COVER PAGE

Tilted Programmers

Kevin Le

Michael Ha

David Luong

David Tran

Jonathan Peng

Stephen Chan

4/6/16

Iteration 2

Introduction to Software Engineering

Professor Ehteshami

**2. Revision History**

**Iteration 2**

**March 25th**

David T: Present

David L: Present

Kevin: Present

Jonathan: Present

Stephen: Present

Michael: Present

Group began coordinating for Iteration 2. Tasks were delegated and will only be 3 user stories. Each member was paired up with another member and were assigned.

1. Mockups
2. Use Cases
3. Test Cases

Each member was also assigned a change that needed to be made from the feedback given in Iteration 1.

**March 27th**

David T: Present

David L: Present

Kevin: Present

Jonathan: Not Present

Stephen: Present

Michael: Not Present

Group gave updates on what they were working on. Worked on a draft of algorithm for saving user’s chapter score. Members were to submit their 3 finished tasks ready to be reviewed in the next meeting.

**April 1st**

David T: Present

David L: Present

Kevin: Present

Jonathan: Present

Stephen: Present

Michael: Present

Worked on final draft of iteration 2. All group members inputted information into the document. Tasks still needed to be finished included the communication diagram, automation, and paragraph summaries.

**April 5th**

David T: Present

David L: Present

Kevin: Present

Jonathan: Present

Stephen: Present

Michael: Present

Working on final draft of iteration 2. All group members worked on automating 2 of the tests. Final Draft of iteration 2 was finalized.

Table of Contents

Content: Page Number:

Revision History...................................................................................................................2-3

Table of Contents…………………………………………………………………………..4

Pre-Game Planning...............................................................................................................5

Use Case Diagram.................................................................................................................5

Use Cases..............................................................................................................................6-7

Staging Grooming.................................................................................................................7

User Story..............................................................................................................................8-9

Sprint Backlog Screenshots...................................................................................................10-11

Explanation: System & Documented Activities....................................................................12

Class Diagram........................................................................................................................13

Communication Diagram.......................................................................................................14

Test Plan, Test Suite, Test Cases............................................................................................14-17

Automation…..……………………………………………………………………………...17-21

Code for 3 functional user stories………………………………………………………..….22-24

User Manual............................................................................................................................24-27

References...............................................................................................................................27

Team Charter...........................................................................................................................27-31

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

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

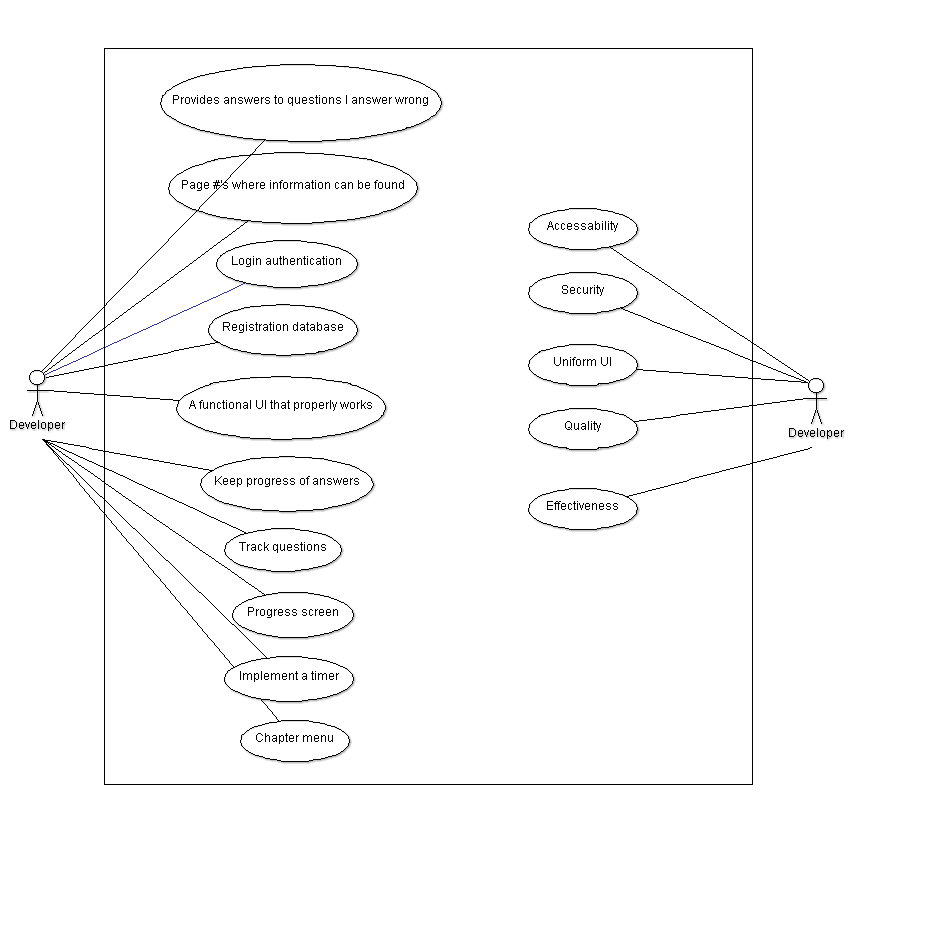
Chapter 10: Requirements Modeling: Class- Based Methods.............................................184-202

**Explain how your team worked on Pre-game Planning**

**Pre Game Planning**

For this Iteration, we wanted to focus on UI improvements as well as developing features that will enhance the user experience. We planned on including a timer that will keep track of the user’s progress as they’re taking the quiz. We wanted to have a method for the user to see the page numbers for the questions they answered in order to assist them with finding the correct answers. Synchronizing the UI took a bit of time as we had to decide on which components of the UI needed to be improved in order to assist the user experience. We did planning poker with the entire group to measure the amount of time it would take to implement each user story. Mockups were created in order to fully express our vision towards the iteration and allowed the developers to see how to implement the user story’s.

**3. Use Case Diagram**



**4. Use-case (textual)**

## **Use Case:** Synchronize UI/Text across all pages

**ID :** UC\_10

**Description:** The buttons are the same size across all pages.

**Level:** Low

**Primary Actor:** Students, professors

**Supporting Actors:**

## **Main Success Scenario**

1. User logins to program.
2. User views questions, chapters, results page
3. Buttons are same size across all pages and text is clearly distinguished from one another.

## **Extensions**

**Use Case:** Page numbers displayed on Results screen

**ID :** UC\_11

**Description:** The user is able to see the pages for each answer in the results screen.

**Level:** low

**Primary Actor:** Students, professors

**Supporting Actors:**

## **Main Success Scenario**

1. User clicks on grade button after completing
2. Program displays results page with corresponding page numbers to questions.

## **Extensions**

11a. User does not answer any questions and continues to result page.

System displays result page regardless marking all questions wrong and displaying all page numbers.

11b. User partially completes review questions and continues to result page.

System displays result page regardless marking all questions wrong and displaying all page numbers.

## **Use Case:** Viewing the Timer

**ID :** UC\_12

**Description:** The user should be able to see the timer while taking the test.

**Level:** High

**Primary Actor:** Students, professors

**Supporting Actors:**

**Precondition:** User logins to program and selects a chapter.

## **Main Success Scenario**

1. User then views running timer on top right corner
2. User sees elapsed time at results page.

## **Extensions**

**Staging - Grooming** In our Iteration 2 of Staging, we first looked for possible issues with our iteration 1. From causal analysis, we found some very small bugs that were easily fixed however, that does not conclude our iteration 2. We brainstormed new ways to better improve our user interface design and provide more information to our users. We came up with the idea of further synchronizing our design across the entire program to make it easy for users to use. We further believed adding a timer could assist with timed exam practices. Lastly we believed it would further benefit our users if we added page numbers to each question that were used in review. As for prioritizing each task, we decided these were all low mainly because they are additional features that are not required for the core functionality but will make the user experience more accessible.

**5. User Story**

**US-9: Synchronize UI**

As a user, I would like the UI to and text to be the same across each page so that it looks efficient.

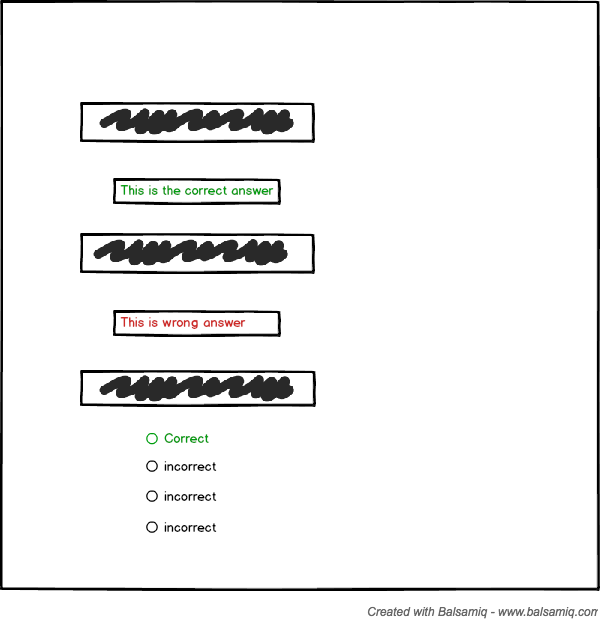
Relation to Use Case: Related to all Use cases since buttons will be same sizes and text will be aligned with correct/wrong answers.

Assigned Developer: Stephen, Jonathan

Due Date: 4/3/16

Priority: Low

MockUp:



**US-10: Timer for Pages**

As a user I would like a timer to keep track of my progress during answering questions.

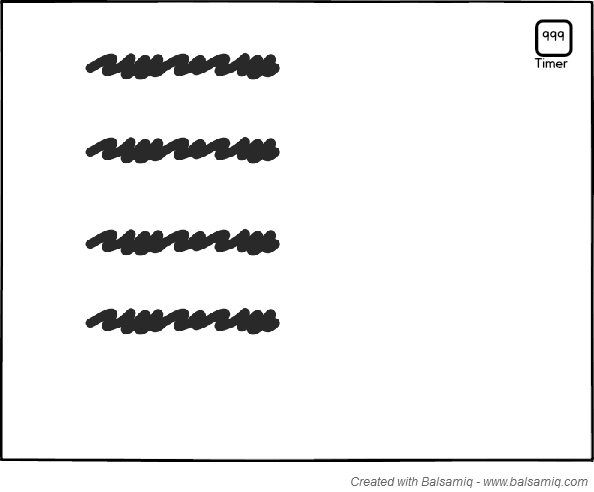
Relation to Use Case: Relates to question screen and result screen

Assigned Developer: Michael, Kevin

Due Date: 4/1/16

Priority: Low

MockUp:



**US-11: Pages Numbers for Results Screen**

As a user, I would like to see the page numbers on the results screen for each question I got correct/incorrect.

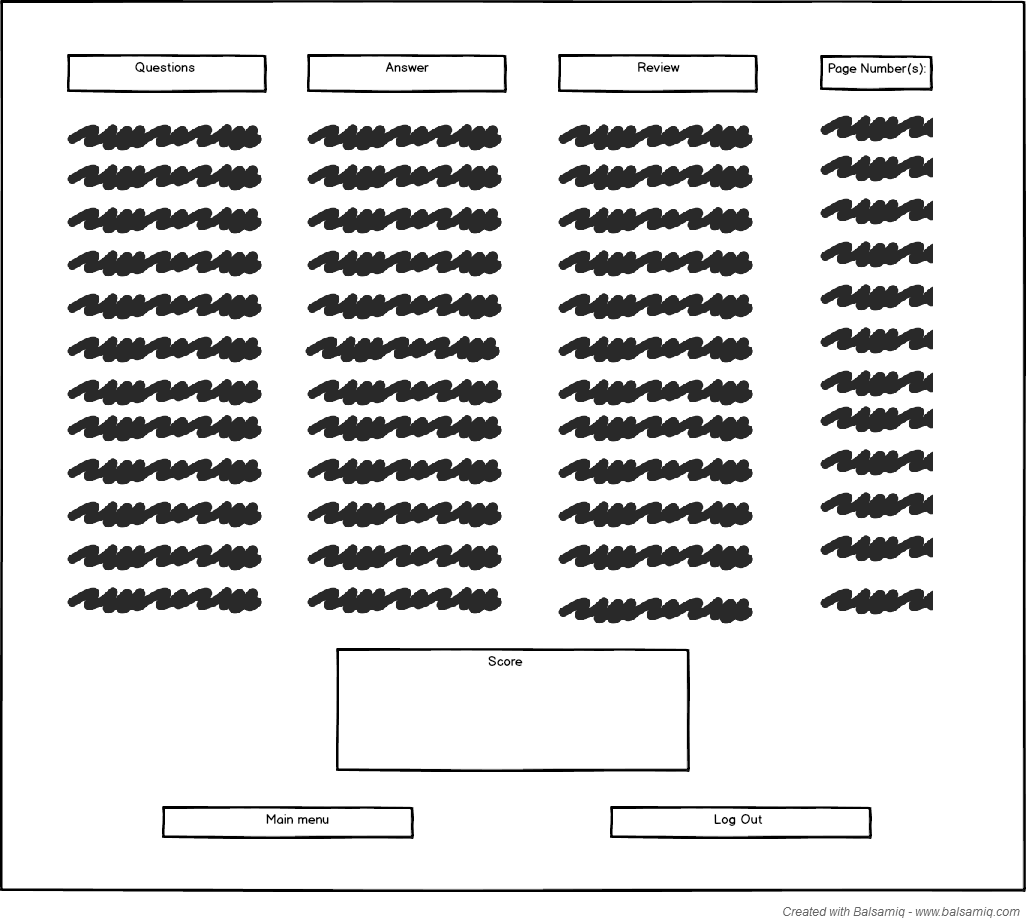
Relation to Use Case: Relates to results screen and answers.

Assigned Developer: David Luong, David Tran

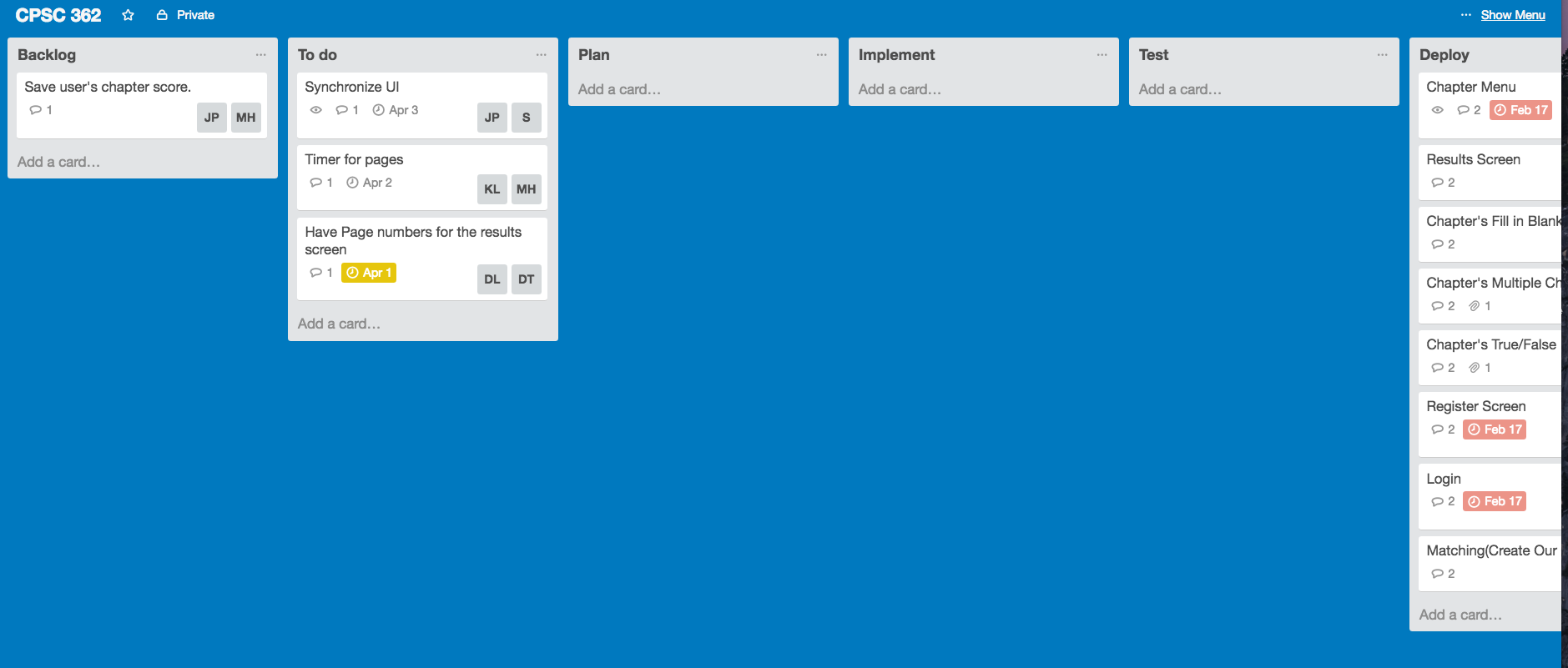
Due Date: 4/1/16

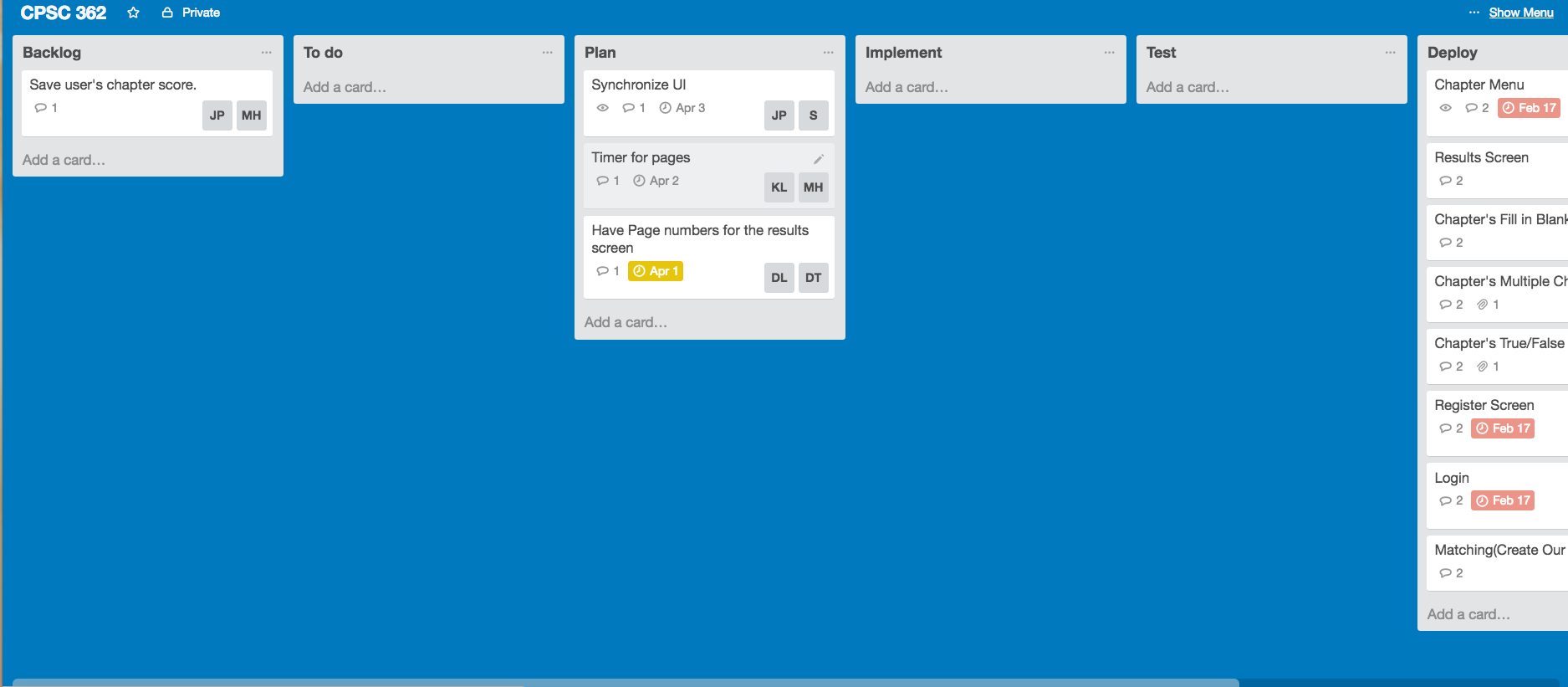
Priority: Low

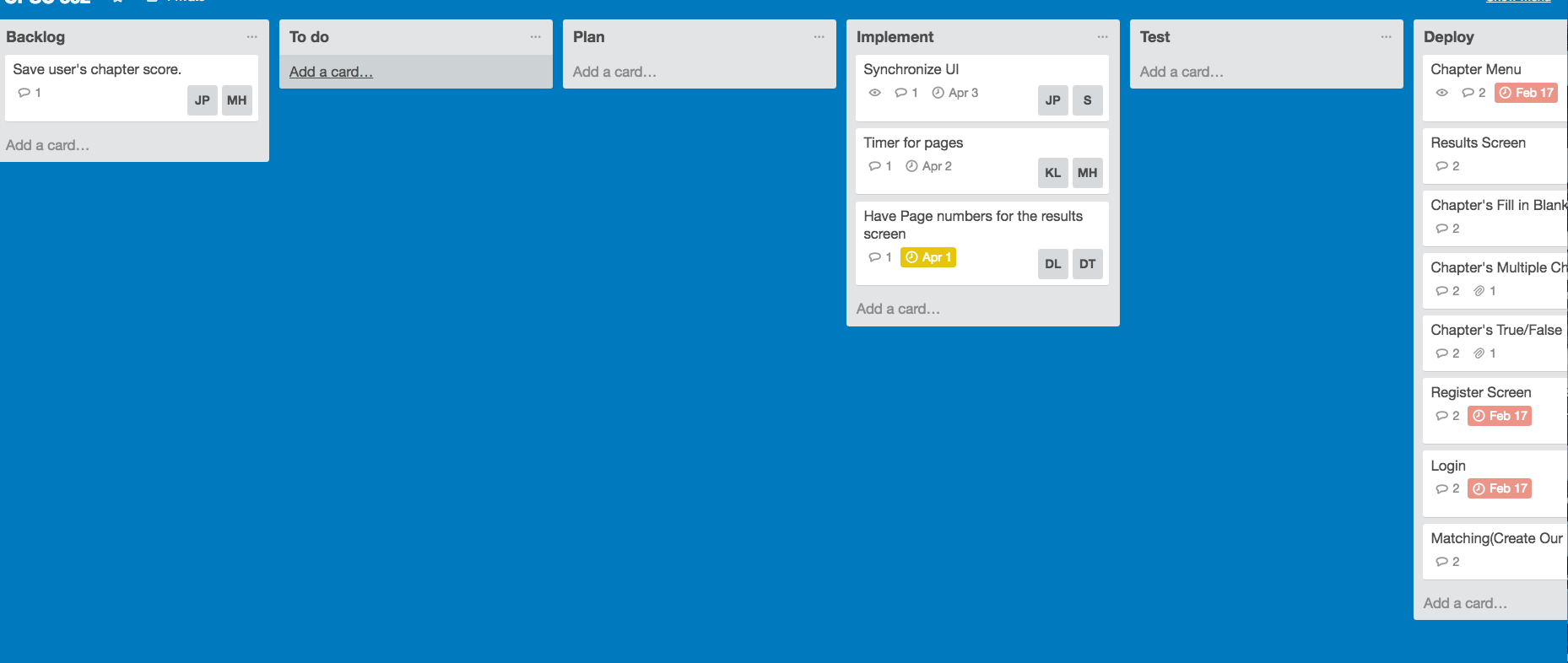
MockUp:

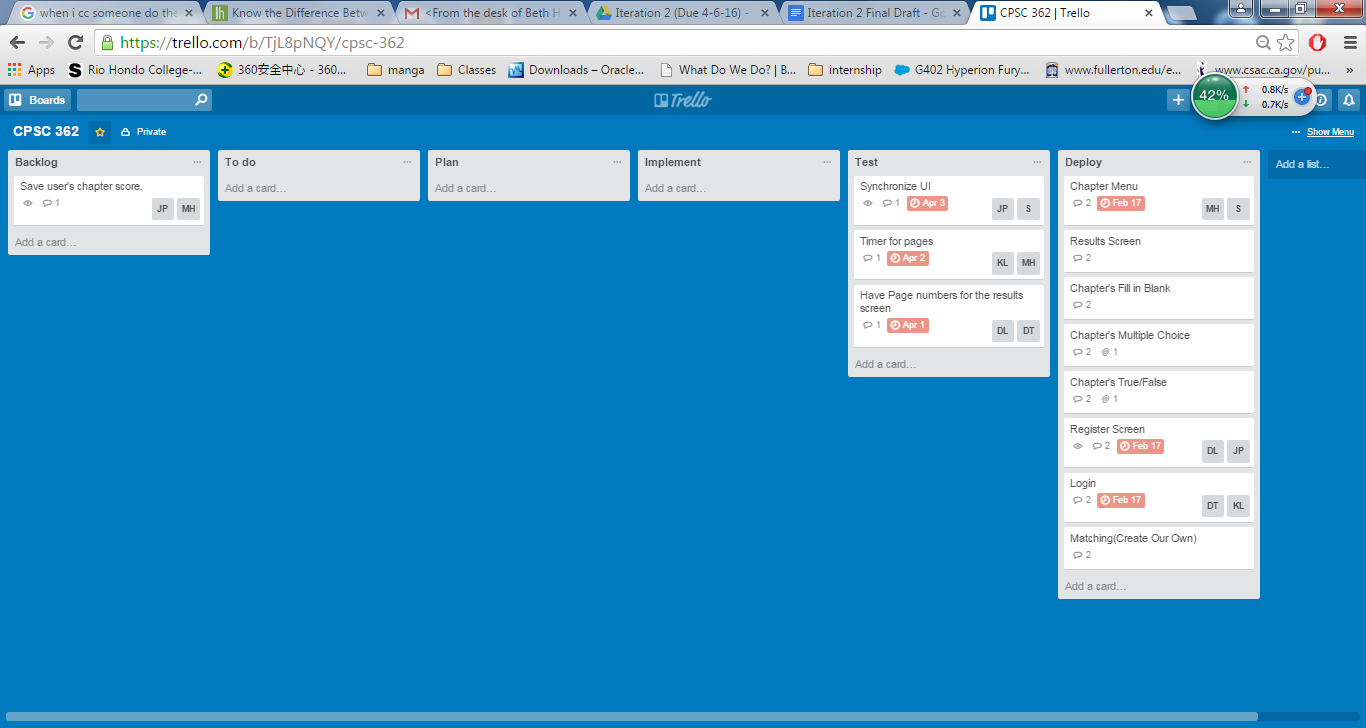


**6. Sprint Backlog Screenshots of your Trello Kanban-Board.**





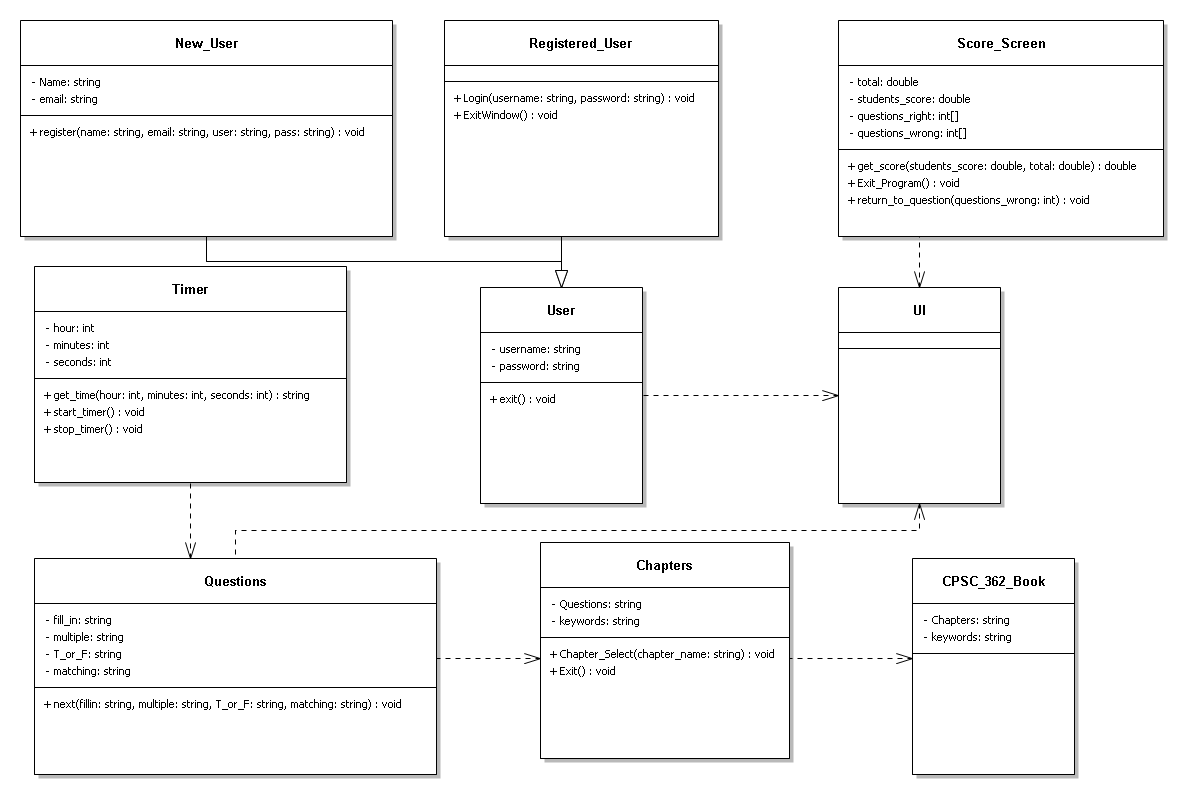




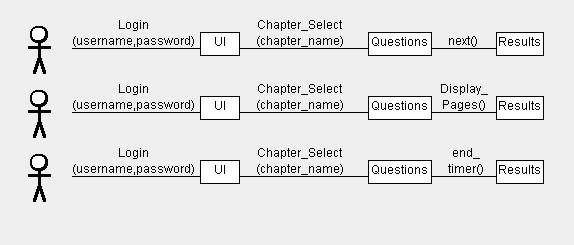
**Explain how your team developed your system and documented activities by producing work products related to this phase.**

Since we finished most of our working product in the last iteration, we decided to focus on three user stories to improve our customer experience. These three aspects were synchronizing our UI, implementing a timer and adding page numbers when users returned to the questions from the results screen. For the timer, we made our programmers work on a timer that will display itself while the user takes the test. When adding the page numbers, we went back to the book and retrieved the page numbers for each question and eloquently added them. After doing the implementation of these user stories, we used our test cases to make sure they worked properly. Automation assisted us by testing many sequences of the program to ensure it runs fluently with minor errors.

**7. Class diagram**



**8. Communication Diagrams based on your Class diagram.**



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

**Iteration 2 Test Cases**

|  |  |
| --- | --- |
| **Test Designed by: Stephen Chan** | **Module Name: Viewing results with colored text** |
| **Test Designed date: 4-3-2016** | **Test Title: Questions correct are highlighted with green text and questions wrong are with red text.** |
| **Test Case ID:23 UC-10** | **Description: User sees buttons that are the same size across all pages and colored text indicates status of their answer.** |
| **Test Priority (Low/Medium/High): Low** | **Pre-conditions: User is logged in. User has attempted all questions.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User is able to see a list of questions that reveals which questions they got correct.** |  | **List of the questions with an indicator of whether they got it right or wrong.** | **Questions appear with an indicator of if they’re right or wrong.** | **Pass** |  |
| **2** | **User clicks on return back to questions button** |  | **User is returned back to all questions.** | **User is returned back to all questions** | **Pass** |  |
| **3.** | **Text is highlighted green or wrong with answer to specify if they got the answer correct or incorrect.** |  | **Users sees answers to all questions** | **User answers are highlighted depending on if it's correct/incorrect** | **Pass** |  |

**Post-conditions:**

**User is finished with the program.**

|  |  |
| --- | --- |
| **Test Designed by: David Luong** | **Module Name: Results page without answered questions** |
| **Test Designed date: 4-3-2016** | **Test Title: Results page and page numbers.** |
| **Test Case ID: 24 UC-11** | **Description: Page numbers display without having answer questions in results page.** |
| **Test Priority (Low/Medium/High): Low** | **Pre-conditions: User is logged in and selects a review chapter.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User does not answer any questions.** |  | **No questions are answered.** | **No questions are answered.** | **Pass** |  |
| **2** | **User proceeds to click on results without having answer questions** |  | **Results page displays questions with their corresponding answers and page numbers** | **Results page displays questions with their corresponding answers and page numbers** | **Pass** |  |

**Post-conditions:**

**User is finished with the program**

|  |  |
| --- | --- |
| **Test Designed by: David Luong** | **Module Name: Results page partial answered questions** |
| **Test Designed date: 4-3-2016** | **Test Title: Results page and page numbers.** |
| **Test Case ID: 25 UC-11** | **Description: Page numbers display having partial answered questions in results page.** |
| **Test Priority (Low/Medium/High): Low** | **Pre-conditions: User is logged in and selects a review chapter.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User partially completes review questions.** |  | **Review partially answered.** | **Review partially answered.** | **Pass** |  |
| **2** | **User proceeds to click on results without having answered all questions** |  | **Results page displays questions with their corresponding answers and page numbers** | **Results page displays questions with their corresponding answers and page numbers** | **Pass** |  |

**Post-conditions:**

**User is finished with the program**

|  |  |
| --- | --- |
| **Test Designed by: Michael Ha** | **Module Name: Timer** |
| **Test Designed date: 4-3-2016** | **Test Title: Timer count test** |
| **Test Case ID: 26 UC-12** | **Description: User can see a timer while taking their questions** |
| **Test Priority (Low/Medium/High): Low** | **Pre-conditions: User is logged in.** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Step** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| **1** | **User is in chapter questions** |  | **User can see timer** | **User sees timer** | **Pass** |  |

**Post-conditions:**

**User is finished with the questions.**

**10. Automation**

**Test Case: Selecting a chapter.**

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 AutomationTest1

{

/// <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.SelectChapter();

}

#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;

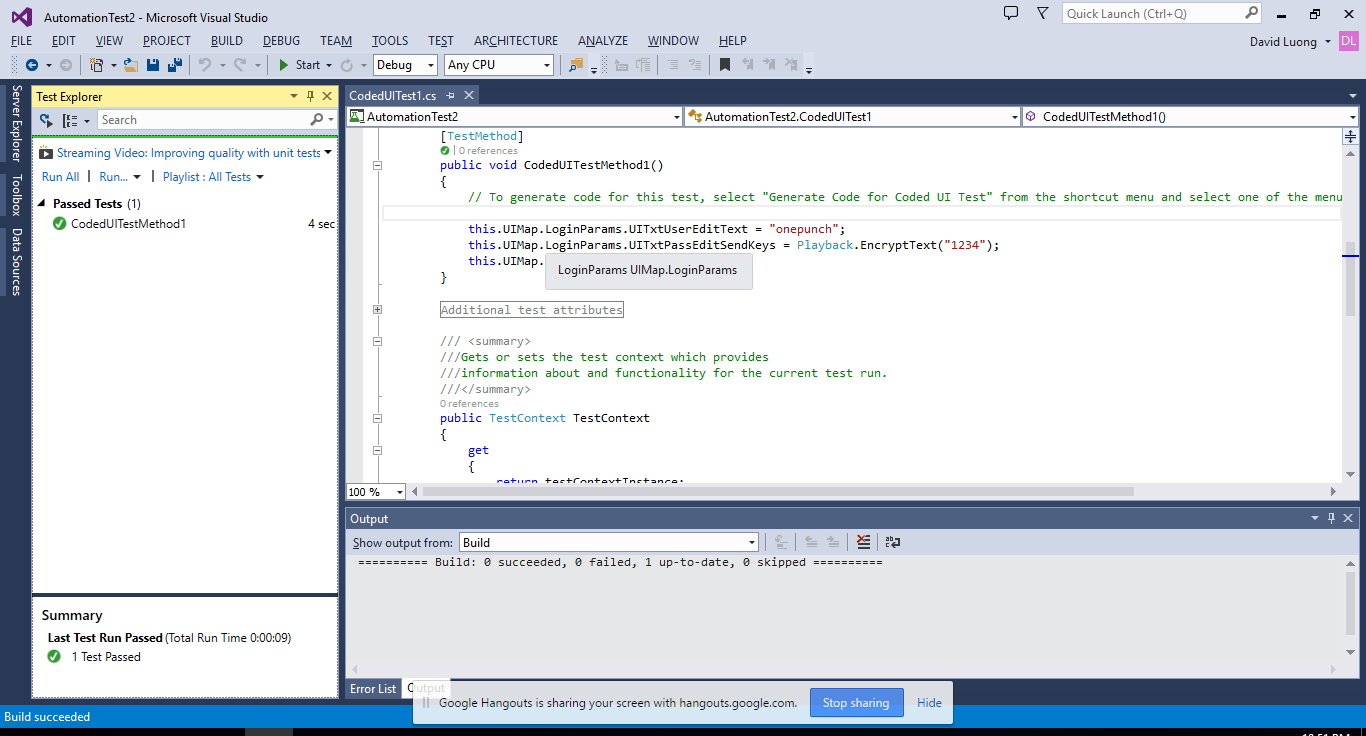
}

}

private UIMap map;

}

}



**Test Case: Registering a User**

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 AutomationTest3

{

/// <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.

Random rand = new Random();

int r = rand.Next(0, 100);

this.UIMap.RegisterUserParams.UITxtUserEditText = "kevinle" + r;

this.UIMap.RegisterUser();

}

#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;

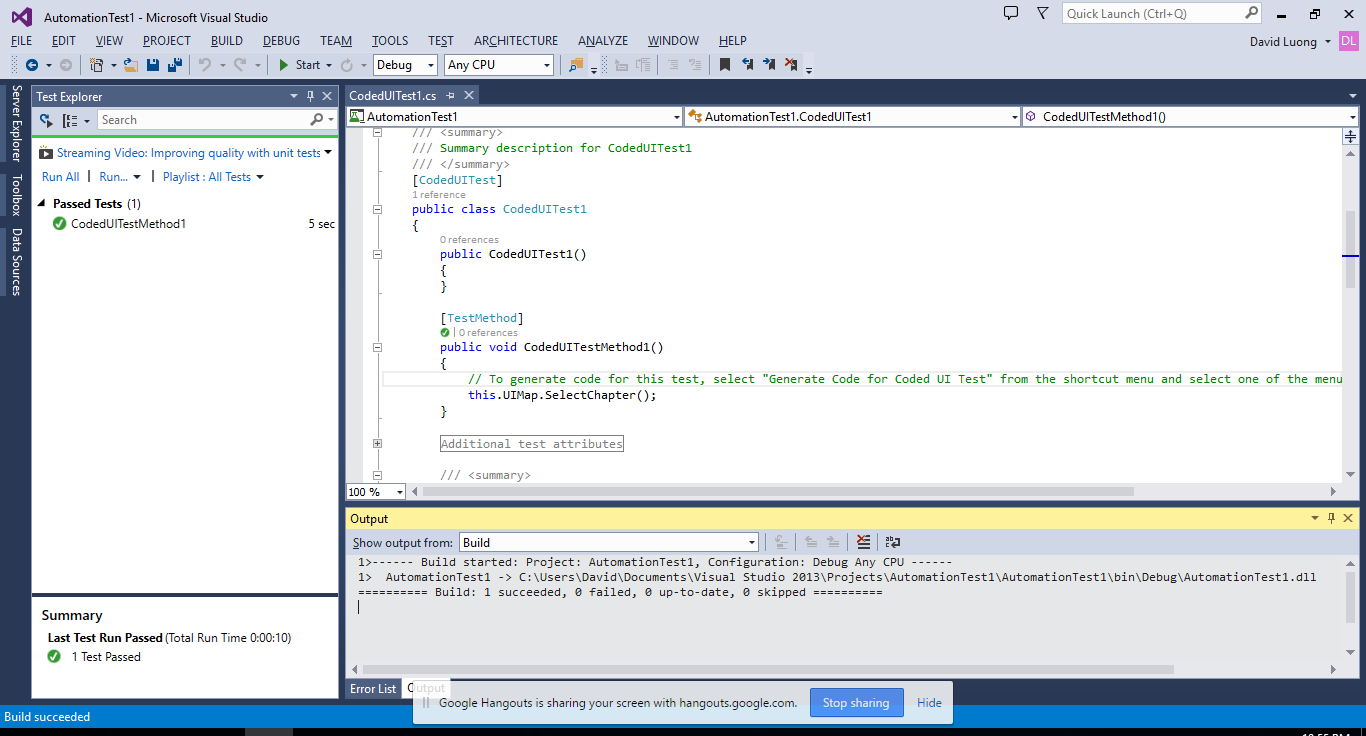
}

}

private UIMap map;

}

}



**11. Your code for the 3 functional user stories.**

**A. Viewing Timer on Chapter Screen**

private void C9timer\_Tick(object sender, EventArgs e)

{

endTime = DateTime.Now;

TimeSpan endoftime = (endTime - startTime);

string elapsedTime = String.Format("{0:00}:{1:00}:{2:00}",

endoftime.Hours, endoftime.Minutes, endoftime.Seconds,

endoftime.Milliseconds / 10);

lbltimer.Text = elapsedTime;

}

}

}

**B. Viewing Page Number in Result Screen**

//A switch case to know which questions are from so it can read the page number from the

//text file of the chapter

switch (chapter)

{

case 3:

pagenumber = System.IO.File.ReadAllLines("chapter3.txt");

break;

case 4:

pagenumber = System.IO.File.ReadAllLines("chapter4.txt");

break;

case 5:

pagenumber = System.IO.File.ReadAllLines("chapter5.txt");

break;

case 8:

pagenumber = System.IO.File.ReadAllLines("chapter8.txt");

break;

case 9:

pagenumber = System.IO.File.ReadAllLines("chapter9.txt");

break;

case 10:

pagenumber = System.IO.File.ReadAllLines("chapter10.txt");

break;

}

//Output the Question number and if the user answered correct or incorrect

//on the data grid.

for(int i = 1; i <= 12; ++i)

{

ResultGrid.Rows.Add("Question " +i, arrayC[i], pagenumber[i]);

give\_back[i] = returnf[i];

}

**C. Text color(Synchronize UI)**

**1.Result Screen Text Color**

//This for loop change the text color of "Incorrect" to red and "Correct" to

//green in the datagridview

for (int i = 0; i < 12; ++i)

{

if (ResultGrid.Rows[i].Cells[1].Value.ToString() == "Incorrect")

ResultGrid.Rows[i].Cells[1].Style = new DataGridViewCellStyle { ForeColor = Color.Red};

else

ResultGrid.Rows[i].Cells[1].Style = new DataGridViewCellStyle { ForeColor = Color.Green};

}

**2.Chapter Screen Text Color**

//This function would show all the correct answer

public void showAns()

{

if (rdbFalse1.Checked == true)

lblA1.ForeColor = System.Drawing.Color.Green;

if (rdbTrue2.Checked == true)

lblA2.ForeColor = System.Drawing.Color.Green;

if (rdbTrue3.Checked == true)

lblA3.ForeColor = System.Drawing.Color.Green;

if (txtTBA1.Text == "structured analysis")

lblA4.ForeColor = System.Drawing.Color.Green;

if (txtTBA2.Text == "goal in context")

lblA5.ForeColor = System.Drawing.Color.Green;

if (txtTBA3.Text == "use cases" || txtTBA3.Text == "use case")

lblA6.ForeColor = System.Drawing.Color.Green;

if (rdb7C.Checked == true)

lblA7.ForeColor = System.Drawing.Color.Green;

if (rdb8A.Checked == true)

lblA8.ForeColor = System.Drawing.Color.Green;

if (rdb9B.Checked == true)

lblA9.ForeColor = System.Drawing.Color.Green;

if (txtQ10.Text == "c")

lblA10.ForeColor = System.Drawing.Color.Green;

if (txtQ11.Text == "a")

lblA11.ForeColor = System.Drawing.Color.Green;

if (txtQ12.Text == "b")

lblA12.ForeColor = System.Drawing.Color.Green;

lblA1.Text = "Answer: False";

lblA2.Text = "Answer: True";

lblA3.Text = "Answer: True";

lblA4.Text = "Answer: Structured Analysis";

lblA5.Text = "Answer: Goal in Context";

lblA6.Text = "Answer: Use Case";

lblA7.Text = "Answer: Swimlane Diagram";

lblA8.Text = "Answer: To make it as elaborate as possible";

lblA9.Text = "Answer: Involves in critical activity";

lblA10.Text = "Answer: C";

lblA11.Text = "Answer: A";

lblA12.Text = "Answer: B";

}

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

**Registration Screen**



*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

*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

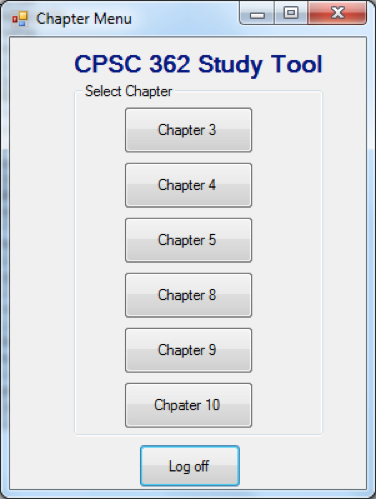
2. Enter your Password

3. Click on “Log in” button

If users want to exit the program, then:

1. Click on the “Exit” button

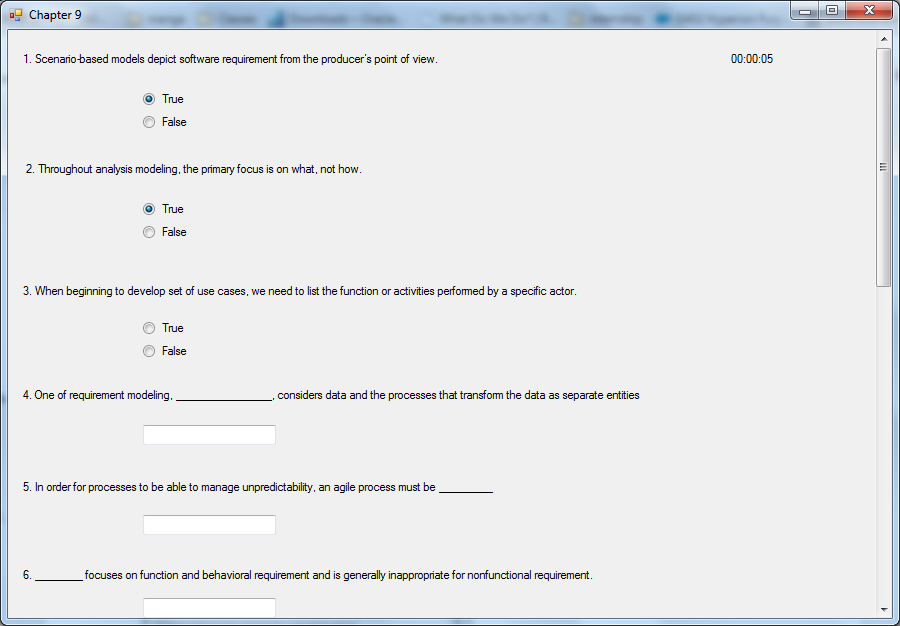
**Chapter Menu Screen**



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

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

**Chapter Screen**

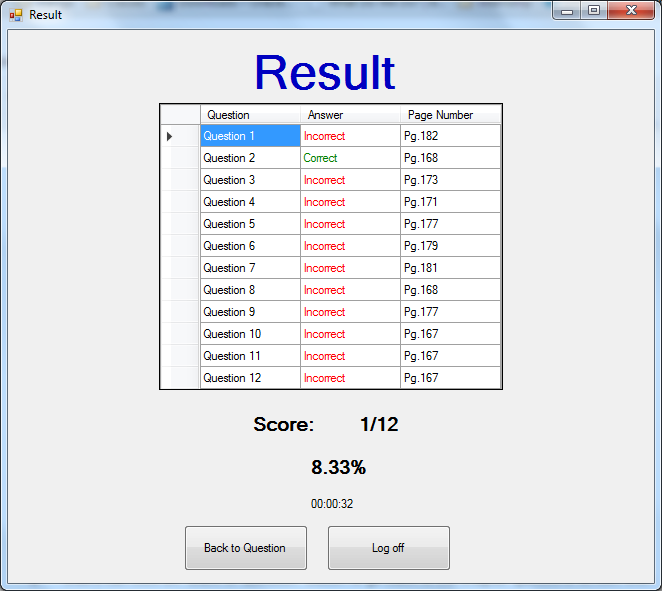


1. Answer the questions given.

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

3. 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.

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

Software Engineering: A Practitioner’s Approach by Roger Pressman

**14. 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. |

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