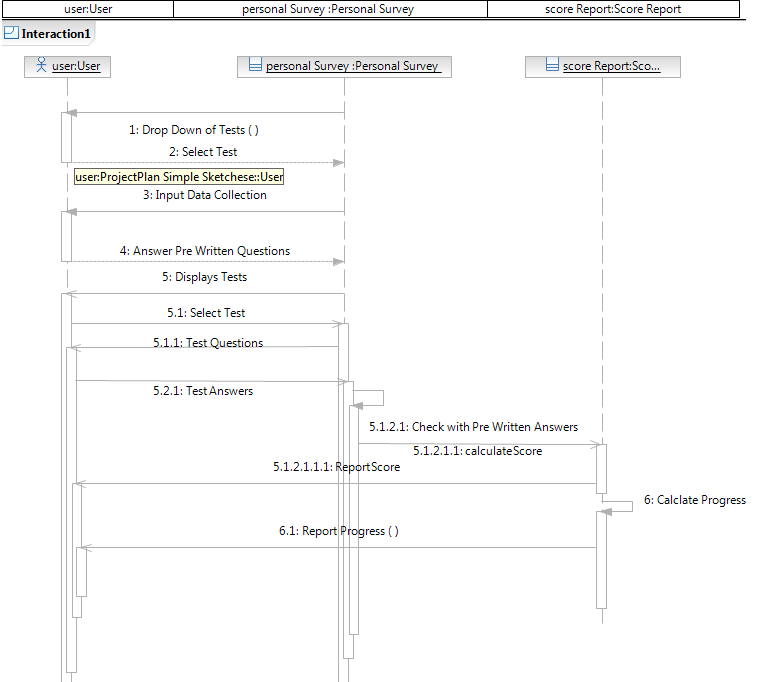


Fig 2 :Personal Quiz Sequence Diagram





#### 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

Increment 3

Check if user successfully begins short passage quiz  
Check if short passage quiz gives correct output

# Increment 4

Check if user correctly plays Identify Objects quiz

Check if summary charts are correct.

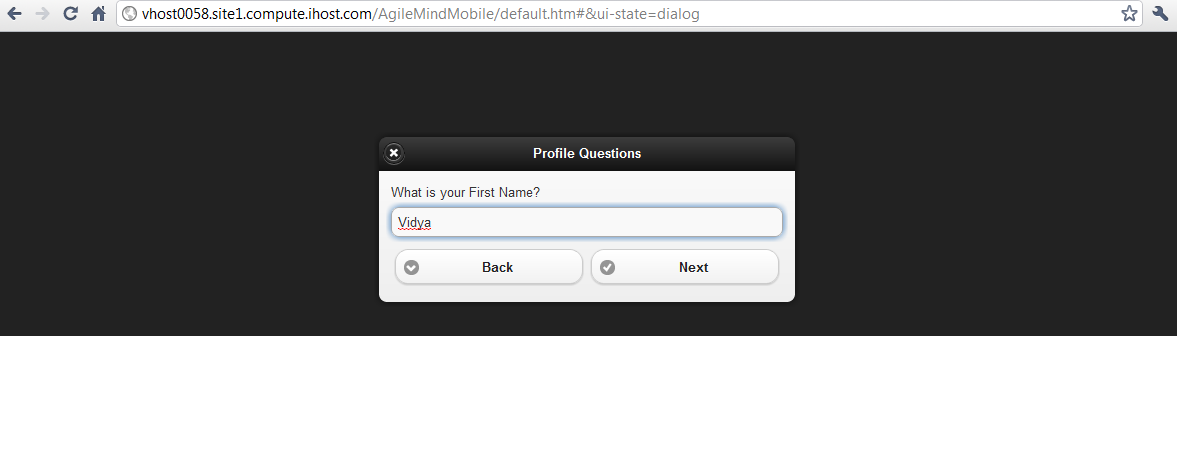
# Implementation

#### Service Implementation

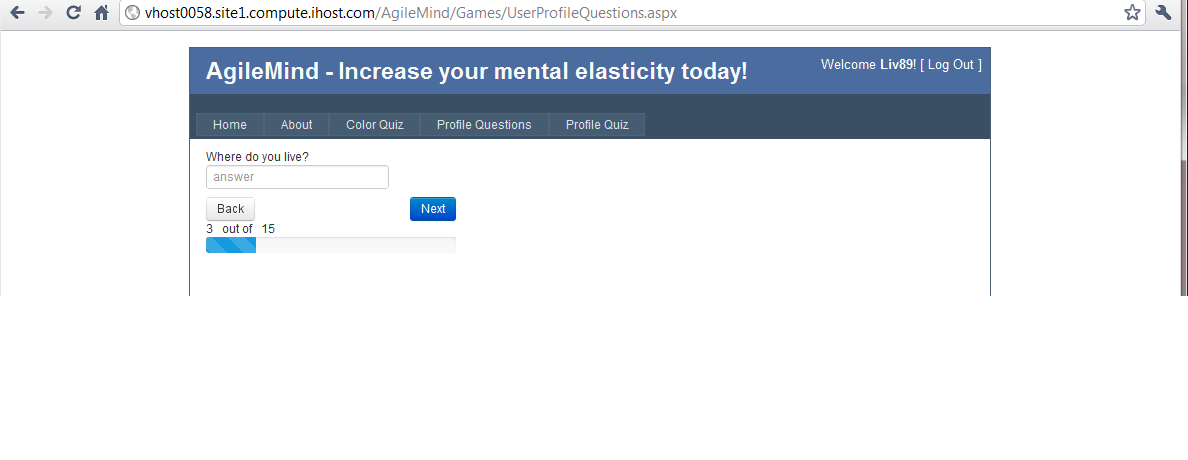
Services include both the mobile and full web based ones.

#### User Interface Implementation

Mobile :



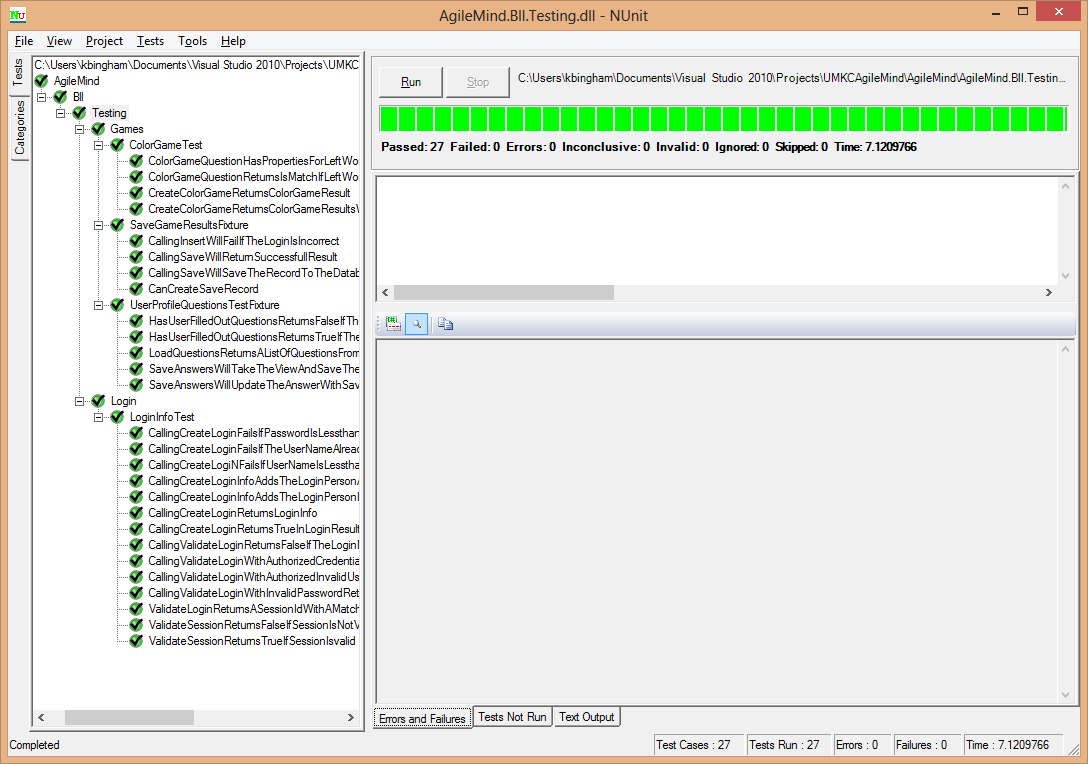
Web:



#### Test Case Implementation

Test Cases Implemented are :

Increment 3



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/Default.aspx>

#### Mobile :

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

#### GIThub URL

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

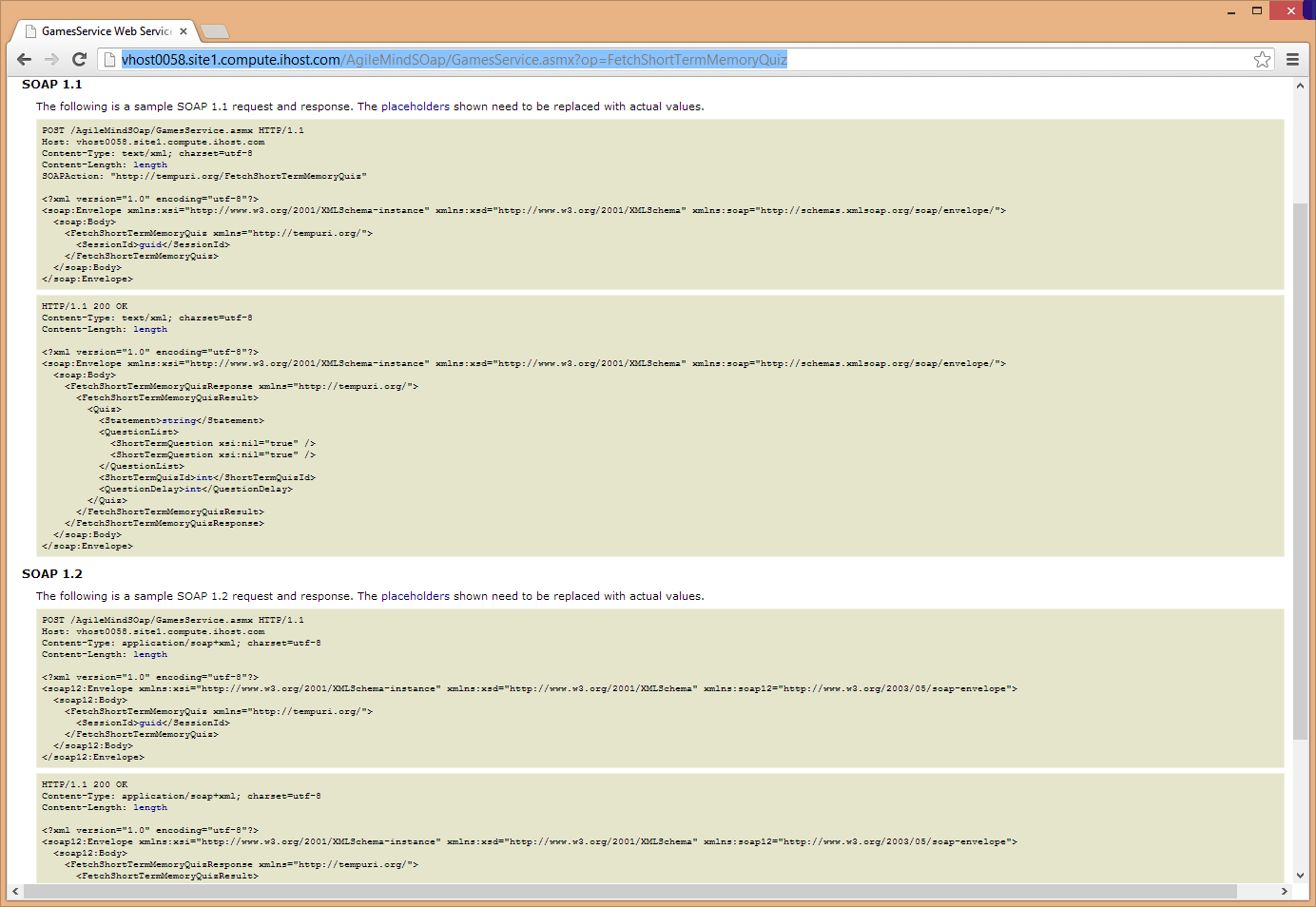
# Project Video

# *url :* [*http://www.youtube.com/watch?v=bFP-N1snhx8*](https://ch1prd0102.outlook.com/owa/redir.aspx?C=KekDmSIGIEeTnO1RCuR46qfobkfxG9AImzDvWvWAW9YwCdPtmU9tY--l3c04g4Quw6ry8w_erzc.&URL=http%3a%2f%2fwww.youtube.com%2fwatch%3fv%3dbFP-N1snhx8)

# Report

**Screen Shots**

**Web services :**

****

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

Unit Test case screenshots

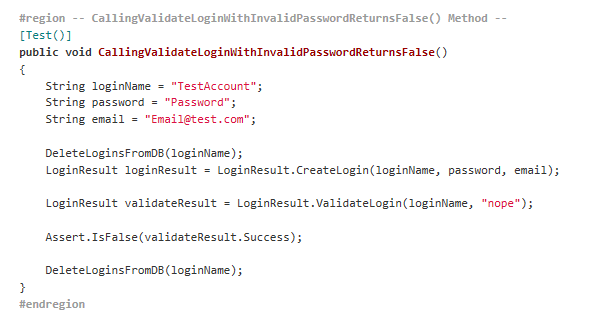
****

Fig 1 : Check if Password is valid

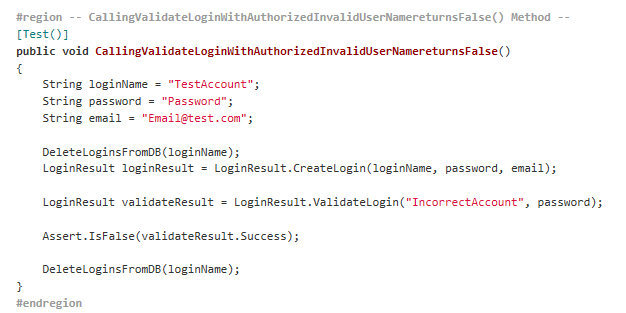


Fig 2 : check if User Name is correct

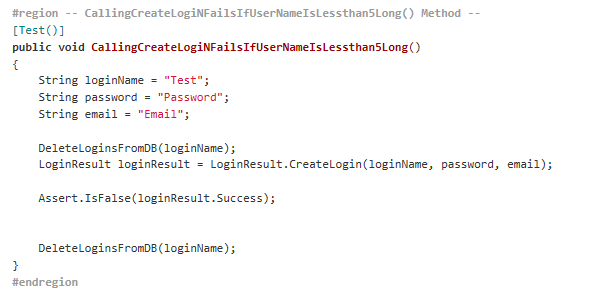
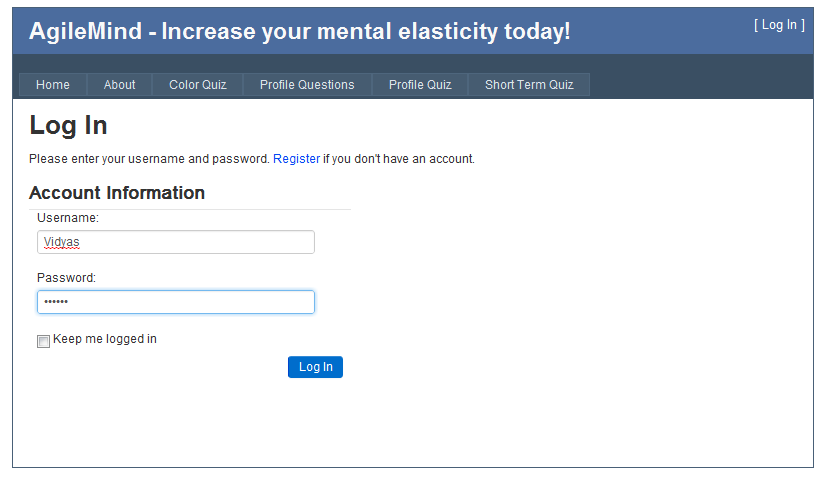
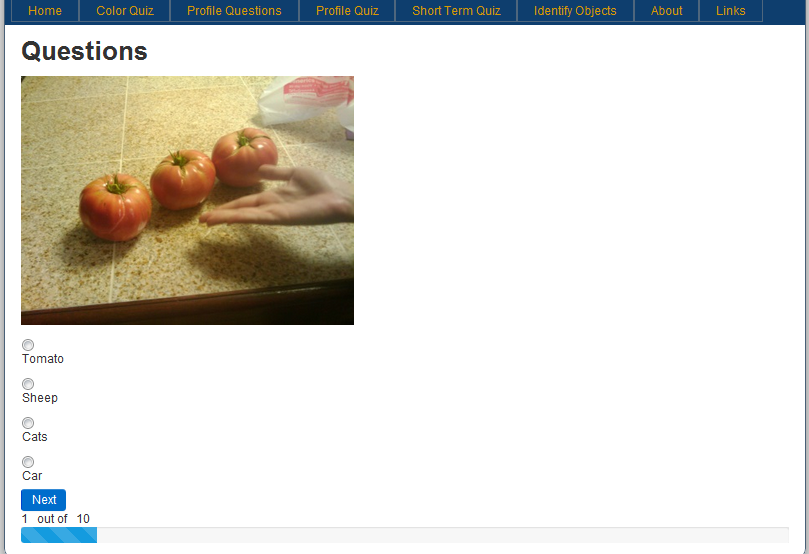


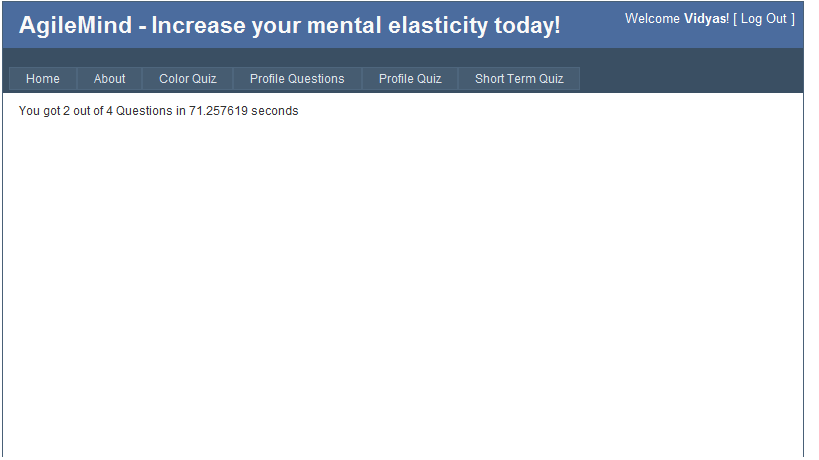
Fig 3: check if Username conforms to length requirement

User Interface :

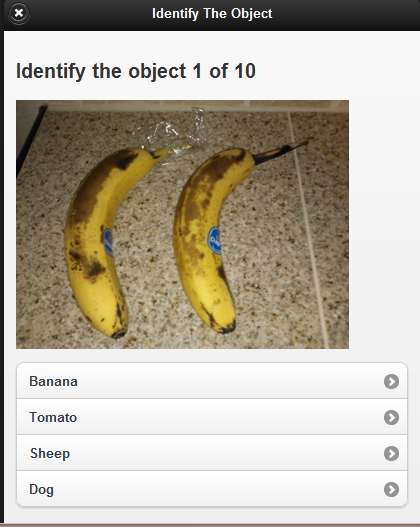
Web :

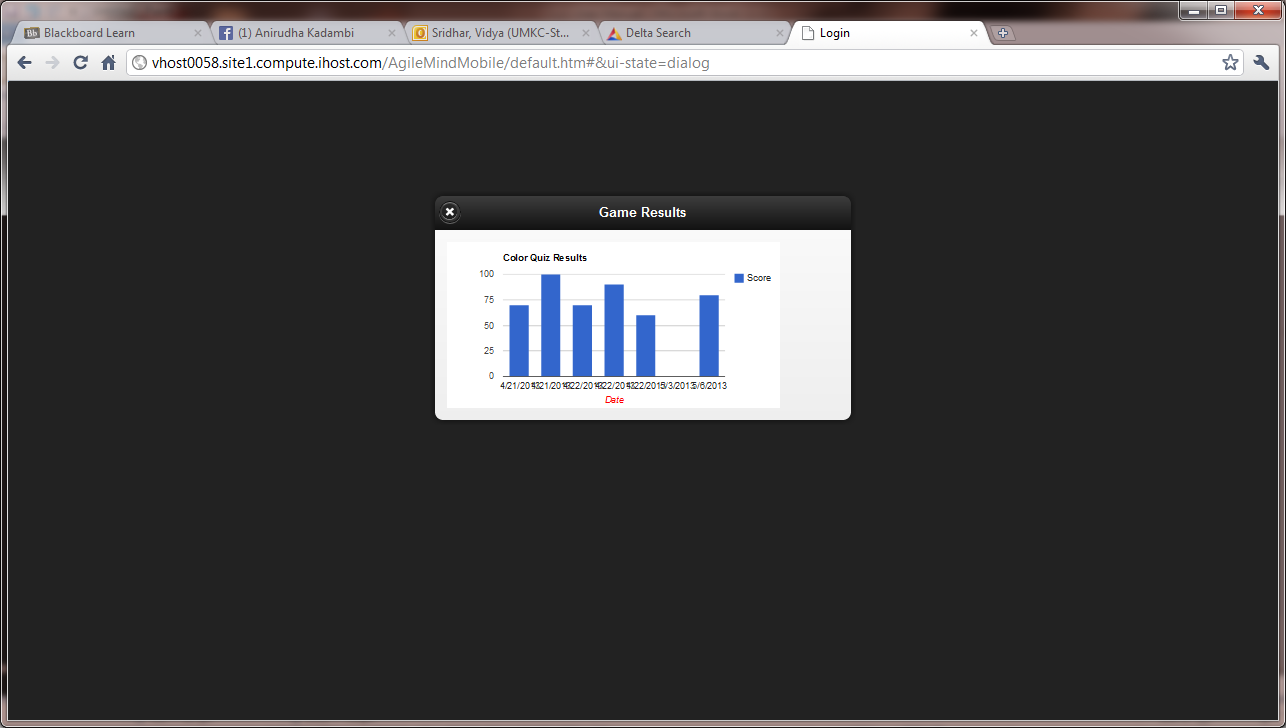






Mobile :





**Explanation on Design, implementation and testing**

The purpose of this service was to show him the progress corresponding to the game as well as to recogninze common images.

Both a web and mobile service are created.

# Project Management

**Implementation Status Report**

**Total services :**

Login

Color Match

User Profile

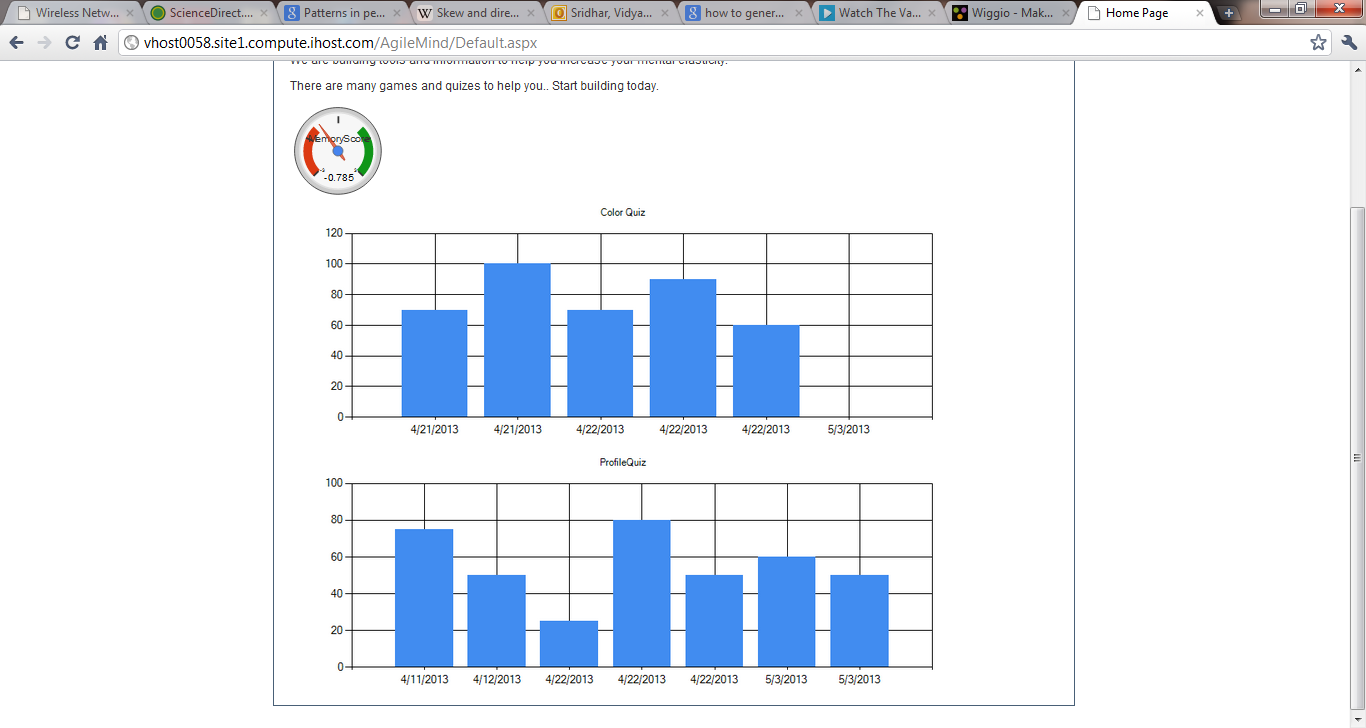
Personal Survey Quiz

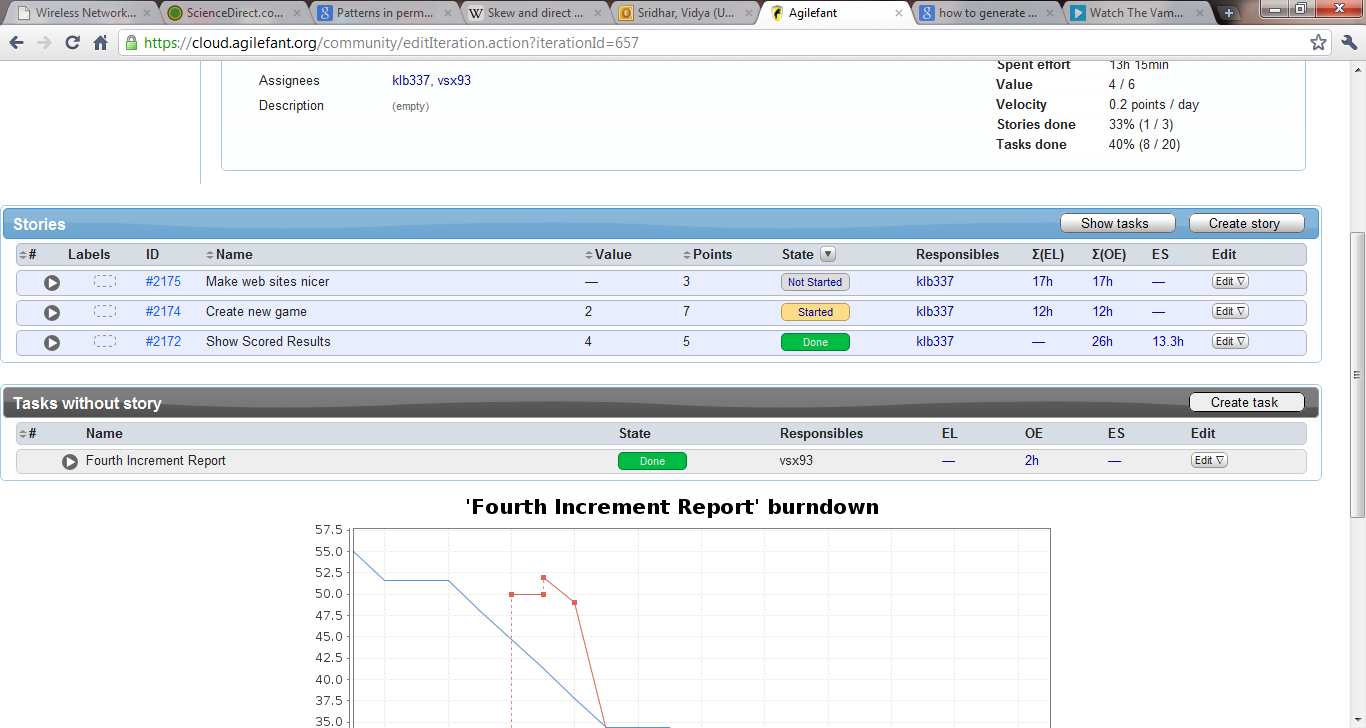
Comprehension Passage

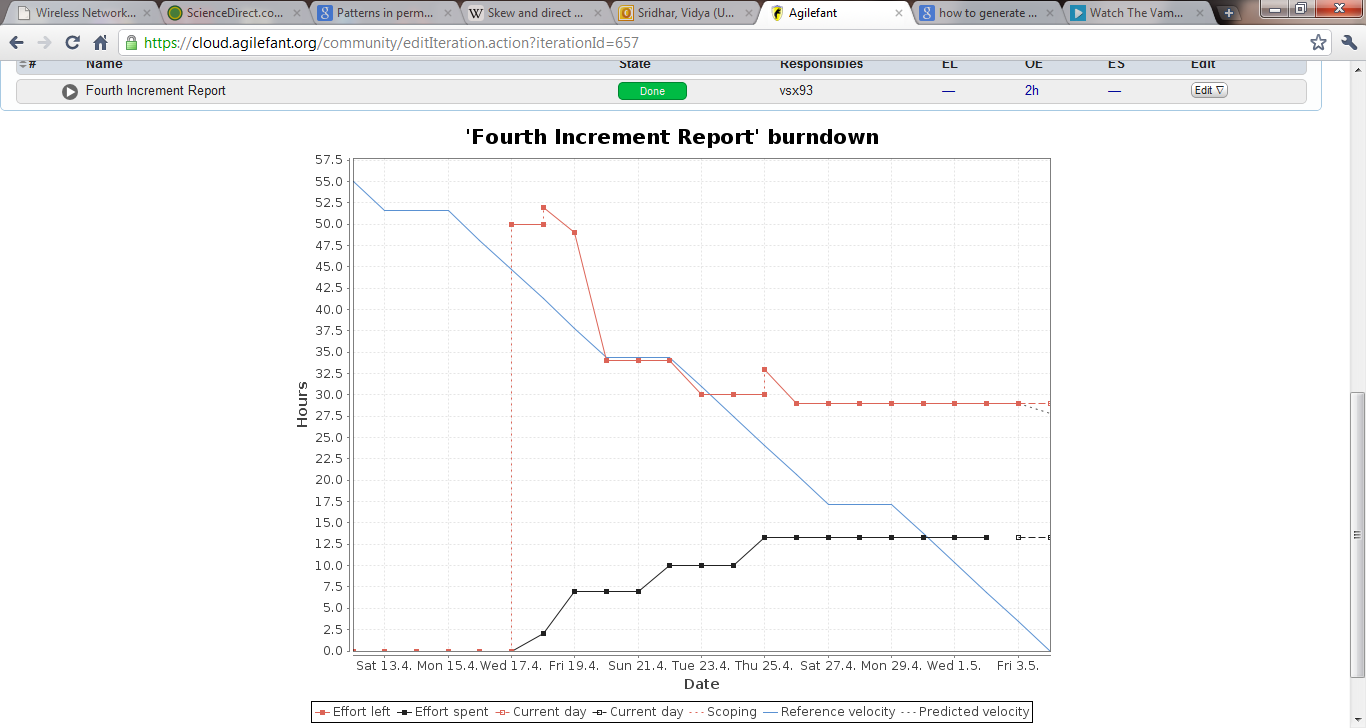
Summary Tracker

Identify Objects Game

**Screenshots**







**Issues/ Concerns :**

Adding this game to the summary charts, which was eventually taken care of.

**Future Work :**

More pictorial based web service games, as well as math puzzles/quizzes aimed for children and senior citizens.