Agile Mind – First Increment Report

## Team Members :

## Kendall Bingham, Vidya Sridhar

## Design of Services

#### Service Description :

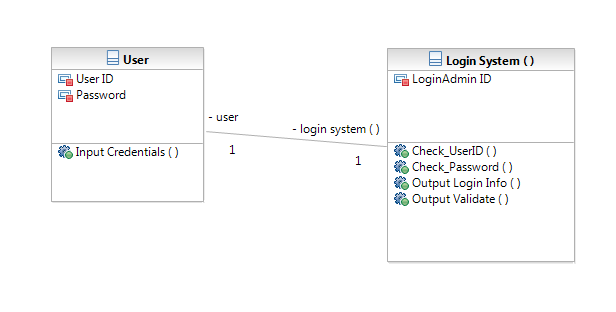
The following features are implemented :

* A login system wherein the user’s credentials are securely put in through a database, and that can be verified, and hence be accessible by the user.
* MatchColor Game : An application wherein depending on the color of the word displayed, the user input is checked, and his progress is noted. This is implemented in jquery. This has a mobile as well as a website.

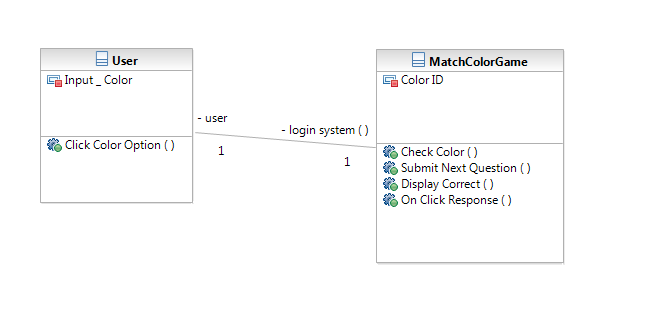
Technology used :

* asp.net Membership - The membership framework allows the programmer to use a standard framework for validation. These classes use a provider class and allows you to override the provider with a custom one. We've overridden the custom provider to check user validation and create user from a webservice instead of directly from a database.  
    
  JSON and MVC 4. - The mobile portion of the project uses an MVC 4 webservice returning JSON. JSON is a lightweight protocol that is almost 50% smaller than a standard XML payload.  
    
  Entity Data Framework. - Standard Crud operations have been created using Entity Data Framework. Keeping the code to a minimum and allowing faster coding of the project

#### Class Diagram



#### Fig 1 : Login System Class Diagram

Fig 2 : Match Color Class Diagram

#### Sequence Diagram

#### Login System :

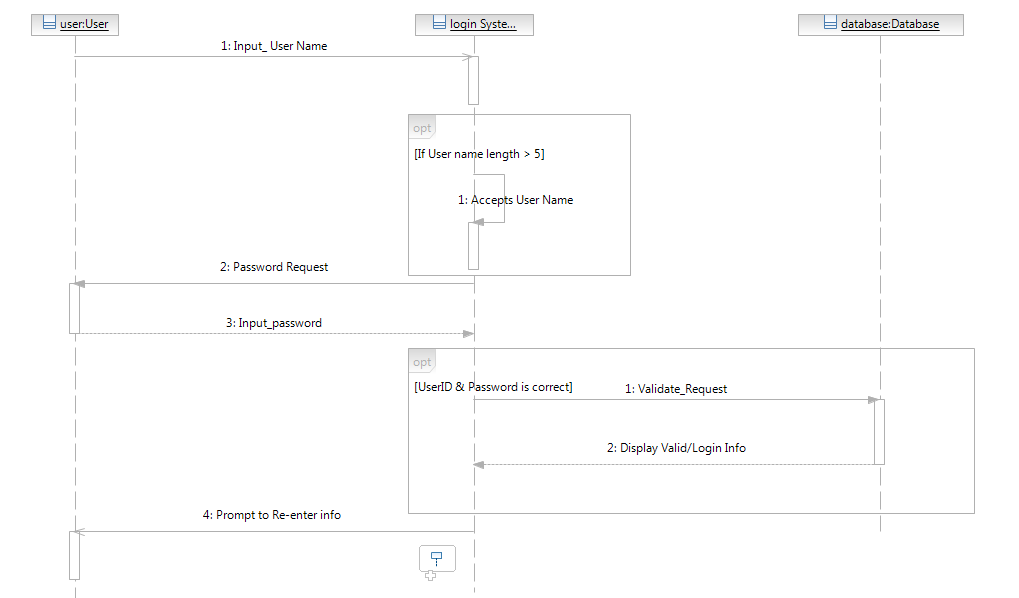
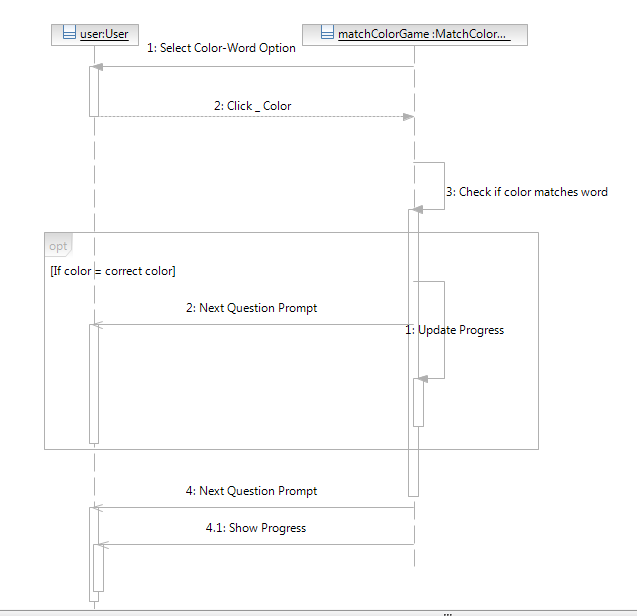


Fig 1 : Login Sequence Diagram

MatchColor Game :



#### Fig 2 : Match Color Game Sequence Diagram

#### Design of Unit Test Cases

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.

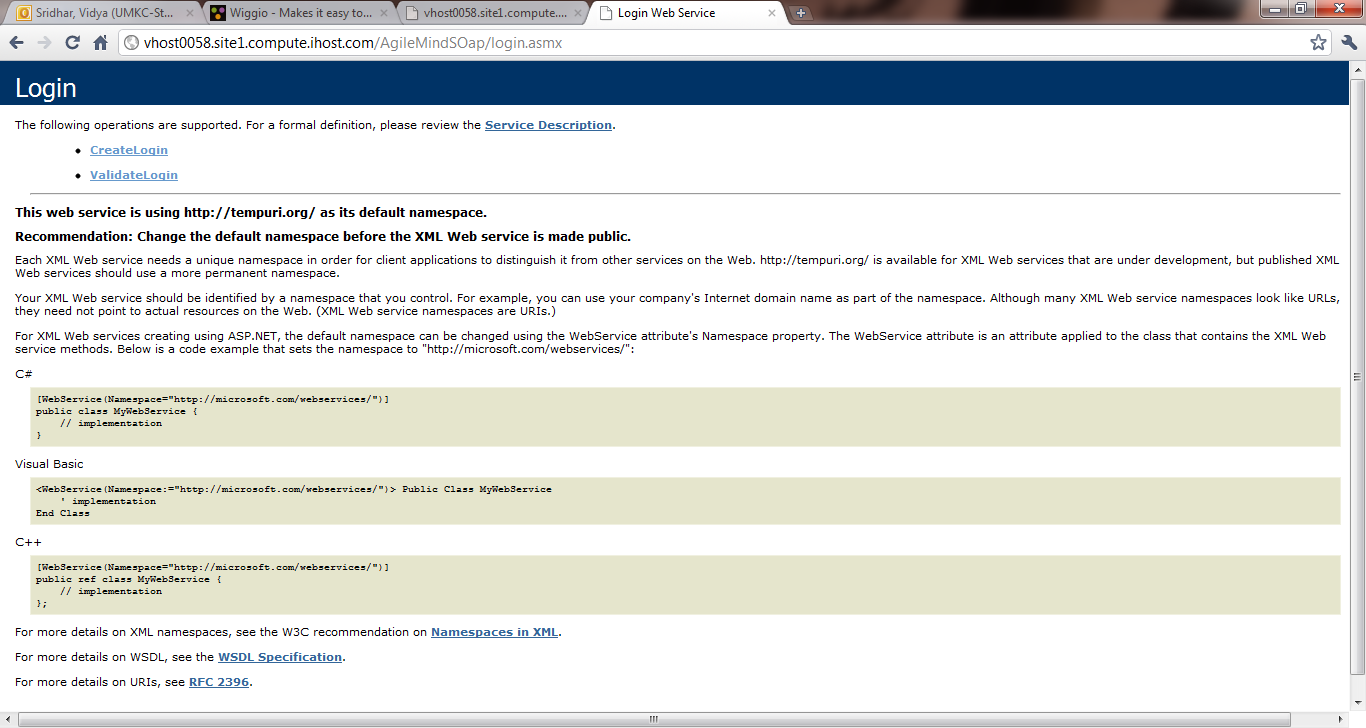
Check if Login is successful depending on whether password and user id is correct.

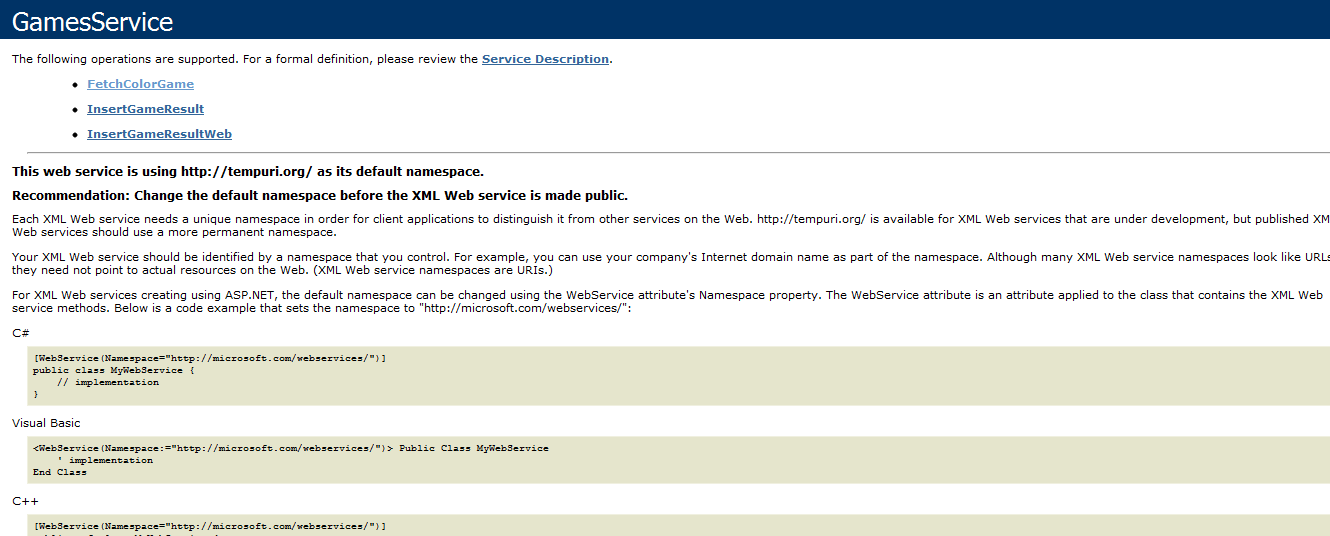
# Implementation

#### Service Implementation

**WebService Links :**

http://vhost0058.site1.compute.ihost.com/AgileMindSOap/login.asmx   
  
http://vhost0058.site1.compute.ihost.com/AgileMindSOap/GamesService.asmx

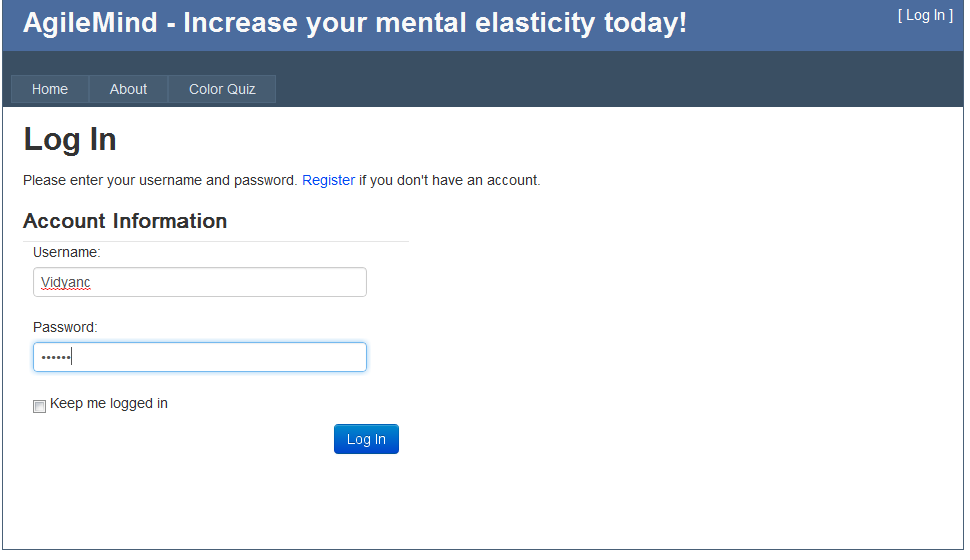




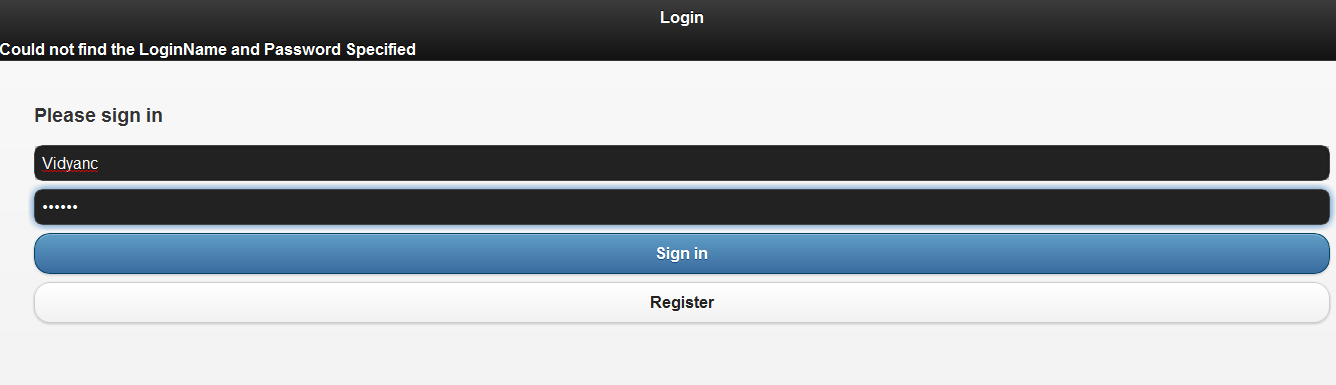
Once the user registers, they can log in and then access the game. The authentication process is done after logging in and registering can be done with a password more than the minimum character criterion.

#### User Interface Implementation

Website :



Mobile :



#### Test Case Implementation

Test Cases Implemented are :

**CallingCreateLoginReturnsLoginInfo()**

**CallingCreateLoginInfoAddsTheLoginPersonIntoTheDatabase()**

**CallingCreateLoginInfoAddsTheLoginPersonAndReturnsItInLoginInfoObject()**

**CallingCreateLoginReturnsTrueInLoginResult()**

**CallingCreateLoginFailsIfTheUserNameAlreadyExists()**

**CallingCreateLogiNFailsIfUserNameIsLessthan5Long()**

**CallingValidateLoginWithAuthorizedCredentialsReturnsTrueAndTheLoginInformation()**

**CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()**

**CallingValidateLoginWithAuthorizedInvalidUserNamereturnsFalse()**

**CallingValidateLoginWithInvalidPasswordReturnsFalse()**

**CallingValidateLoginReturnsFalseIfTheLoginNameAndPasswordAreCorrectButActiveIsFalse()**

# Testing

**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();

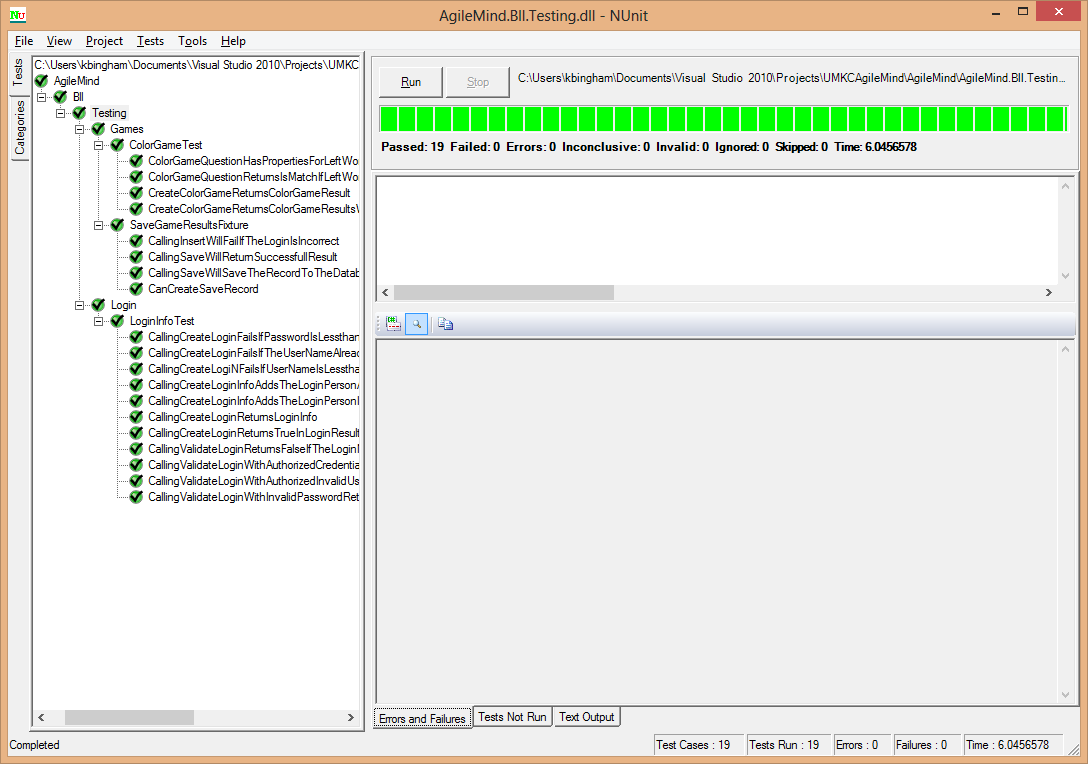
}

#endregion

}

}

# C:\Users\Vidya\Desktop\Sem 4\ASE\Project\RE-_Report_1\nUnitPassed.png



# C:\Users\Vidya\Desktop\Sem 4\ASE\Project\RE-_Report_1\ResultTable.png

# Deployment

#### Cloud site URL :

#### Websites :

http://vhost0058.site1.compute.ihost.com/AgileMind/Default.aspx

Mobile Website :

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

#### GIThub URL

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

# Report

**Explanation on Design, implementation and testing**

Technology used :

* asp.net Membership - The membership framework allows the programmer to use a standard framework for validation. These classes use a provider class and allows you to override the provider with a custom one. We've overridden the custom provider to check user validation and create user from a webservice instead of directly from a database.  
    
  JSON and MVC 4. - The mobile portion of the project uses an MVC 4 webservice returning JSON. JSON is a lightweight protocol that is almost 50% smaller than a standard XML payload.  
    
  Entity Data Framework. - Standard Crud operations have been created using Entity Data Framework. Keeping the code to a minimum and allowing faster coding of the project

Once the authentication of the user is done (name and password matches) after he registers, he is allowed to choose the game. After playing the game, his score and time are displayed.

**Screenshots**

Unit Test case screenshots

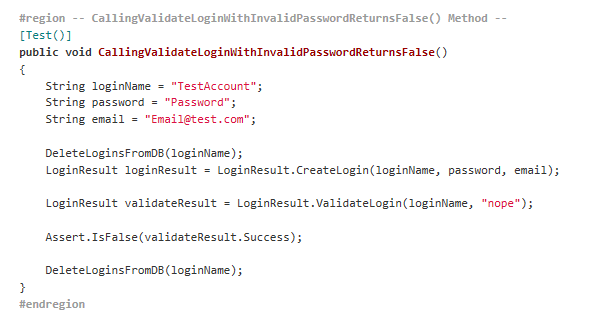
****

Fig 1 : Check if Password is valid

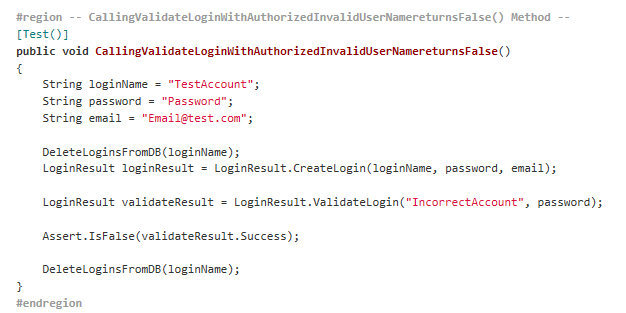


Fig 2 : check if User Name is correct

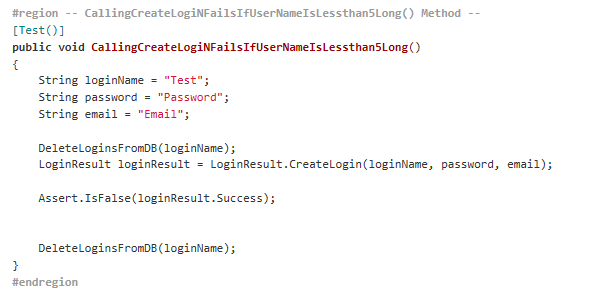
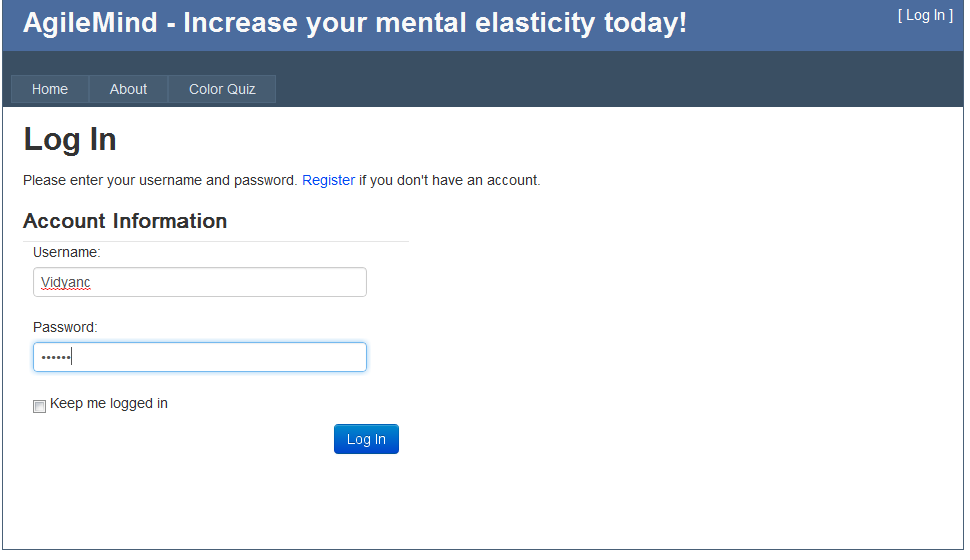
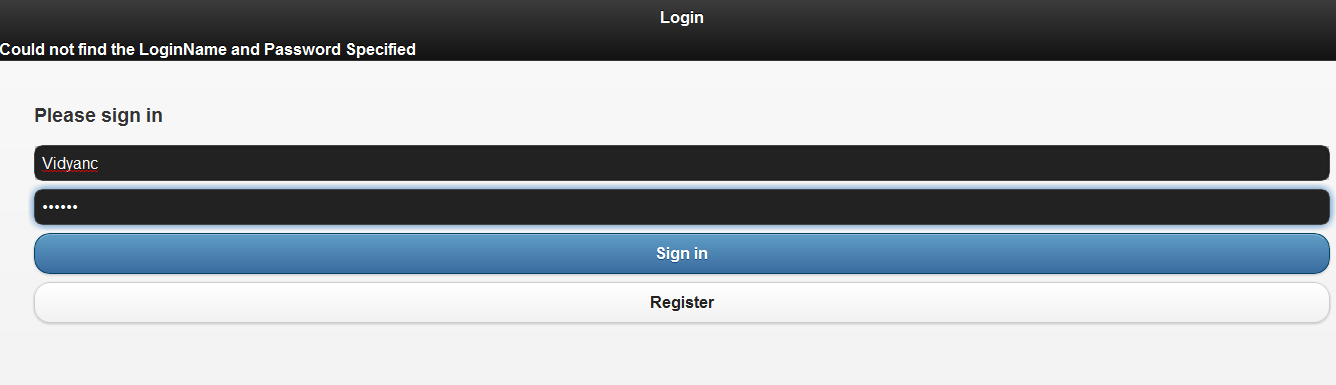


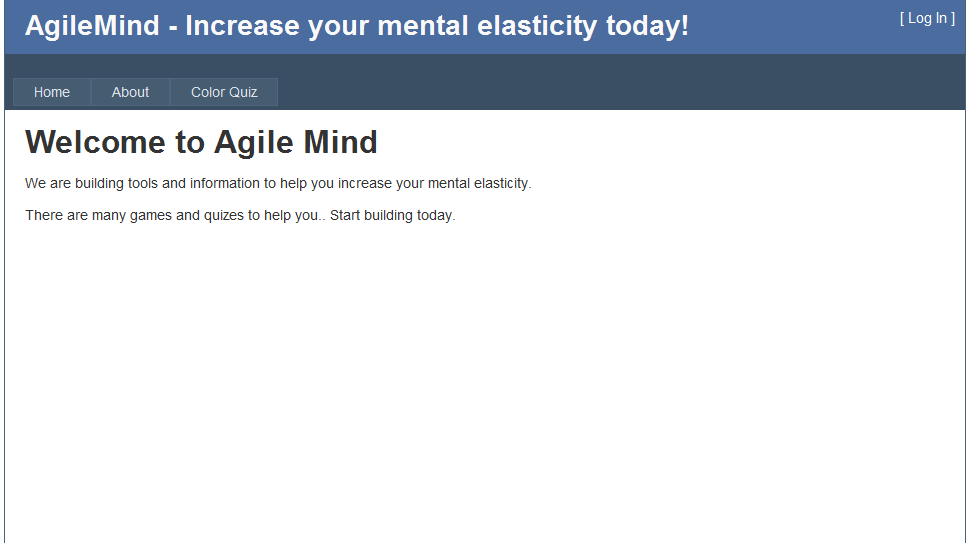
Fig 3: check if Username conforms to length requirement

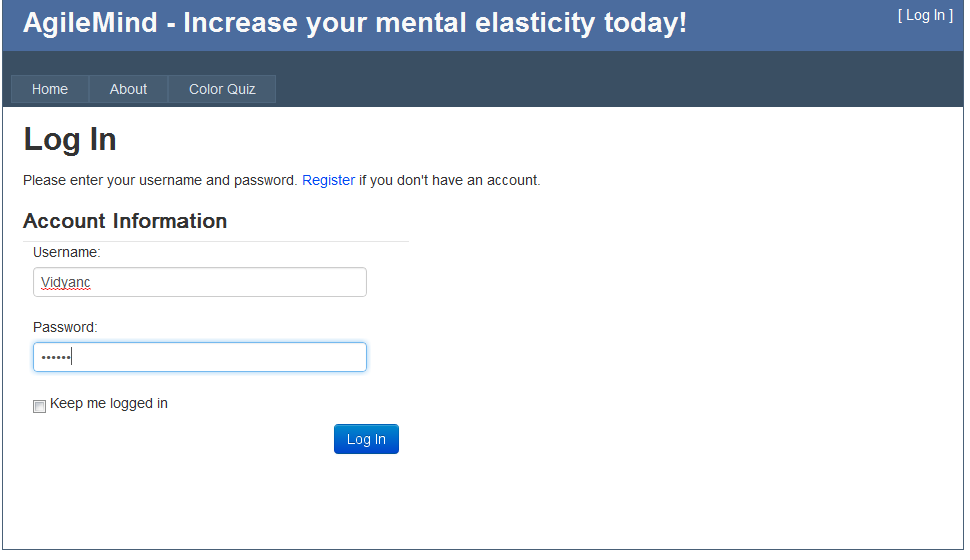
User Interface : Website and mobile user interface views are provided :

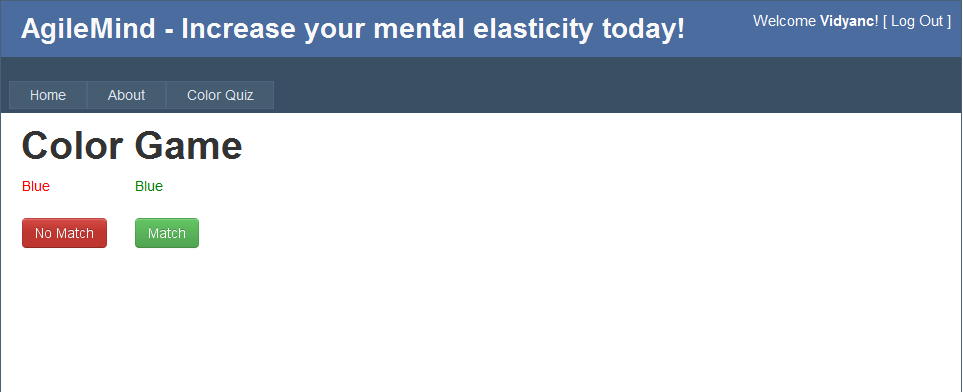


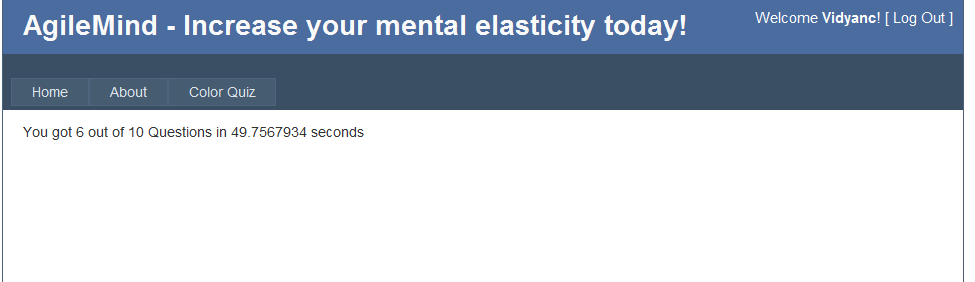


Website :

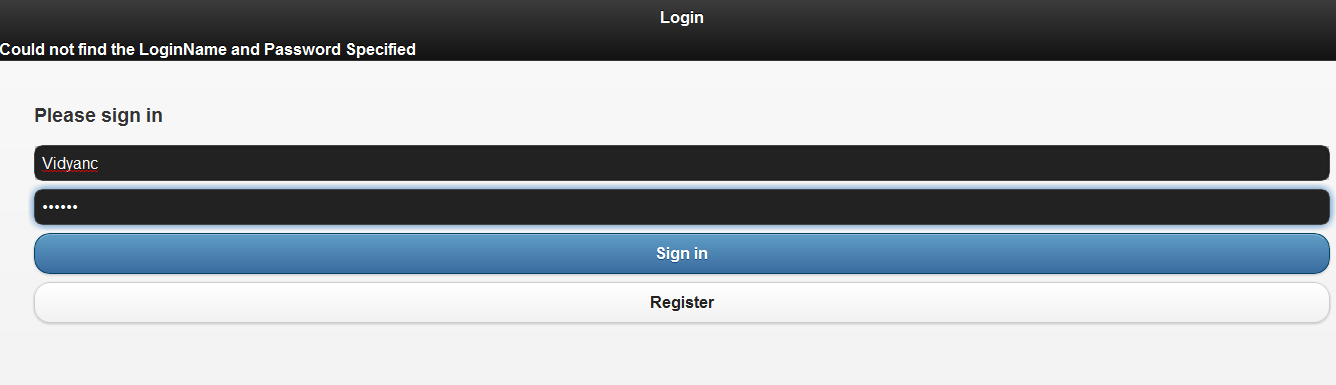


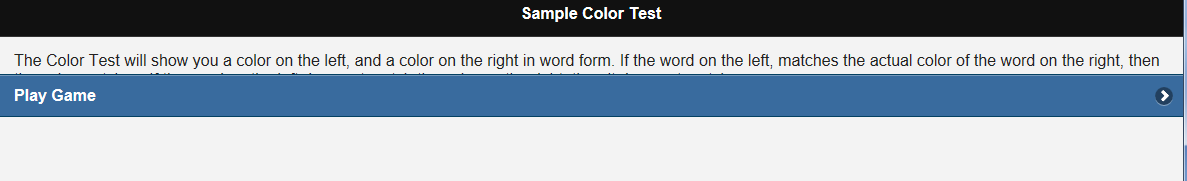


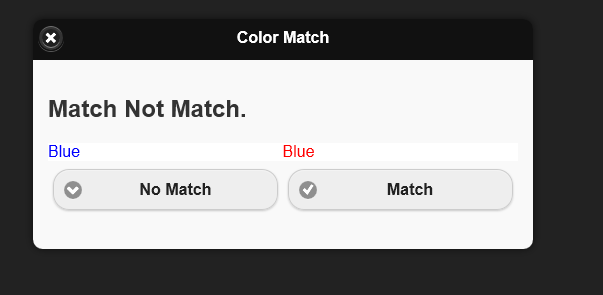




Mobile Site :







# 

# Project Management

**Implementation Status Report**

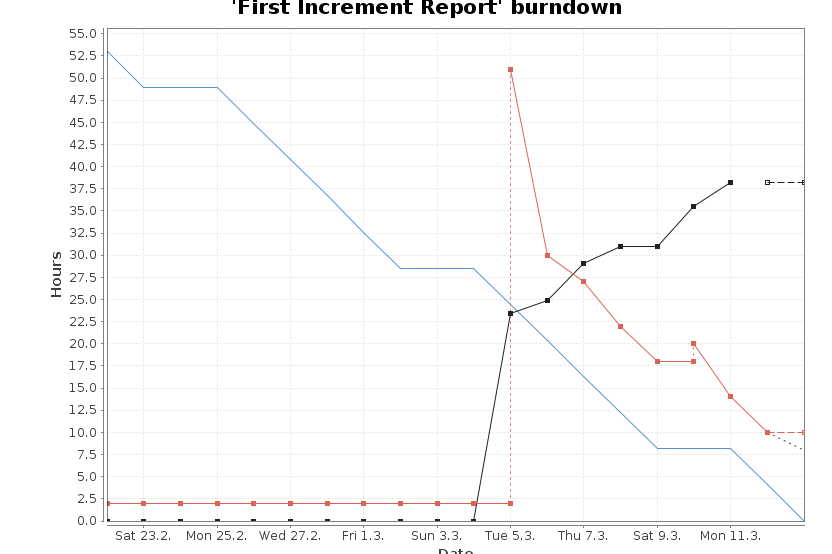
Design and implementation of :

Login Service

Color Match Quiz

**Screenshots**





**Issues/concerns :**

The webservice for posting back scores from the main website currently has no way to validate the information provider.  If a user found the webservice they could post any scores they like for whatever user they want.

A login session key would eliminate this liability.

Since we are using webservices called by Ajax in the mobile site with the users password and login name, a user could view that code, and knowing a username/password could submit scores for that username.  However, since this is a self help program this may not likely be an issue.  No one else's personal data can be revealed since the attacker would still needs the username and password of the user.