

Deliverable I

Team Name:

The Pirates

Team Website:

<https://github.com/John-Stovall/ThePirates>

Team Contacts:

Ryan Hansen - ryanchansen@hotmail.com

Rand Almaroof - Rand3@uw.edu

Robert Cordingly - robertcordingly@gmail.com

Reagan Stovall - reaganstovall@gmail.com

User stories:

- 1) As a user, I'd like to have the program help collect data, make basic calculations, and help weigh costs versus benefits for smaller sized projects so I can make the best financial decisions.
- 2) As a user, I'd like to have the option to make several possible projects and choose between them to find the one which best fits my budget.
- 3) As a user, I'd like for the program to have some sort of bill estimator built into it to help me forecast my potential monthly savings.
- 4) As a user, I'd like for the program to be portable and usable on multiple devices for easy access on the go.
- 5) As a user, I'd like for the program to have some sort of export capability so I can save my files for later use and hand them off to possible contractors.
- 6) As a user, I'd like for the application to be accessible without a data connection while using it from a mobile device for ease of access.
- 7) As a user, I'd like for the program to be able to let me edit current projects for small changes or errors I could have made in the initial setup.

Business Rules:

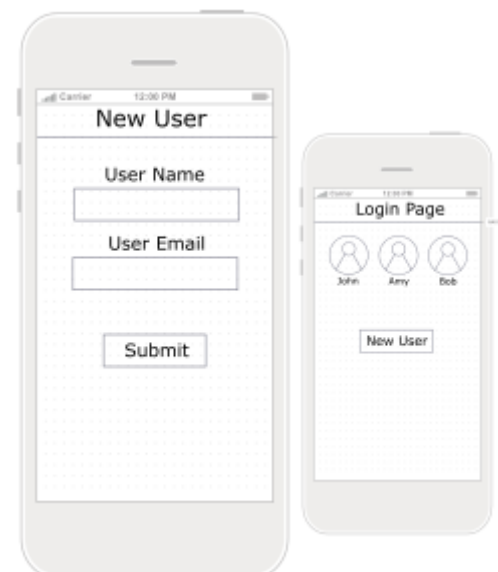
- 1) The program will ask the user for specific data pertaining to each type of project.
- 2) The program will have check boxes which will include or ignore which projects will be considered while calculating monthly savings.
- 3) The program will be able to run on a mobile platform as well as desktop.
- 4) The program will have an edit feature for each different type of project.

Application Design:

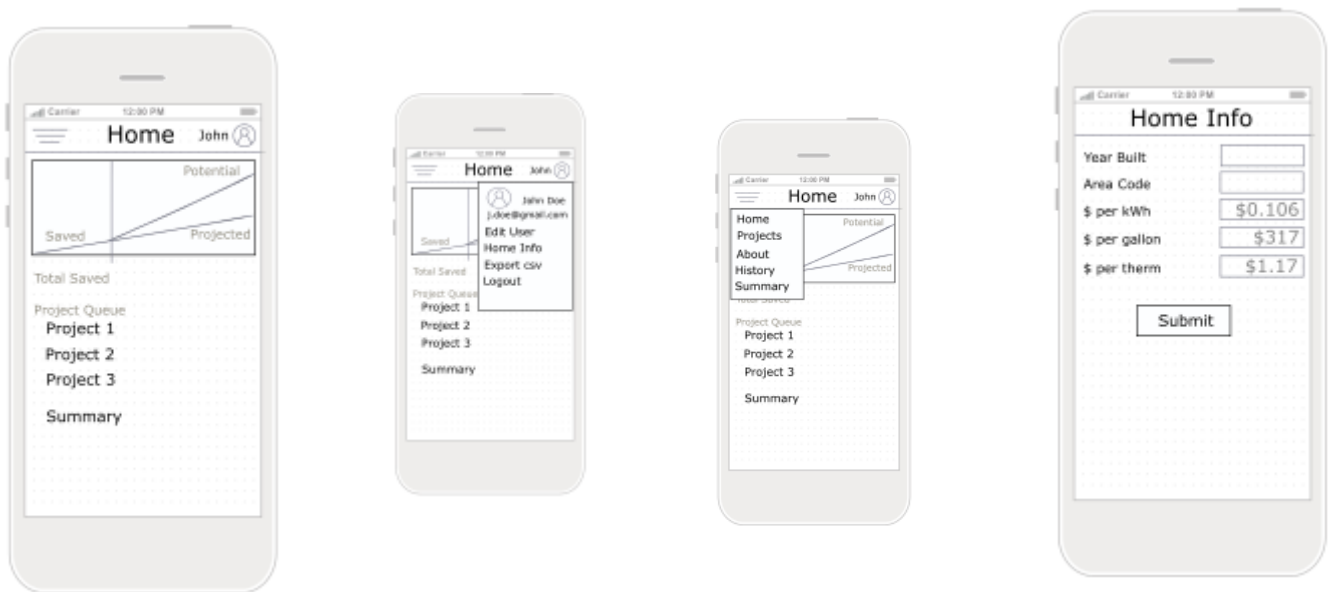
First time use will require a user name as well as a valid email but will stay logged in till the user logs out for convenience.

New Users will also be required to fill out basic home information for improvement calculations. Multiple users will also be supported, but will only be accessible when the current user logs out

The typical application open page will be the Home page where past and future projected savings will be shown in a simple graph. A secondary graph projection will show future possible savings based on projects in the Queue.

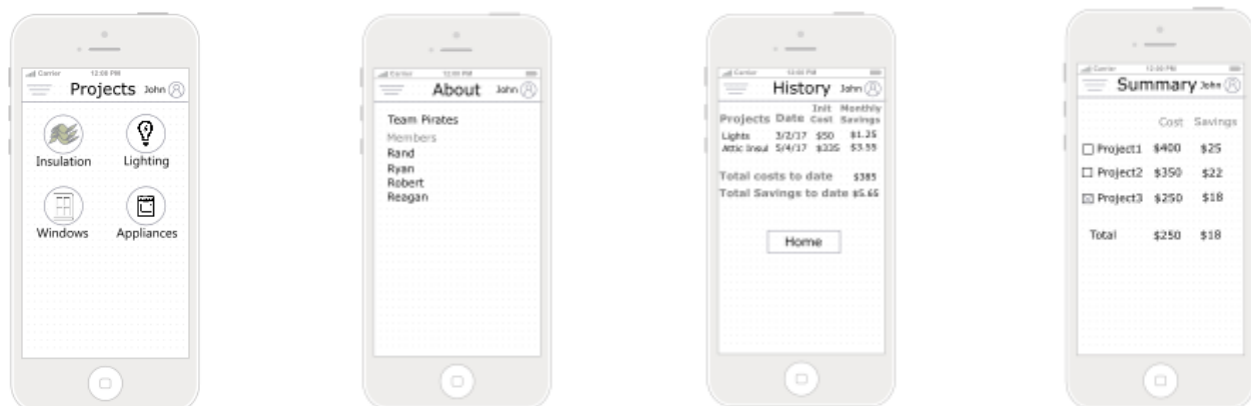


The Icon on the top Left is a directory that leads to other pages.

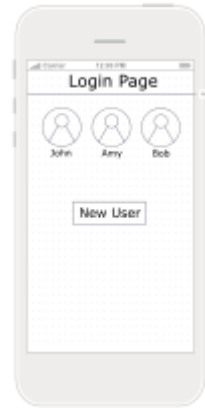
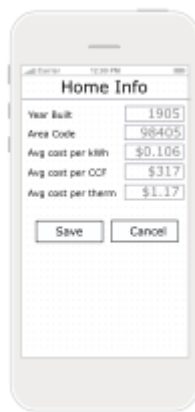


While the user icon on the upper Right opens more settings related options

- **Projects:** The projects page will have icons that when clicked all lead to a new projects.
 - For now we will focus on only insulation and add the rest when we fill comfortable with the design.
- **About:** A simple page that tells the user who made the app and why.
- **History:** A page for reviewing past projects and how much the user has saved from making those changes
- **Summary:** The Summary page shows all currently considered projects and allows for simple cost comparisons. This is also nice in that users can price out different materials and different amounts as different projects for the same fix.



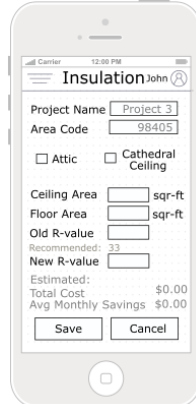
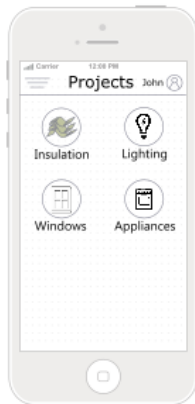
- **Edit User:** Allows the user to change their user-name and or email.
- **Home Info:** Allows the user to update Home Information.
- **Export csv:** This will export a csv file of all information on past projects.
- **Logout:** This takes the user to the login page.



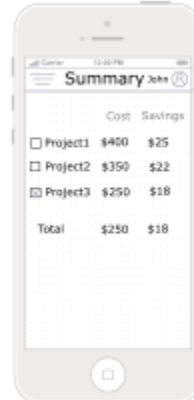
A Little More Detail

Projects:

The projects page has icons that open new projects like Insulation.

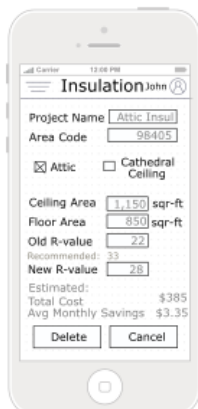


A generic name is left as optional, or the user can specify a new one. The page retrieves the necessary information to calculate the cost and also generates an insulation recommendation depending on the area code.



A current project can also be viewed and accessed from the summary page as well as the Home page.

While past projects can be viewed and accessed from the History Page.



When viewing past projects, only the delete or cancel buttons will be enabled.

From either the Summary Page or Home, if no text fields are selected, A complete option is made available at the bottom of the screen. The complete button removes this project from the Project Queue and Summary Page and



adds it to the History page. It also updates the calculated monthly savings graph to incorporate the new savings

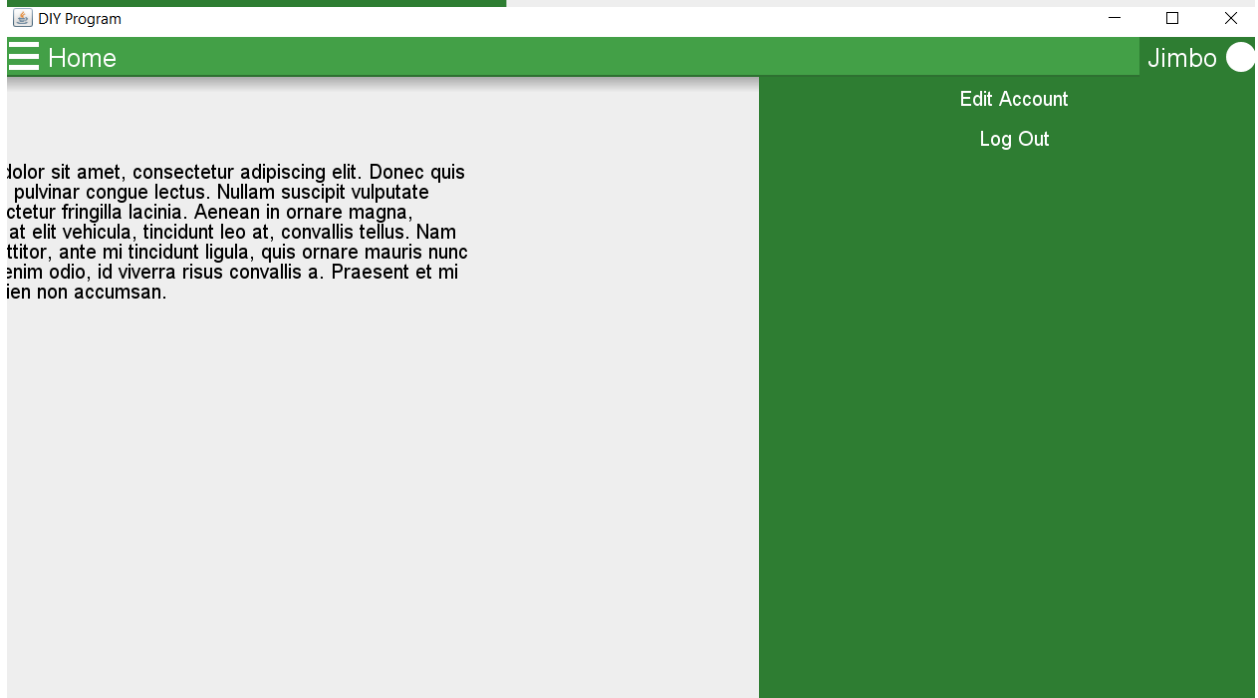
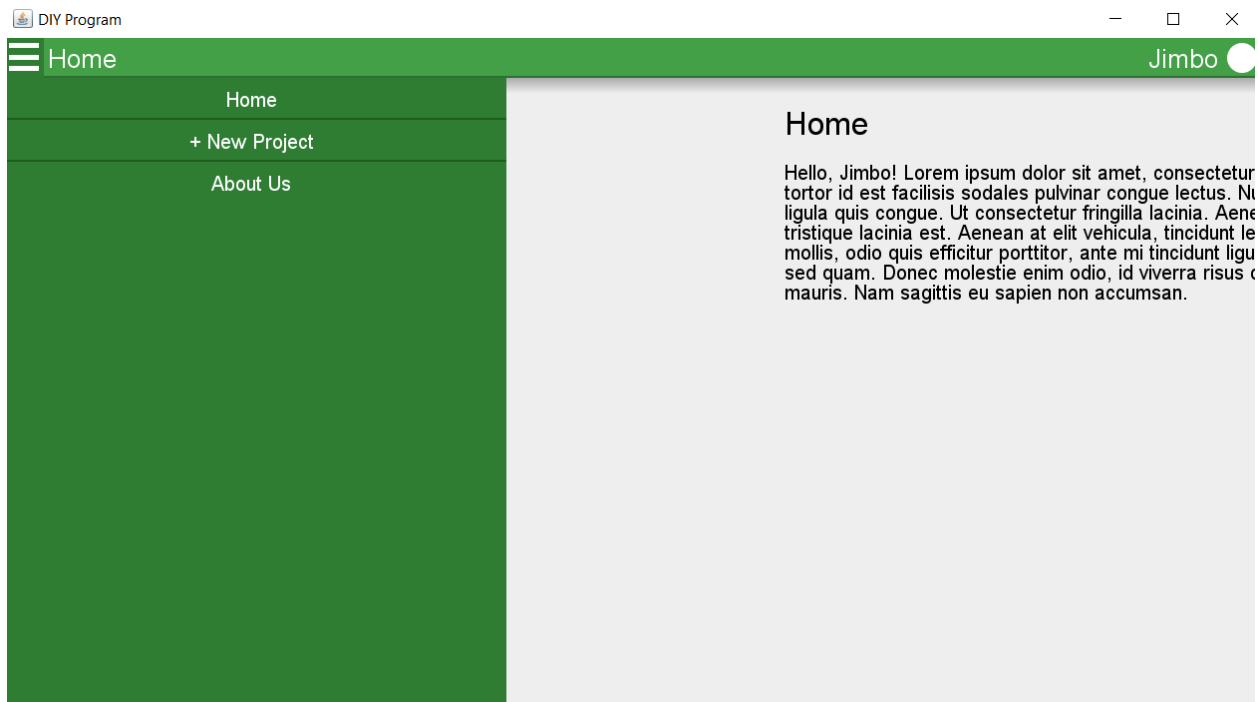
GUI:

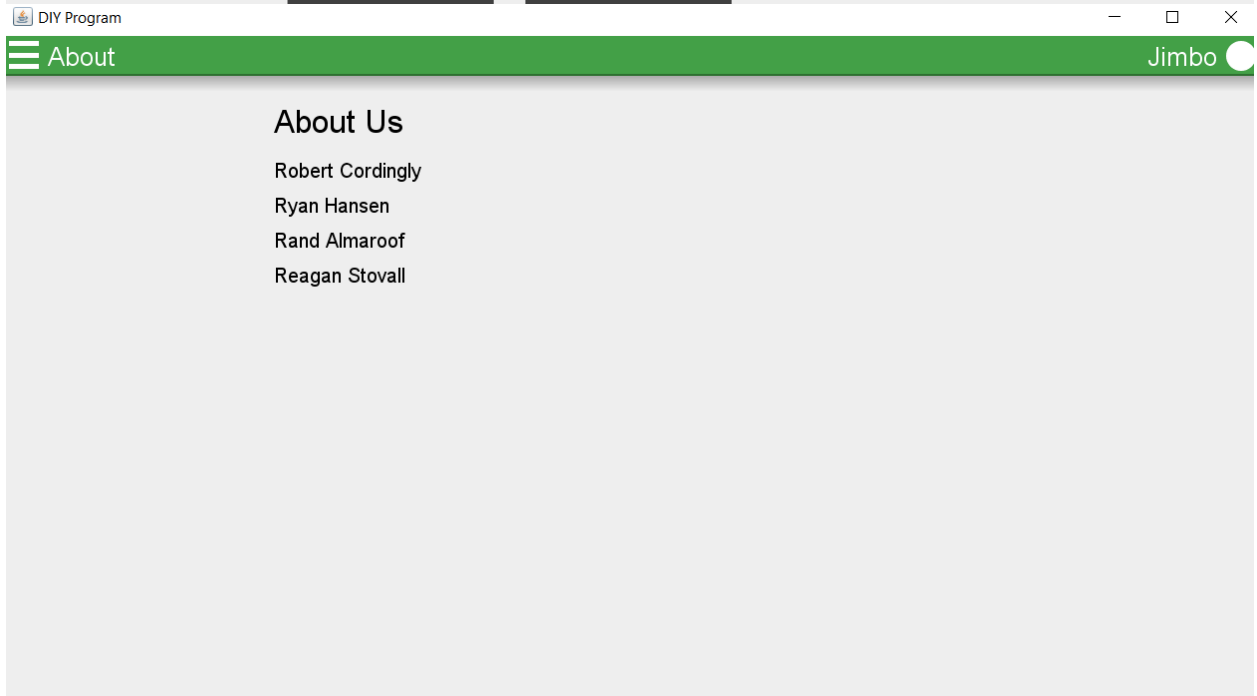
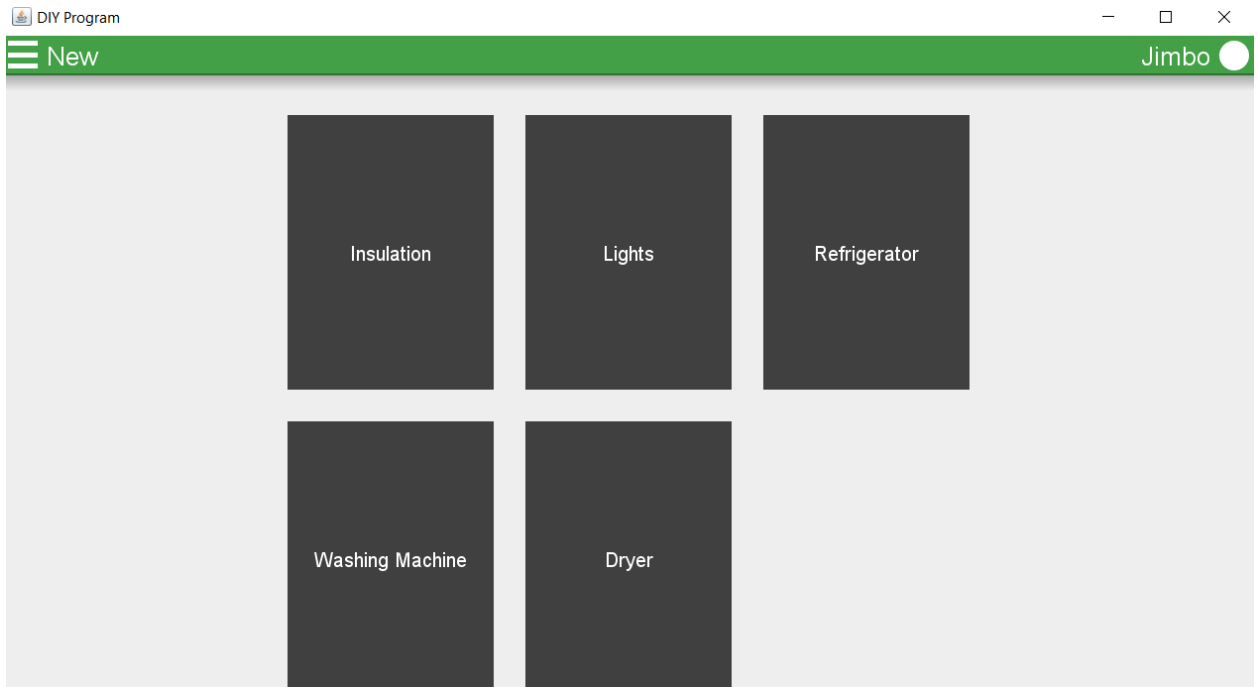
Currently the app opens to the Create Account Page if no user is saved, but gives the option for known users as seen below. The menu bar displays the current User and dropdown is used to navigate. Also, the GUI has New Project addition below where it allows users to create new projects according to their needs.

The image displays two screenshots of a web application titled "DIY Program".

The top screenshot shows the "Create Account" page. It features a form with two input fields: "Name:" and "Email:". Below these fields are two buttons: "Submit" and "Skip Login".

The bottom screenshot shows the "Home" page. It has a green navigation bar at the top with a hamburger menu icon on the left, the text "Home" in the center, and a user profile icon labeled "Jimbo" on the right. The main content area has the heading "Home" followed by a paragraph of Lorem Ipsum text.





Iteration I:

This is what the group had before the discussion we had in class with other groups.

DIY Program

Create Account

Name:

Email:

Submit

Skip Login

Back

DIY Program

home

Jimbo

Home

Hello, Jimbo! Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec quis tortor id est facilisis sodales pulvinar congue lectus. Nullam suscipit vulputate ligula quis congue. Ut consectetur fringilla lacinia. Aenean in ornare magna, tristique lacinia est. Aenean at elit vehicula, tincidunt leo at, convallis tellus. Nam mollis, odio quis efficitur porttitor, ante mi tincidunt ligula, quis ornare mauris nunc sed quam. Donec molestie enim odio, id viverra risus convallis a. Praesent et mi mauris. Nam sagittis eu sapien non accumsan.

[Home](#)[New Project](#)[Page 3](#)[Page 4](#)[About Us](#)

Home

Hello, Jimbo! Lorem ipsum dolor sit amet, consectetur id est facilisis sodales pulvinar congue ligula quis congue. Ut consectetur fringilla la tristique lacinia est. Aenean at elit vehicula, mollis, odio quis efficitur porttitor, ante mi tir sed quam. Donec molestie enim odio, id viv mauris. Nam sagittis eu sapien non accums

[Page 1](#)[Page 2](#)[Log Out](#)

et, consectetur adipiscing elit. Donec quis congue lectus. Nullam suscipit vulputate illa lacinia. Aenean in ornare magna, cula, tincidunt leo at, convallis tellus. Nam mi tincidunt ligula, quis ornare mauris nunc id viverra risus convallis a. Praesent et mi sumsan.

About Us

Robert Cordingly

Ryan Hansen

Rand Almaroof

Reagan Stovall

Junit Tests:

- That the user name was at least 4 characters long, excluding leading and trailing spaces.
- The email provided contained at least one @ sign.
- The email provided contained a ".com" as we are somewhat exclusive.
- Built the Check methods to easily accommodate future tests if needed.

The screenshot displays an IDE with two main panels. The left panel shows the 'Package Explorer' and 'JUnit' test results. The 'JUnit' tab indicates 'Finished after 0.031 seconds' with 'Runs: 8/8', 'Errors: 0', and 'Failures: 0'. Below this, a tree view shows 'gui.Test [Runner: JUnit 4] (0.004 s)' containing seven test cases: test1 (0.000 s), test2 (0.000 s), test3 (0.000 s), test4 (0.000 s), test5 (0.000 s), test6 (0.001 s), and test7 (0.000 s). The right panel shows the source code for 'JTest.java'. The code includes comments for naming tests and email tests, and implements seven test methods (test1 through test7) using Assert assertions to validate user names and email addresses.

```
9 //Naming Tests
10 @Test
11 public void test1() {
12     String test1 = "bob";
13     Assert.assertFalse(Main.testName(test1));
14 }
15
16 @Test
17 public void test2() {
18     String test2 = "boby";
19     Assert.assertTrue(Main.testName(test2));
20 }
21
22 @Test
23 public void test3() {
24     String test3 = "Big John";
25     Assert.assertTrue(Main.testName(test3));
26 }
27
28 @Test
29 public void test4() {
30     String test4 = " bob ";
31     Assert.assertFalse(Main.testName(test4.trim()));
32 }
33
34 //emailTests
35 @Test
36 public void test5() {
37     String test5 = "bob@gmail.com";
38     Assert.assertTrue(Main.testEmail(test5));
39 }
40
41 @Test
42 public void test6() {
43     String test6 = "boby.uwt.edu";
44     Assert.assertFalse(Main.testEmail(test6));
45 }
46
47 @Test
48 public void test7() {
49     String test7 = "boby.uwt.edu";
50 }
```

```
179  /**
180   * This is a simple character counter that returns true if there are
181   * less than 4 characters.
182   * @param name
183   * @return
184   */
185  public static boolean testName(String name){
186      if(name.length() < 4){
187          System.out.println("Less than 4 characters");
188          return false;
189      }else{
190          System.out.println("LengthTest Passed");
191          return true;
192      }
193  }
194
195  /**
196   * This tests that there is an @ sign and a '.com' a bit lacking,
197   * but we can expand it if we want.
198   *
199   * @param email
200   * @return
201   */
202  public static boolean testEmail(String email){
203      // contains @
204      if(email.indexOf('@') == -1){
205          System.out.println("has doesn't have an @");
206          return false;
207      }else{
208          System.out.println("@test Passed");
209      }
210      // contains a '.com' at the end
211      if(!email.toLowerCase().endsWith(".com")) {
212          System.out.println("does not end with .com");
213          return false;
214      }else{
215          System.out.println(".com-test Passed");
216      }
217      return true;
218  }
```