COVER PAGE

Tilted Programmers

Kevin Le

Michael Ha

David Luong

David Tran

Jonathan Peng

Stephen Chan

Hot Fix Iteration

Introduction to Software Engineering

Professor Ehteshami

Submitted

5/11/2016

**2. Revision History**

**May 8th 2016**

David Tran: Present

David Luong: Present

Kevin Le: Present

Jonathan Peng: Present

Stephen Chan: Present

Michael Ha: Present

Group discussed updates on the diagrams as well as the small bus that needed to be addressed. Worked on the paragraph that discusses the pricing of our software and also the presentation. Group planned to have document ready by tuesday night and structured the presentation based on each member discussing a feature and a specific screen of the learning tool.

Table of Contents

Content: Page Number:

Revision History...................................................................................................................1

Table of Contents…………………………………………………………………………..2

User Stories Worked On.........................................................................................................3

Use Case Diagram………………………………………………………………………..4

Class Diagram………..........................................................................................................5

System Sequence Diagram...................................................................................................6

Trello Screenshots………………………….....................................................................7-9

Activity Diagram...................................................................................................................10

Test Cases……………………………………………………………………………...….11-12

Automation…………...........................................................................................................12-19

Hot fix User story code……………............ .....................................................................20-22

User manual…..……………………………………………………………………………...22-27

References………………..………………………………………………...…………..…...28

Team Charter............................................................................................................................28-31

Team Evaluation..........................................................................................................................31

Paragraph..................................................................................................................................32

Software Engineering: A Practitioner’s Approach:

Chapter 3: Software Process Structure..................................................................................30-40

Chapter 4: Process Models...................................................................................................40-66

Chapter 5: Agile Development...............................................................................................66-87

Chapter 8: Understanding Requirements............................................................................131-166

Chapter 9: Requirements Modeling: Scenario-Based Methods..........................................166-184

Chapter 10: Requirements Modeling: Class- Based Methods.............................................184-202

**3. User Stories worked on Plus Mockups**

**US-12: Save User’s chapter scores**

As a user, I would like my chapter score saved in order to view it later on.

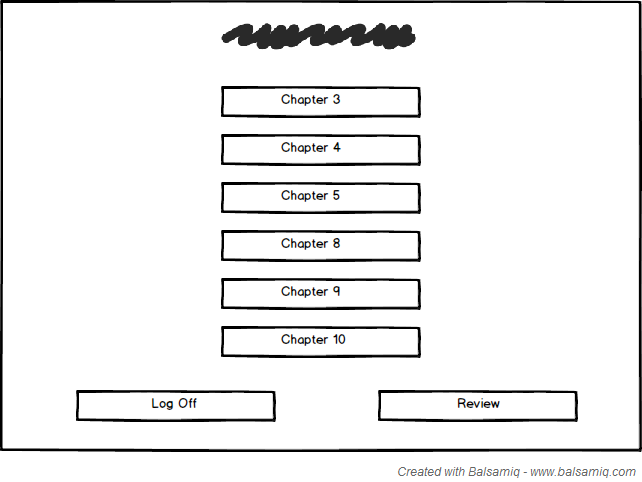
Relation to Use Case: Relates to chapter select screen as well as results screen.

Assigned Developer: Kevin Le, David Tran, Jonathan Peng

Due Date: 4/25/16

Priority: High

MockUp:



**US-13: View User’s chapter scores**

As a user, I would like to view my chapter scores in order to compare them.

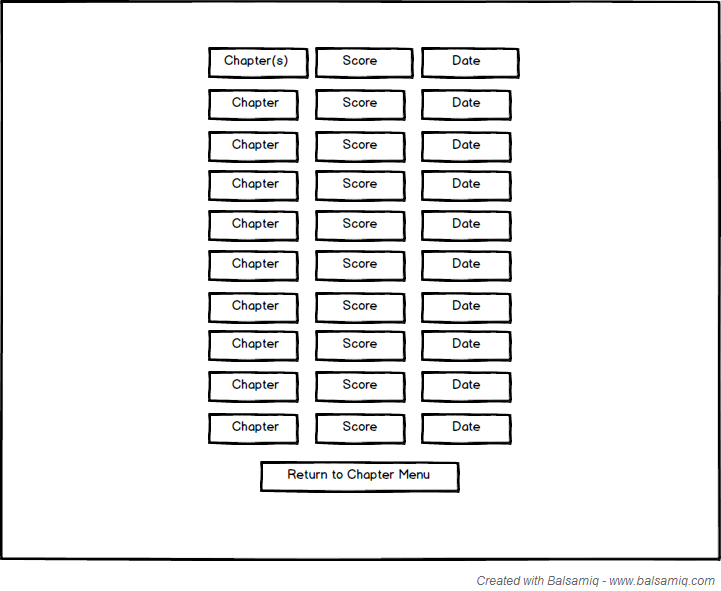
Relation to Use Case: Relates to chapter select screen as well as results screen.

Assigned Developer: Kevin Le, David Tran, Jonathan Peng

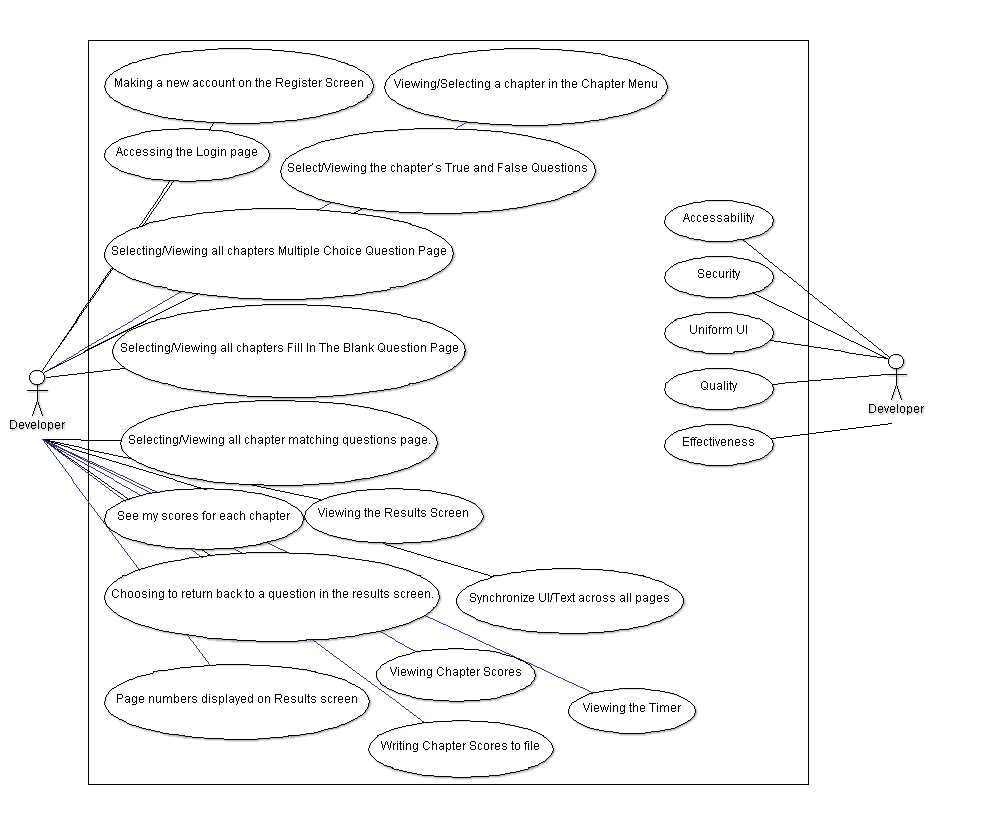
Due Date: 4/25/16

Priority: High

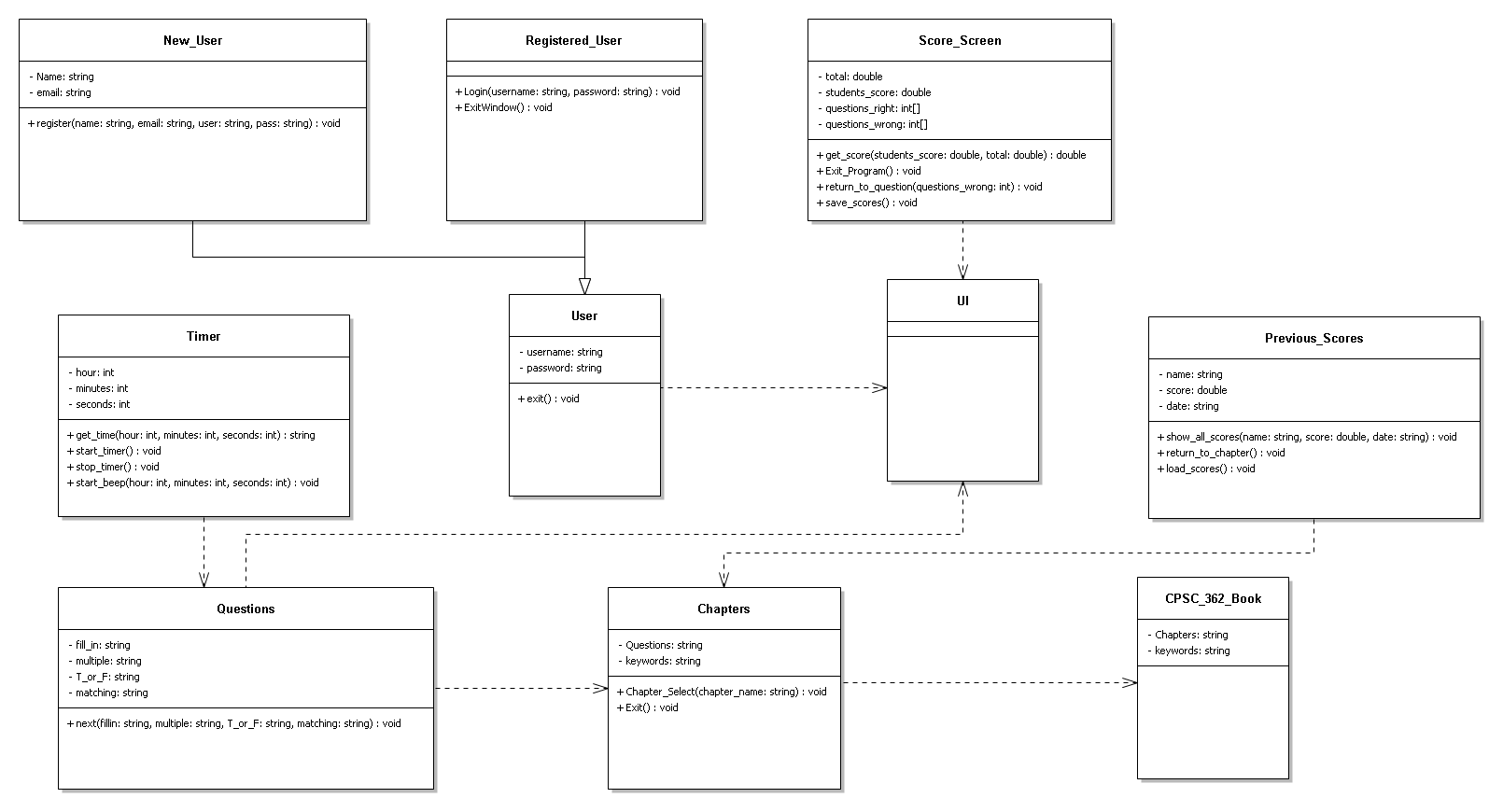
MockUp:



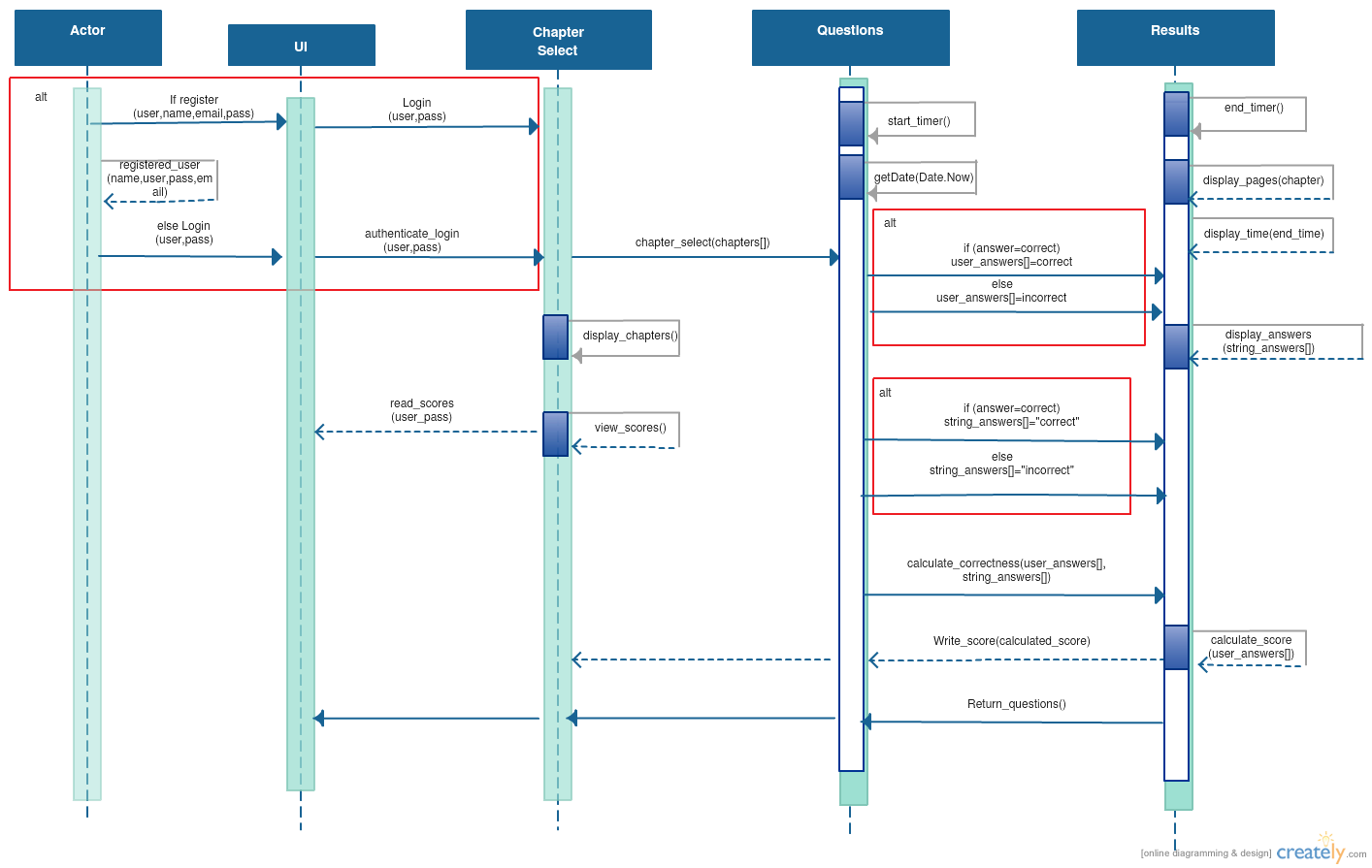
**Use Case Diagram**



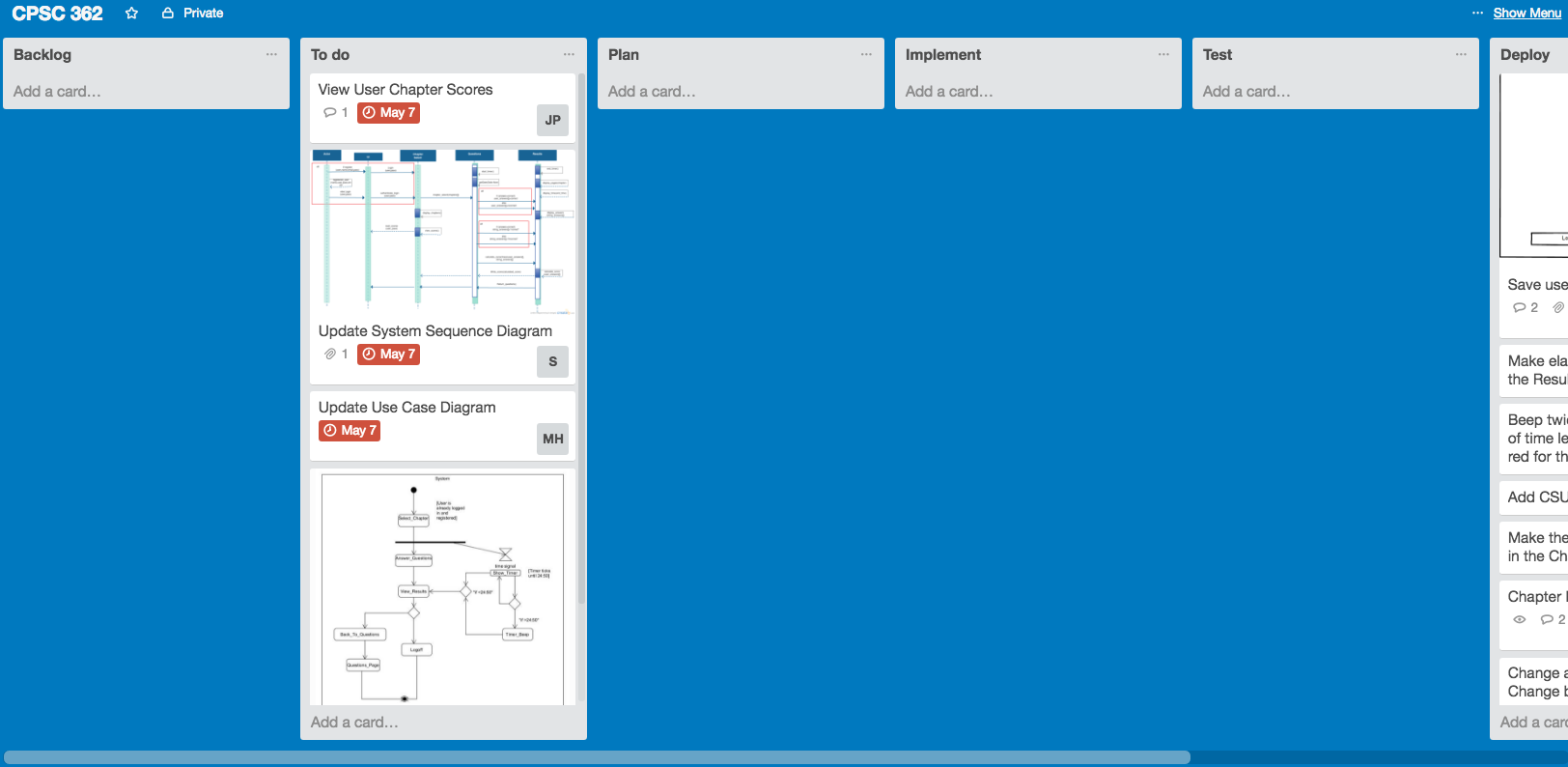
**Class Diagram**

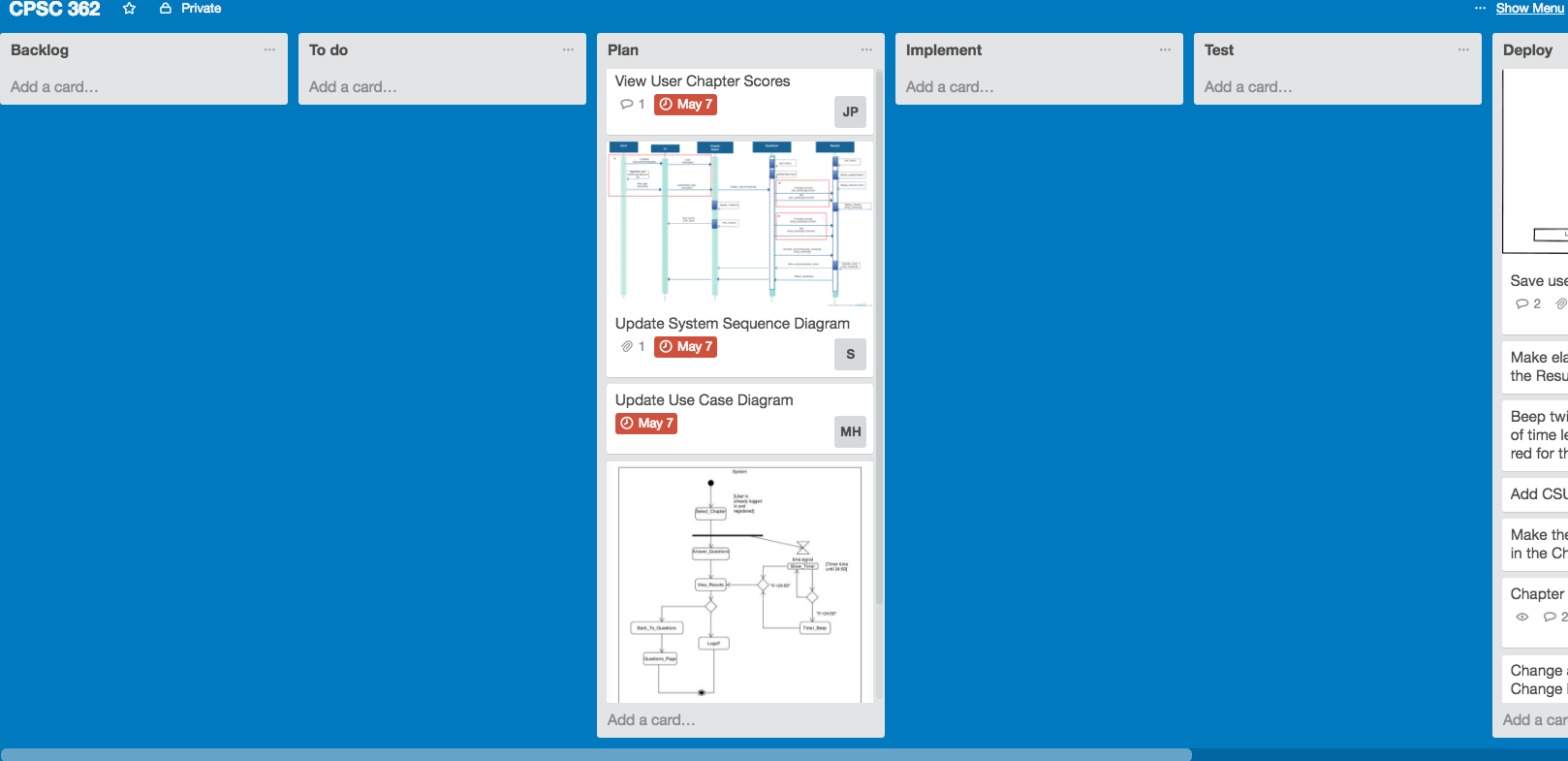


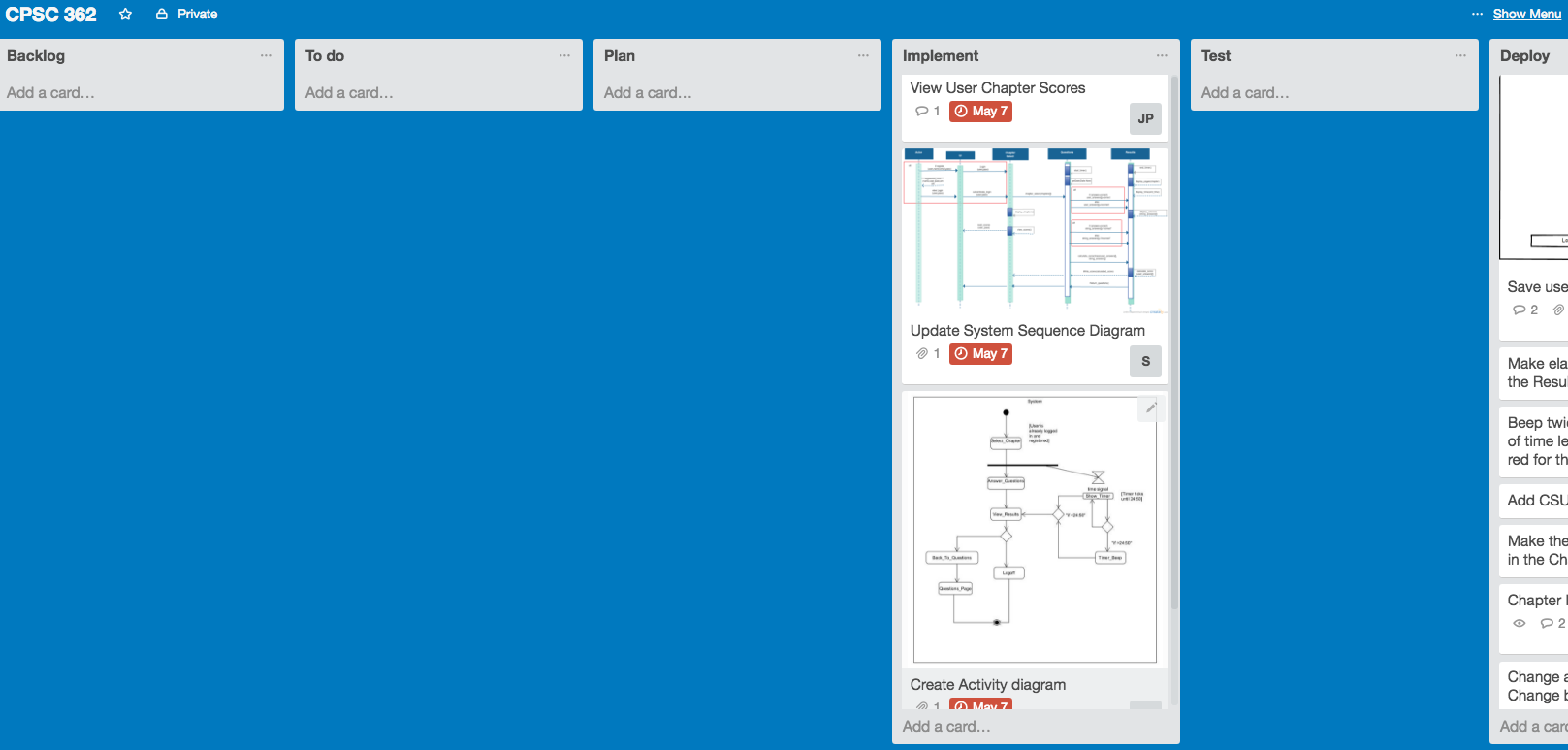
**System Sequence Diagram**

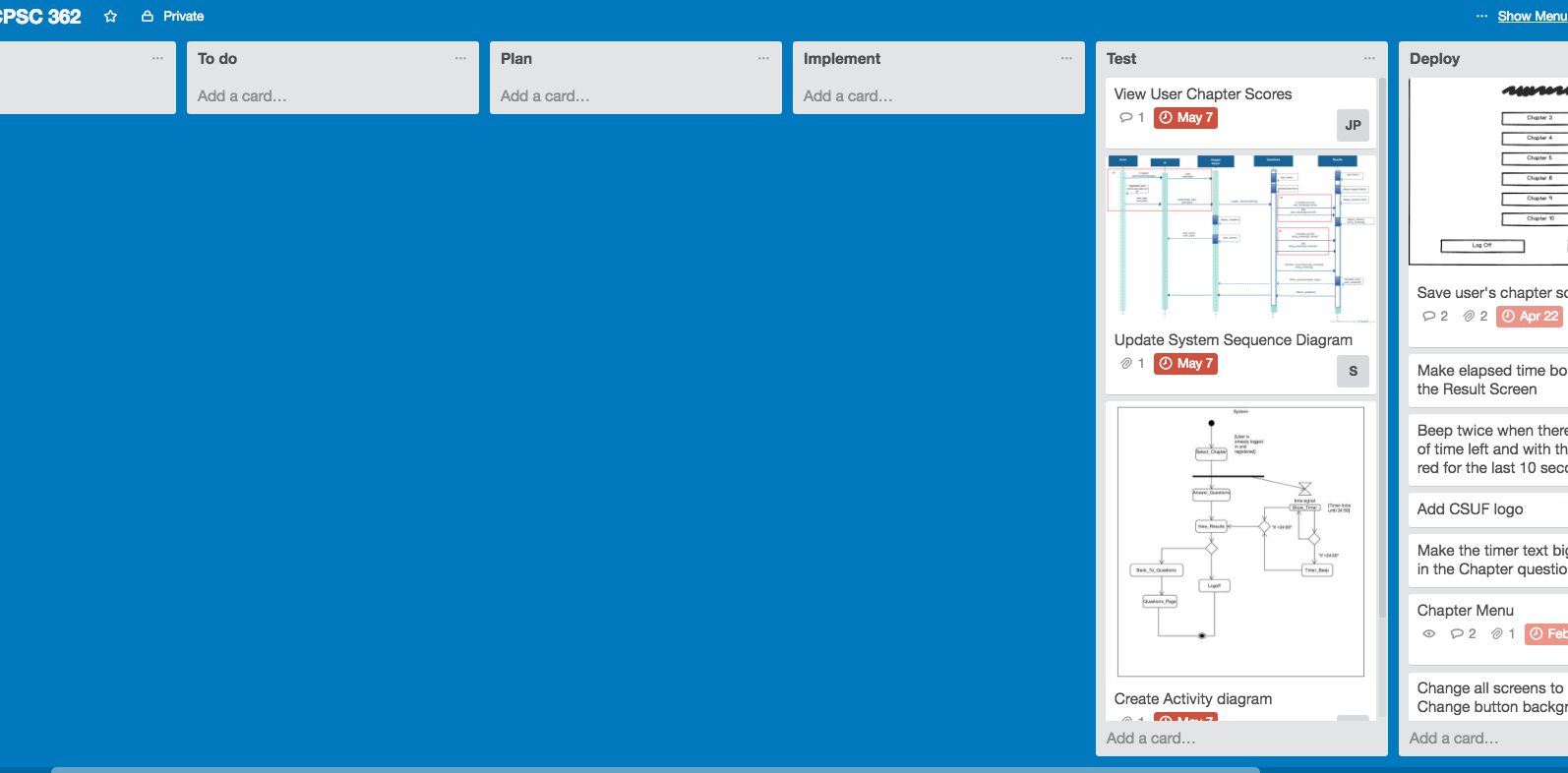


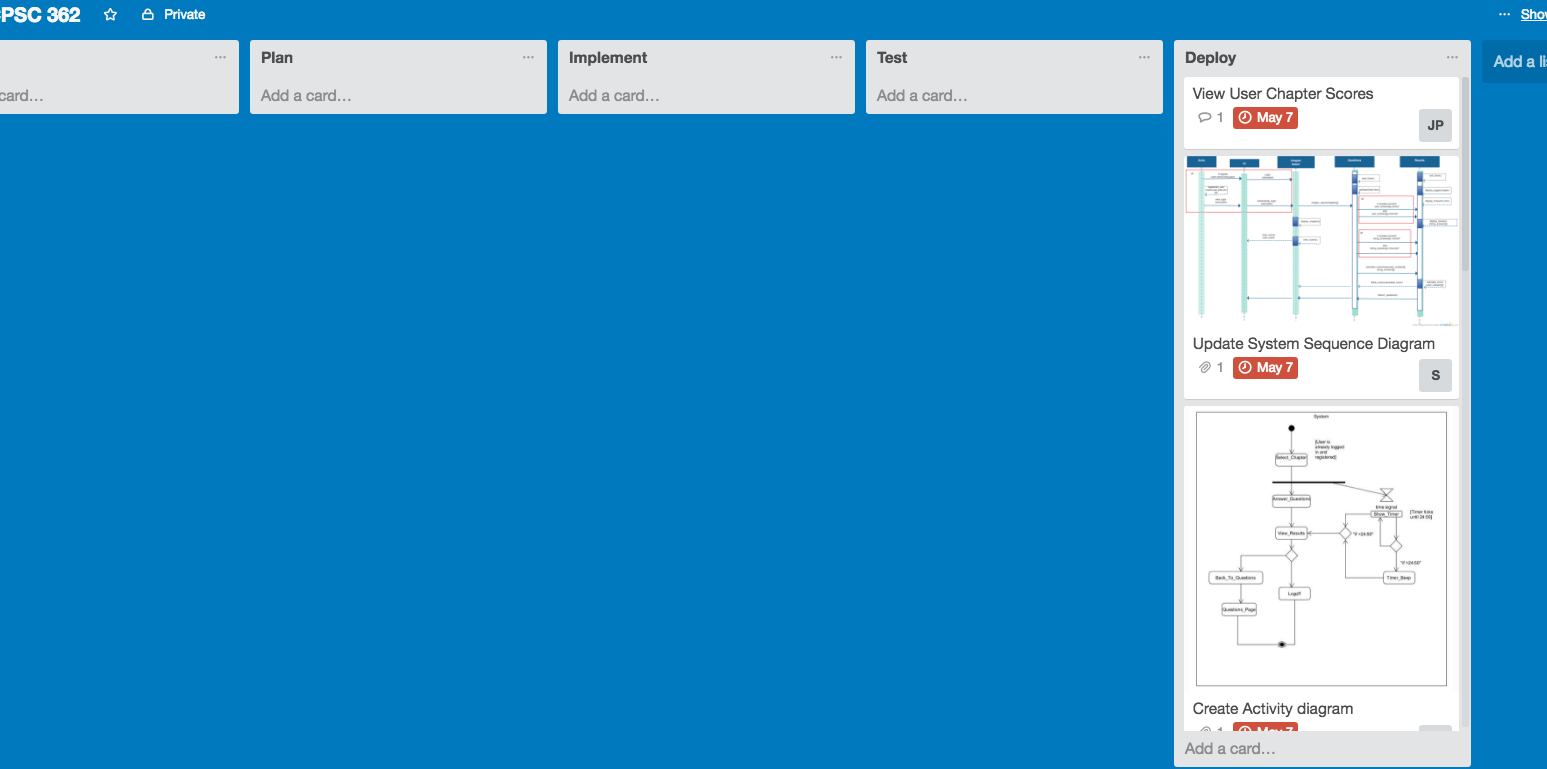
**4. Sprint Backlog Screenshots of your Trello Kanban-Board.**



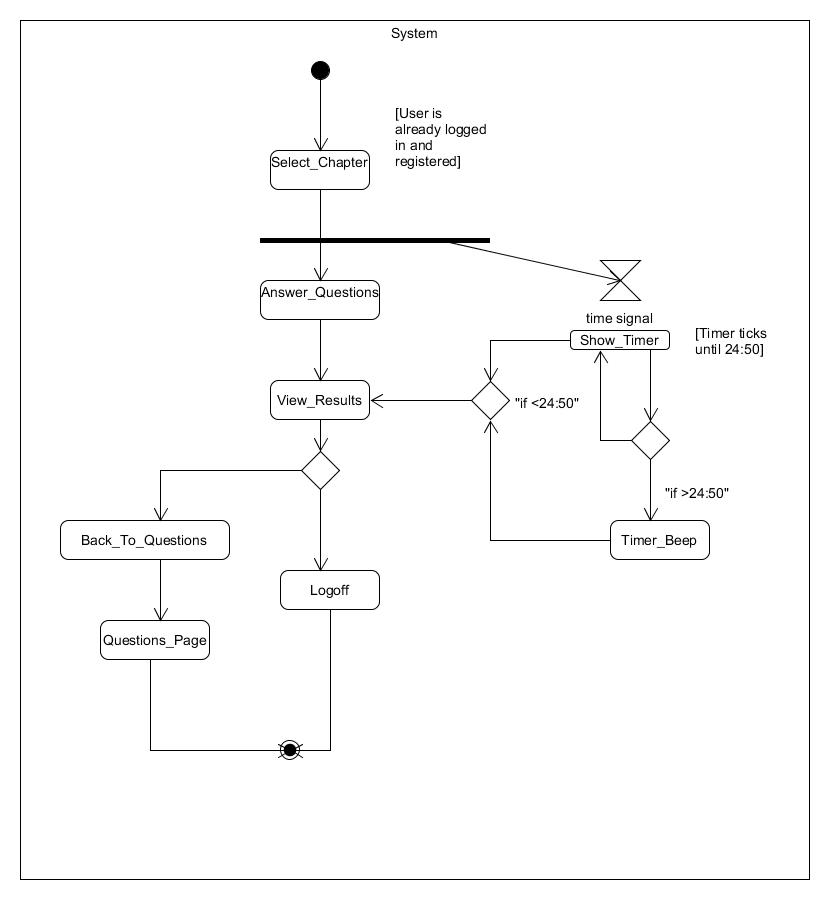








**5. Activity Diagram**



**6. Test Plan, Test suite and Test Cases.**

**Iteration 4 Test Cases**

|  |  |
| --- | --- |
| **Test Designed by: David Tran** | **Module Name: Review page information** |
| **Test Designed date: 5-8-16** | **Test Title: View the date of past chapter quiz scores in the review page for an account** |
| **Test Case ID: 30** | **Description: In the review page, the user is able to view the date on when they took the chapter’s quiz.** |
| **Test Priority (Low/Medium/High): High** | **Pre-conditions: User is logged in, taken a chapter quiz at least once, and is currently on the chapter menu selection screen.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User clicks on the review button.** |  | **User is able to view the dates of their past quiz or quizzes.** | **User is able to view the dates of their past quiz or quizzes.** | **Pass** |  |

**Post-conditions:**

**User is finished with the program.**

|  |  |
| --- | --- |
| **Test Designed by: David Tran** | **Module Name: Result screen information** |
| **Test Designed date: 5-8-16** | **Test Title: View the date of the chapter quiz taken during this session on a different account.** |
| **Test Case ID: 31** | **Description: In the results page, a different account user is able to view the date of the quiz taken during their session.** |
| **Test Priority (Low/Medium/High): High** | **Pre-conditions: User is logged in and is on the chapter menu.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User selects the chapter to take a quiz on.** |  | **The quiz loads for that specific chapter and the user would complete it.** | **The quiz loads for that specific chapter and the user would complete it.** | **Pass** |  |
| **2** | **User completes the chapter’s quiz.** |  | **When the quiz ended, the user would be redirected to the result screen where they can view all the information including the date.** | **When the quiz ended, the user would be redirected to the result screen where they can view all the information including the date.** | **Pass** |  |

**Post-conditions:**

**User is finished with the program.**

**7. Automation For Iteration 4 Test Cases**

**Beeper Sound Automation**

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using System.Windows.Input;

using System.Windows.Forms;

using System.Drawing;

using Microsoft.VisualStudio.TestTools.UITesting;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Microsoft.VisualStudio.TestTools.UITest.Extension;

using Keyboard = Microsoft.VisualStudio.TestTools.UITesting.Keyboard;

namespace CodedUITestProject1

{

/// <summary>

/// Summary description for CodedUITest1

/// </summary>

[CodedUITest]

public class CodedUITest1

{

public CodedUITest1()

{

}

[TestMethod]

public void CodedUITestMethod1()

{

// To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

this.UIMap.makeBeepingSound();

}

#region Additional test attributes

// You can use the following additional attributes as you write your tests:

////Use TestInitialize to run code before running each test

//[TestInitialize()]

//public void MyTestInitialize()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

////Use TestCleanup to run code after each test has run

//[TestCleanup()]

//public void MyTestCleanup()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

#endregion

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

public TestContext TestContext

{

get

{

return testContextInstance;

}

set

{

testContextInstance = value;

}

}

private TestContext testContextInstance;

public UIMap UIMap

{

get

{

if ((this.map == null))

{

this.map = new UIMap();

}

return this.map;

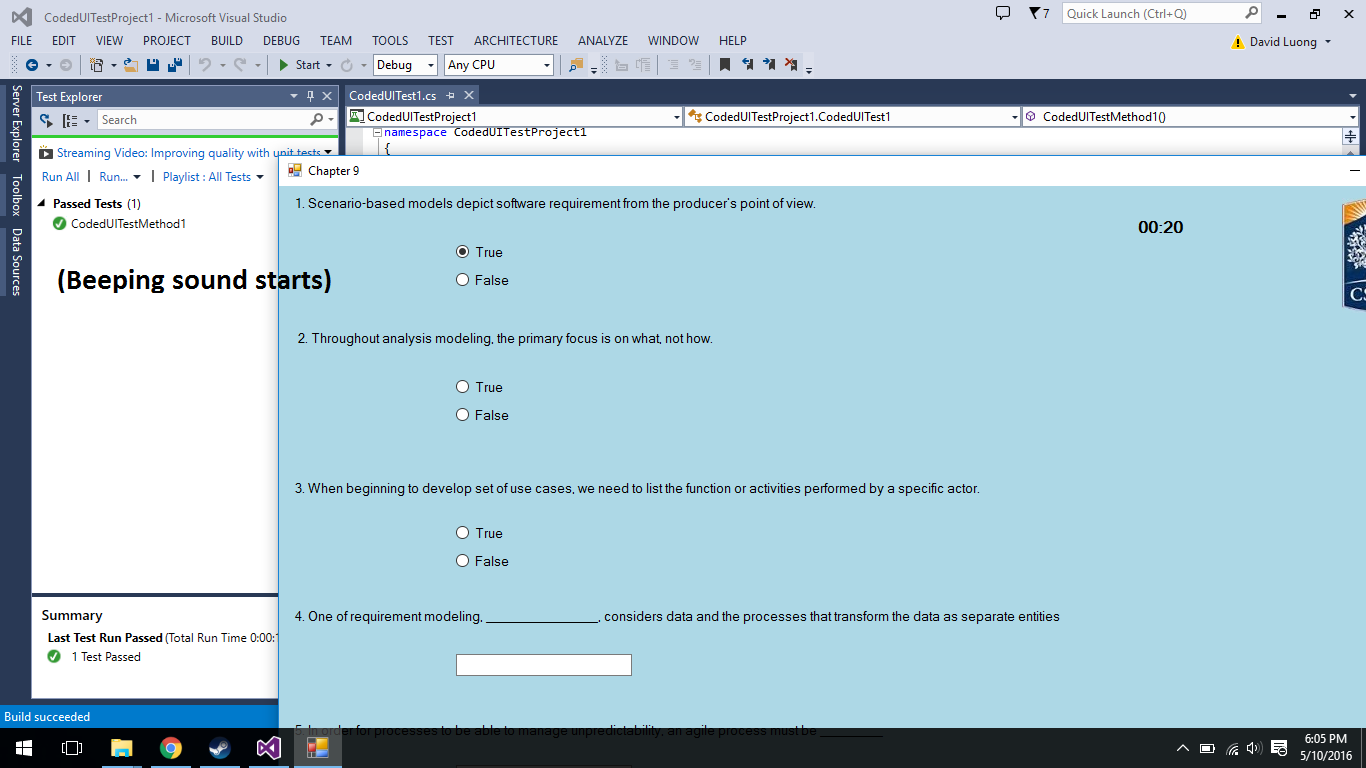
}

}

private UIMap map;

}

}



**Red Flashing Timer Automation Test**

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using System.Windows.Input;

using System.Windows.Forms;

using System.Drawing;

using Microsoft.VisualStudio.TestTools.UITesting;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Microsoft.VisualStudio.TestTools.UITest.Extension;

using Keyboard = Microsoft.VisualStudio.TestTools.UITesting.Keyboard;

namespace CodedUITestProject1

{

/// <summary>

/// Summary description for CodedUITest1

/// </summary>

[CodedUITest]

public class CodedUITest1

{

public CodedUITest1()

{

}

[TestMethod]

public void CodedUITestMethod1()

{

// To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

this.UIMap.redFlashingTimer();

}

#region Additional test attributes

// You can use the following additional attributes as you write your tests:

////Use TestInitialize to run code before running each test

//[TestInitialize()]

//public void MyTestInitialize()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

////Use TestCleanup to run code after each test has run

//[TestCleanup()]

//public void MyTestCleanup()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

#endregion

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

public TestContext TestContext

{

get

{

return testContextInstance;

}

set

{

testContextInstance = value;

}

}

private TestContext testContextInstance;

public UIMap UIMap

{

get

{

if ((this.map == null))

{

this.map = new UIMap();

}

return this.map;

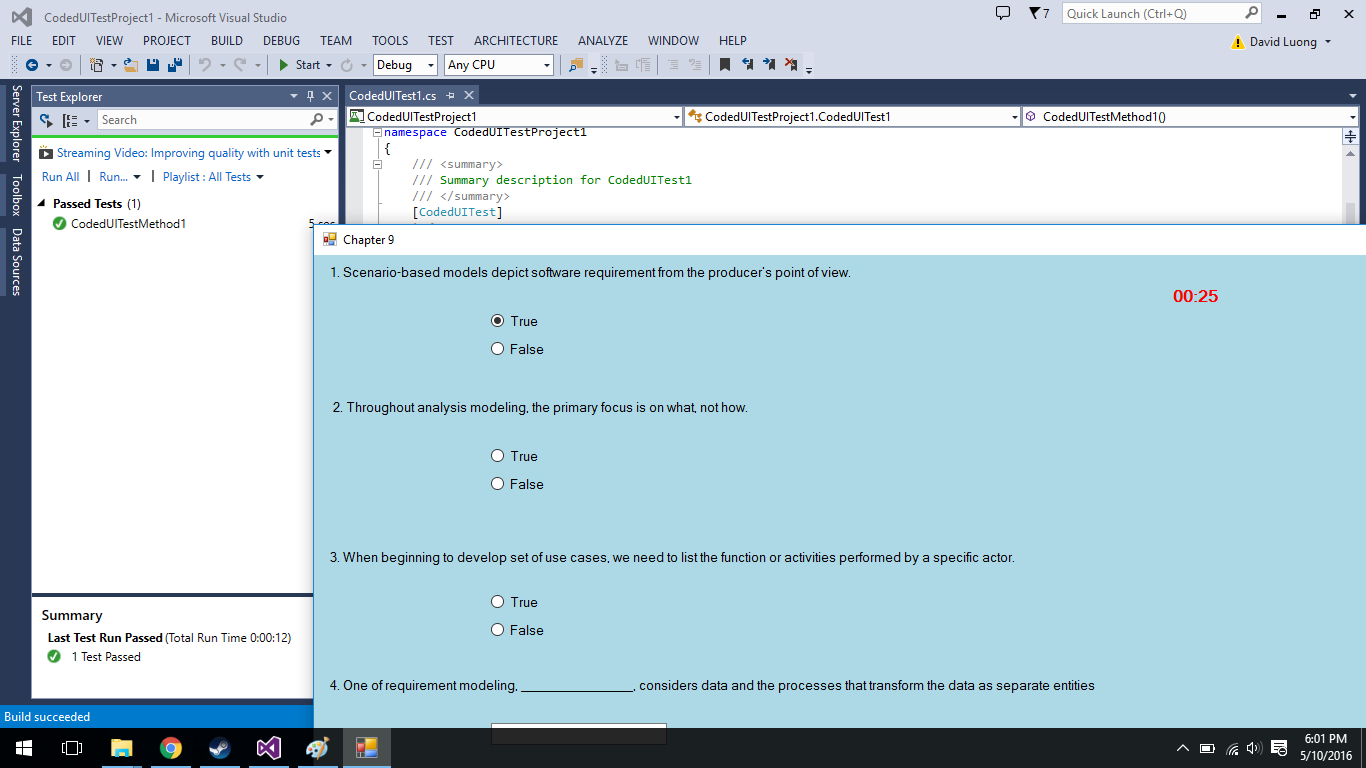
}

}

private UIMap map;

}

}



**Review Score Automation**

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using System.Windows.Input;

using System.Windows.Forms;

using System.Drawing;

using Microsoft.VisualStudio.TestTools.UITesting;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Microsoft.VisualStudio.TestTools.UITest.Extension;

using Keyboard = Microsoft.VisualStudio.TestTools.UITesting.Keyboard;

namespace CodedUITestProject1

{

/// <summary>

/// Summary description for CodedUITest1

/// </summary>

[CodedUITest]

public class CodedUITest1

{

public CodedUITest1()

{

}

[TestMethod]

public void CodedUITestMethod1()

{

// To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

this.UIMap.reviewScore();

}

#region Additional test attributes

// You can use the following additional attributes as you write your tests:

////Use TestInitialize to run code before running each test

//[TestInitialize()]

//public void MyTestInitialize()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

////Use TestCleanup to run code after each test has run

//[TestCleanup()]

//public void MyTestCleanup()

//{

// // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the menu items.

//}

#endregion

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

public TestContext TestContext

{

get

{

return testContextInstance;

}

set

{

testContextInstance = value;

}

}

private TestContext testContextInstance;

public UIMap UIMap

{

get

{

if ((this.map == null))

{

this.map = new UIMap();

}

return this.map;

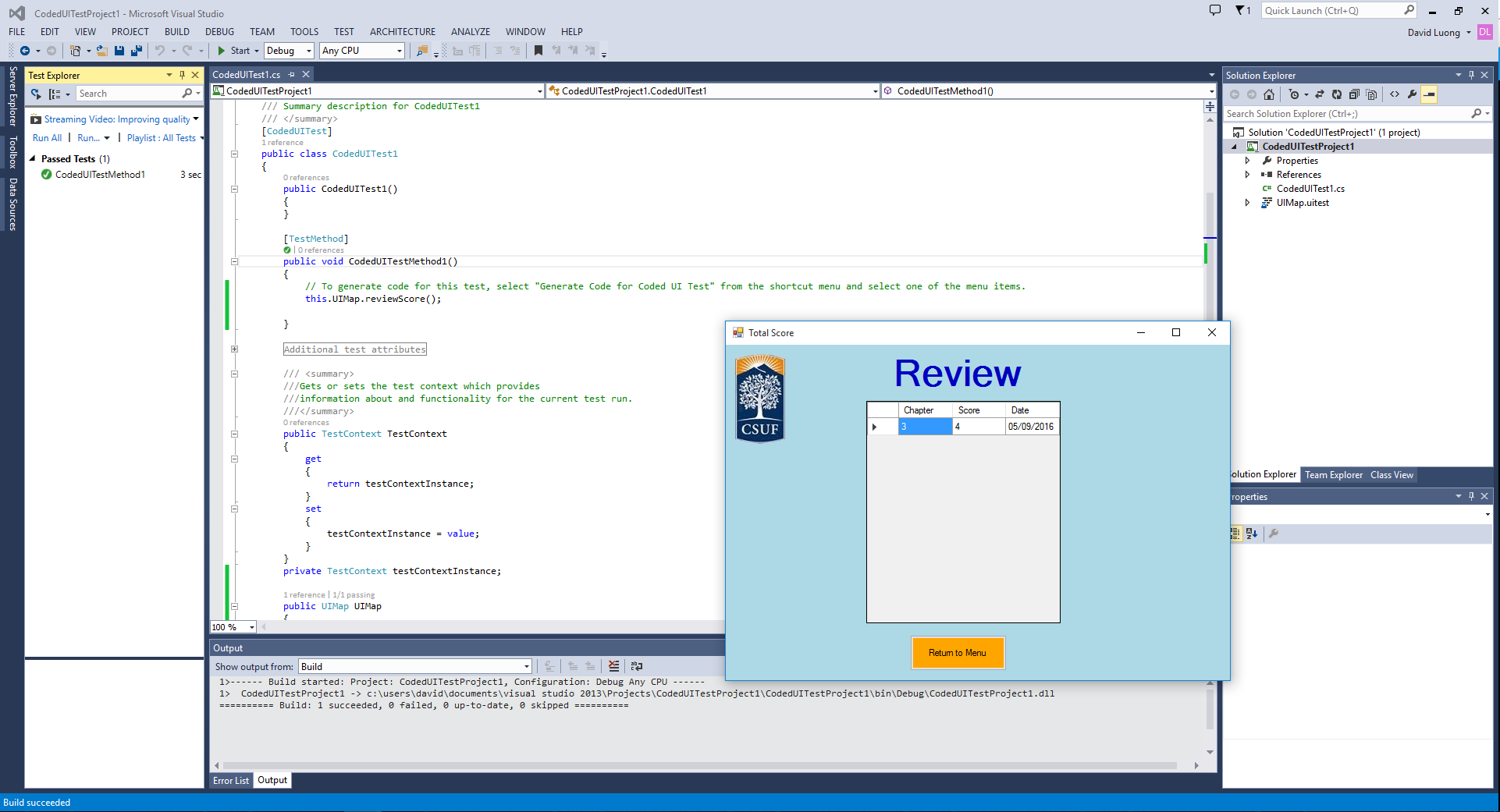
}

}

private UIMap map;

}

}



**8. Code for Hotfix user story.**

**Read the scores**

string[] separators = { "|", "." };

//Read all the users and their name into list

StreamReader sr = new StreamReader("score.txt");

while ((line = sr.ReadLine()) != null)

{

list.Add(line);

}

//count number of index in list

people\_list=list.Count();

string ch; //chapter

string chp\_score; //chapter score

string date\_taken; //The date of when test was taken

for (int i = 0; i < people\_list; ++i)

{

//In the word array, it will contain a username and his or her score

string[] words = list[i].Split(separators, StringSplitOptions.RemoveEmptyEntries);

//The first value in word[] is the username, so when the username is match to the

//username inputted from the login screen, then the datagrid will output the user's

//past test score

if (words[0] == user)

{

//This count how many chapter scores are in the words array

int m = words.Count();

//Input the chapter and the chapter score into the DataGrid

for (int j = 1; j < m; ++j)

{

ch = words[j];

++j;

chp\_score = words[j];

++j;

date\_taken = words[j];

TotalGrid.Rows.Add(ch, chp\_score, date\_taken);

}

}

}

sr.Close();

**Write the scores**

//String of the current date

InputDate = DateTime.Now.ToString("MM/dd/yyyy");

string[] separators = { "|", "." };

//The check was use as a counter when user return back to Result Screen

//after viewing the correct answer, the program will not add the score

//again.

if (check == 0)

{

//Will write the new score into the score.txt score = System.IO.File.ReadAllLines("score.txt");

//Find how many in the textfile

string line = "";

StreamReader sr = new StreamReader("score.txt");

int counter = 0;

while ((line = sr.ReadLine()) != null)

{

++counter;

}

score = new String[counter];

sr.Close();

//Copy every line into array

StreamReader sr2 = new StreamReader("score.txt");

int counter2 = 0;

while ((line = sr2.ReadLine()) != null)

{

score[counter2] = line;

++counter2;

}

sr2.Close();

people\_list = score.Count();

//change the chapter from int to string

string chp = c.ToString();

//change the number of correct to string

string chapter\_score = \_correct.ToString();

//This check if there are any users in the text file

if (people\_list > 0)

{

//This function add new name, chapter or score to the text file contain

//the record of the user's testing score

for (int i = 0; i < people\_list; ++i)

{

string[] split = score[i].Split(separators, StringSplitOptions.RemoveEmptyEntries);

//The first value in word[] is the username, so when the username is match to the

//username inputted from the login screen, then the datagrid will output the user's

//past test score

if (split[0] == user)

{

string new\_score = chp + "." + chapter\_score + "." + InputDate + "|";

score[i] = score[i].Replace(user + "|", user + "|" + new\_score);

break;

}

//If the user doesn't have score in text file containing the score (score.txt) then add the name, chapter and score

else if (i == people\_list - 1 && split[0] != user)

{

new\_tester = user + "|" + chp + "." + chapter\_score + "." + InputDate + "|";

break;

}

}

}

//If there are no users in the textfile just add it in

else

{

new\_tester = user + "|" + chp + "." + chapter\_score + "." + InputDate + "|";

}

sr.Close();

//This would write back all the name, chapter and score back to the text file

StreamWriter sw = new StreamWriter("score.txt");

for (int i = 0; i < score.Length; ++i)

{

sw.WriteLine(score[i]);

}

//If the user have not yet take the test, then this function will write the user

//name, chapter and score in the text file

if (new\_tester != "")

{

sw.WriteLine(new\_tester);

}

sw.Close();

}

**9. User manual – Screen shots of your working product with explanation on How to use your system.**

**Registration Screen**



1. If you are not a member, then

1. Enter your full name

2. Enter your E-mail

3. Enter a username

a. Username cannot be the same as existed one

4. Enter a password

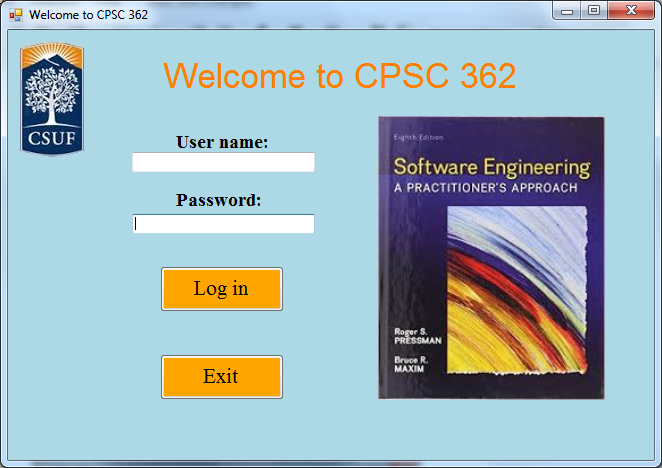
a. Need to be more than 8 characters

5. Click on the “Register” button to become a member

2. If you are a member, then

1. Click on the “Member Login” button to go to Login screen

**Login Screen**



If users want to go to Chapter Menu Screen

1. Enter your User name

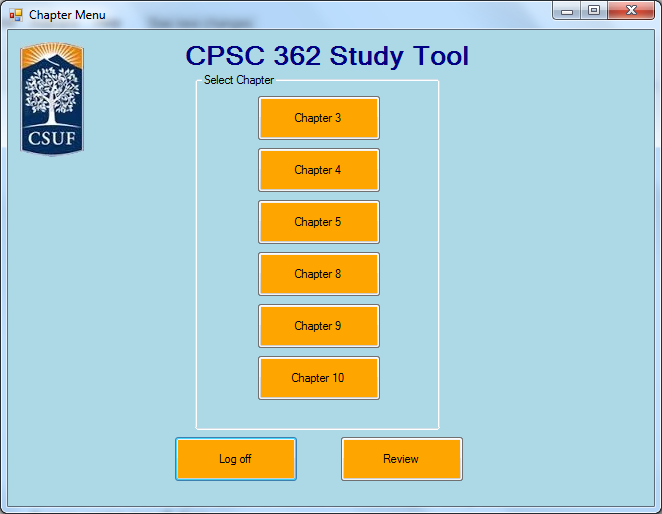
2. Enter your Password

3. Click on “Log in” button

If users want to exit the program, then

1. Click on the “Exit” button

**Chapter Menu Screen**



If users want to start answer the questions on study tool,

1. Click one of the chapter button to start the selected chapter study tool

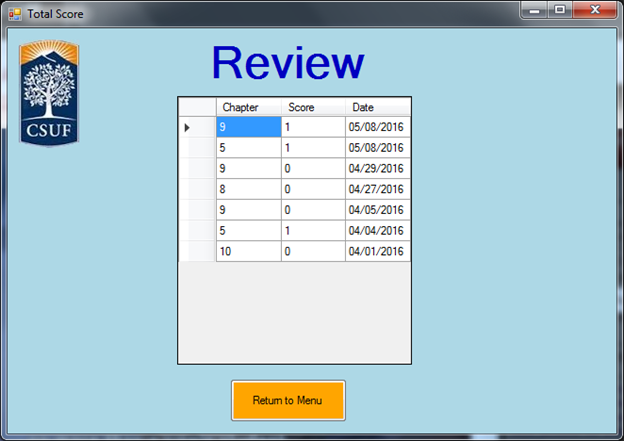
If users want to log off, then

1. Click the “Log off” button to log off the account.

If users want to see all their scores that they have gotten before, then

1. Click the “Review” button

**Review Screen**

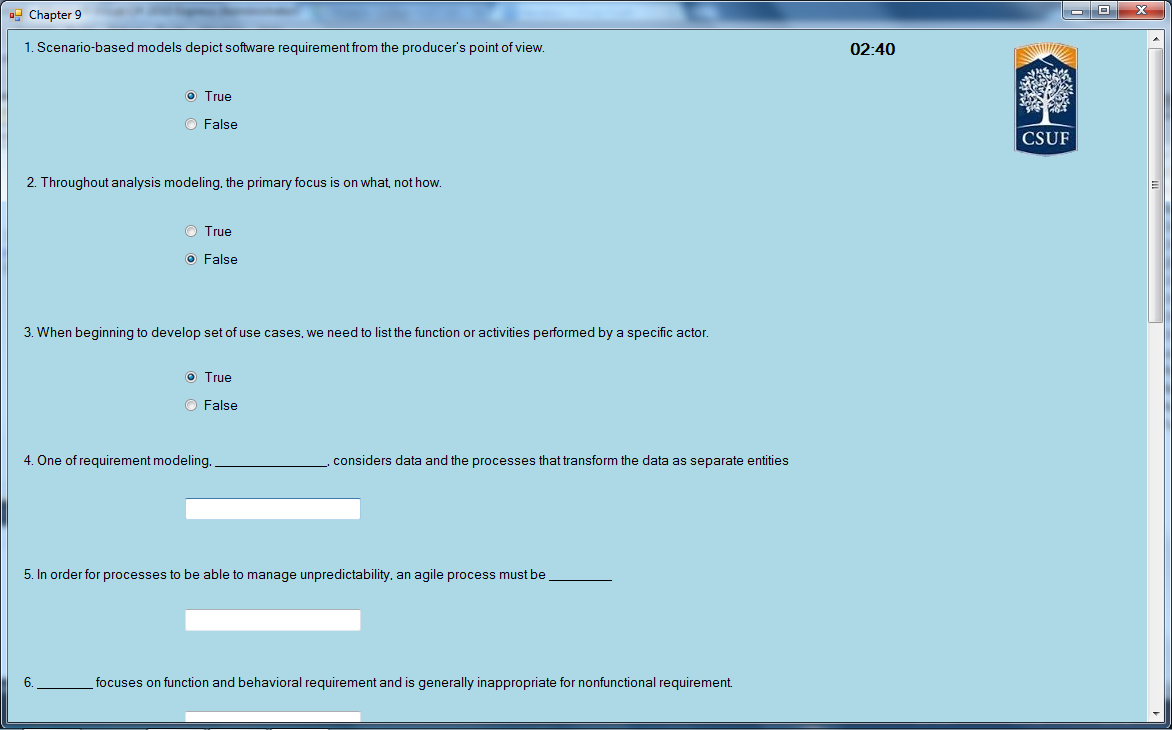


1. The users can see all their previous scores taken from the newest to oldest or if they want to view from oldest to newest just click on the “Date” tab.

If users want to return to the Chapter Screen, then

1. Click the “Return to Menu” button

**Chapter Screen**



1. The users will be given 25 minutes to answer the given questions.

2. When the timer reaches 24:50, there will be beeping sounds, and the timer will start flashing red to signal users that they only have 10 seconds left.

3. If the timer reaches 25 minutes, then it will automatically take the users to the Result Screen.

4. Click “Finish” button to submit the answers and go to Result Screen

5. Users can click “Back to menu” button to go back to Chapter Menu Screen

**Result Screen**



1. Users can see if they answered their questions correctly and what page they can find the answers to the question. They can see the score and percentage they got for the current chapter they are answering.

2. Users can click on “Back to Question” button to go back to the chapter questions screen with answers shown.

3. Users can click on “Log off” button to log off their account and go to the log in screen.

**10. References (list references here, and cite them in appropriate places in the report)**

Software Engineering: A Practitioner’s Approach by Roger Pressman

UML Distilled Third Edition by Martin Fowler

**11. Team Charter (in the given format)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Title** | CPSC 362 |  | All team members participated in the creation of this charter and agree with its content. **Date** 01/28/2014 |
| **Instructor** | Yasamin Ehteshami |  |
| **Course Dates** | MW 5:00-6:50 |  |

**Team Members** (Contact Information)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address (city, state, country) | Phone | Cell | Email |
| Stephen Chan | La Mirada, CA | 562-650-2700 |  | stephenchan@csu.fullerton.edu |
| David Tran | Anaheim, CA | 714-487-2653 |  | DavidTran794@csu.fullerton.edu |
| Michael Ha | Westminster, CA | 714-725-9569 |  | Mha94@csu.fullerton.edu |
| Kevin Le | Westminster, CA | 714-925-3210 |  | lekevin42@csu.fullerton.edu |
| Jonathan Peng | Chino Hills, CA | 909-348-4201 |  | jspeng@csu.fullerton.edu |
| David Luong | Santa Ana, CA | 714-360-3083 |  | davidluong@csu.fullerton.edu |

**Team Member Skill Inventory** (Areas individual members can contribute)

|  |  |
| --- | --- |
| Stephen Chan | § C++,SQL,Python |
| David Tran | § C++, Python, HTML5, SQL |
| Michael Ha | § C++, Python, Assembly |
| Kevin Le | § C++, python, C, Assembly |
| Jonathan Peng | § C++,Assembly, Java, C# |
| David Luong | § C++, Java, Python, CSS, HTML server management and hosting experience. |

**Team Goals** (Project goals, team process goals, quality goals, etc.)

|  |
| --- |
| § Efficient Application, Finish a few days before deadline. Program will compile without errors. |

**Team Roles** (Define roles of each member to achieve goals)

|  |  |
| --- | --- |
| Stephen Chan | § Scrum Master, Recorder, Developer |
| David Tran | § Developer, Tester |
| Michael Ha | § Developer, tester |
| Kevin Le | § Developer, tester |
| Jonathan Peng | § Developer, tester |
| David Luong | § Developer, Tester |

**Ground Rules** (Meeting schedule/locations, attendance expectations, agenda, assignment completion, communication methods, etc.)

|  |
| --- |
| § Group will meet and discuss project in Google Hangout sessions.  § All team members must be punctual and prepared for each team meeting.  § Participation and input is expected from all team members. All opinions will be considered and equally valued.  § The team will meet at least once each week via chat or conference call to discuss current and upcoming projects or assignments (tentatively scheduled for every Saturday @ 8:30 pm).  § Team members will notify the lead in advance if they are not going to be able to attend a scheduled meeting.  § Team members should check email at least once a day to stay on top of things.  § Team members should reply to email within 24 hours.  § Team members will turn in team assignments no later than two weeks prior to the due date.  § All team members will be held accountable for their portions of the projects and are expected to complete them in a timely manner and doing the best job they can.  § Notify team of emergencies that may result in not being able to meet deadlines or meetings. The rest of the team will do their best to pitch in on the team assignment.  § The team must maintain open, clear, and effective communication at all times.  § Assist fellow team members when they are in need.  Team will collectively decide when to meet.  Team will not form alliances or teams against one another.  § Maintain a positive, honest, and open atmosphere by respecting other members’ suggestions, using constructive criticism, and encouragement. |

**Time Commitments/Availability** (Pacific Time)

|  |  |
| --- | --- |
| Stephen Chan | § Friday ALL DAY, Weekend(free) |
| David Tran | § Friday(afternoon-night), Sunday(Morning,Night) |
| Michael Ha | § All day fri-sun, mon-thurs anytime except 5-9pm |
| Kevin Le | § Friday - Sunday |
| Jonathan Peng | § Weekend(free) |
| David Luong | § Everyday(morning) |

**Conflict Management** (What are potential conflicts that might arise among or between team members during this course? How will team members deal with these and other conflicts?)

|  |
| --- |
| Scrum Master will decide course of action.  § In order to avoid conflict clear roles and responsibilities must be assigned, so that there is no confusion.  § If a team member is not performing, the team lead will speak to the member and try and resolve the issue.  § If conflicts arise, please bring them up to the whole team so that everybody can help to resolve the issue in a peaceful and harmonious manner.  § All team members must settle conflicts within the group as quickly as possible. |

**Risk Management** (What are potential barriers to the achievement of these goals?)

|  |
| --- |
| Project Must be in a programming language all developers are comfortable with.  § Scrum Master and Recorder will be in charge of managing  § Any issues between the Scrum Master and the Recorder will be resolved by involving the entire group  § List risks that are chances or possibilities of suffering loss or danger in the project.   * Computer breaks.   + Solution: Make sure to backup files on USB or email * Files are lost for any reason.   + Solution: Upload to Google Drive. * The possibility that we will not finish project on time.   + Solution: Make sure to keep an eye on progress made throughout the semester, set deadlines * Arguments that threaten the group project   + Solution: Make sure to address everyone’s concerns before moving forward |

**Team Evaluation Criteria** (List evaluation criteria that will be used to evaluate team members objectively.)

|  |
| --- |
| * Response time for emails, texts. * Updates on current progress of assignments. * Whether they’re able to attend meetings. * Actual completion of project components. * Evaluate performance based upon efficient code and time of submission. |

**12. Team Evaluation (in the given format).**

**13. Paragraph**

For our software, we’re pricing it at $10. There is a timer that beeps twice in the last 10 seconds at 24:50 indicating to the user that there is only 10 seconds and the program will advance to the next screen. We have also implemented a saving chapter scores section in order for our user to review previous scores they have taken for the chapter. This allows the user to get a clearer view of where they stand in regards to the other chapters. We created the ability for users to create multiple unique accounts allowing many users to use the program and save their scores to their respective accounts with ease of access. The results screen displays all the information (pages number, correct/incorrect answers) for the user to refer back to the text and find what they need to improve on for the next time they take the quiz. The program is uniform across all screens and information is presented in a straightforward manner allowing for the user to focus primarily on the questions and easily find the information they need to improve on in the future.