|  |  |  |
| --- | --- | --- |
| **Deliverable II** | | |
| **Team Name:**  The Pirates | **Team Website:**  <https://github.com/JohnStovall/ThePirates> |
| **Team Contacts:**  Ryan Hansen **-** [ryanchansen@hotmail.com](mailto:ryanchansen@hotmail.com)  Rand Almaroof  **-** [Rand3@uw.edu](mailto:Rand3@uw.edu)  Robert Cordingly **-** [robertcordingly@gmail.com](mailto:robertcordingly@gmail.com)  Reagan Stovall **-** [reaganstovall@gmail.com](mailto:reaganstovall@gmail.com) | | |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Introduction:**

The document in hand is to help users and developers to have an understanding for DIY Energy Saver application that is designed and implemented by the Pirates team in Software Development and Quality Assurance Techniques class at University of Washington, Tacoma. There are two aspects of the application: the general user interface (GUI or UI) and Unified Modeling Language (UML). From the users’ point view, GUI (especially friendly use GUI) is more important. The UML is more likely for developers and debugger; it includes the underlining of the GUI and how to save, modify, and remove objects based on commands that users instantiate in GUI.

This document will go over, not particularly the same order, Project general rational summary, user stories, UML design, UML implementation, GUI design, GUI implementation, Sequence diagrams for system and user, and a final summery of the DIY Energy Saver application. The Contents section will go thoroughly over the order and where each appropriate section reside in this document.

Contents

[I. General Project Rational Summary: 5](#_Toc482642129)

[II. Class Diagrams: 5](#_Toc482642136)

[A. GUI: 6](#_Toc482642137)

[B. Pages: 7](#_Toc482642154)

[C. UML: 8](#_Toc482642162)

[III. User Story Sequence Diagrams: 8](#_Toc482642169)

[IV. System Sequence Diagram for startup: 10](#_Toc482642177)

# **General Project Rational Summary:**

**The DIY Energy Saver project is an application written in Java by the Pirates group. The project aims to help users to have theoretical projects around their house that calculate the costs of the project and the amount of savings that might apply to the user’s power bill. The application is meant to be computationally helpful and user-friendly; to achieve so, we (the Pirates) made few choices over others.**

## **Java as programming language: the initial choice of programming language was JavaScript because the group felt it would be easier to achieve one of the project’s user stories, which was mobility. After the first two weeks, the group agreed to switch to Java due to everyone having exposure to Java rather than having the group to improve in JavaScript, which members lightly used in the past. Java is advanced language and all members (and class) have done class requirements of taking Java classes. There is also easy access to Java troubleshooting problem through the internet.**

## GUI implementation from AWT: .awt is a java library that consists of any user interface implementation. The group decided to use this library to gain freedom and full control of any user interface object. This has given us the power to control our run time and draw costume buttons, labels, menus, and pages that would fit with the purposes of the final product.

## The GUI and UML separation: both GUI and UML are separated in two different packages. This helped us to be more organized and gave us the freedom upon debugging the code.

## Using interfaces/abstract classes in UML: having interfaces and abstract classes in UML makes the code clean and easy for both user interface and debuggers. The interfaces give developers the power on paving the way for the classes to implement the appropriate methods according to the functionality of the class.

## Separation of GUI classes: we have separated each GUI component into a class: buttons have GButton, Pages have GPage. This allows programmers to change those components in a neat way; no need to destroy the whole GUI to change the color of just one label. Also, each project type will have its own class. Previously, all pages were created in main() method and the group agreed to clean it up. The pages of each project type provide easy access in case of improvement and debug.

## Future addition- more UML classes: after hearing form different groups, the group decided to add more UML classes to ease the users’ interfaces and make the application more efficient. For example: the group will add a class for green calculator, using data from Washington state website, to have more reliable project.

# **Class Diagrams:**

Our class diagrams separate in three categories (further revisions might be considered to have more separation to keep cleanness of the project), those categories represent the packages in the project, which are:

## GUI:

Add GUI classes tree here

### GButton: a class that creates a press button for user interface.

### GButton.java

### GDivider.java

### GDropdown.java

### GGraph.java

### GKeyListener.java

### GMenuBar.java

### GMouseListener.java

### GSpacer.java

### GSubList.java

### GText.java

### GTextBox.java

### GUI.java

### GUIComponent.java

### GUIPage.java

### GWrapper.java

## Pages:

Add PAGES classes tree here

### **About.java**

### **CreateProject.java**

### **EditAccount.java**

### **Home.java**

### **Login.java**

### **NewProject.java**

### **RegisterAccount.java**

## **UML:**

Add UML classes tree here

### AbstractProject.java

### InsulationProject.java

### Project.java

### User.java

### UserManager.java

### ProjectManager.java

# **User Story Sequence Diagrams:**

For the previous Delivelable, we had the following user stories:

## **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.**

## **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.**

## **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.**

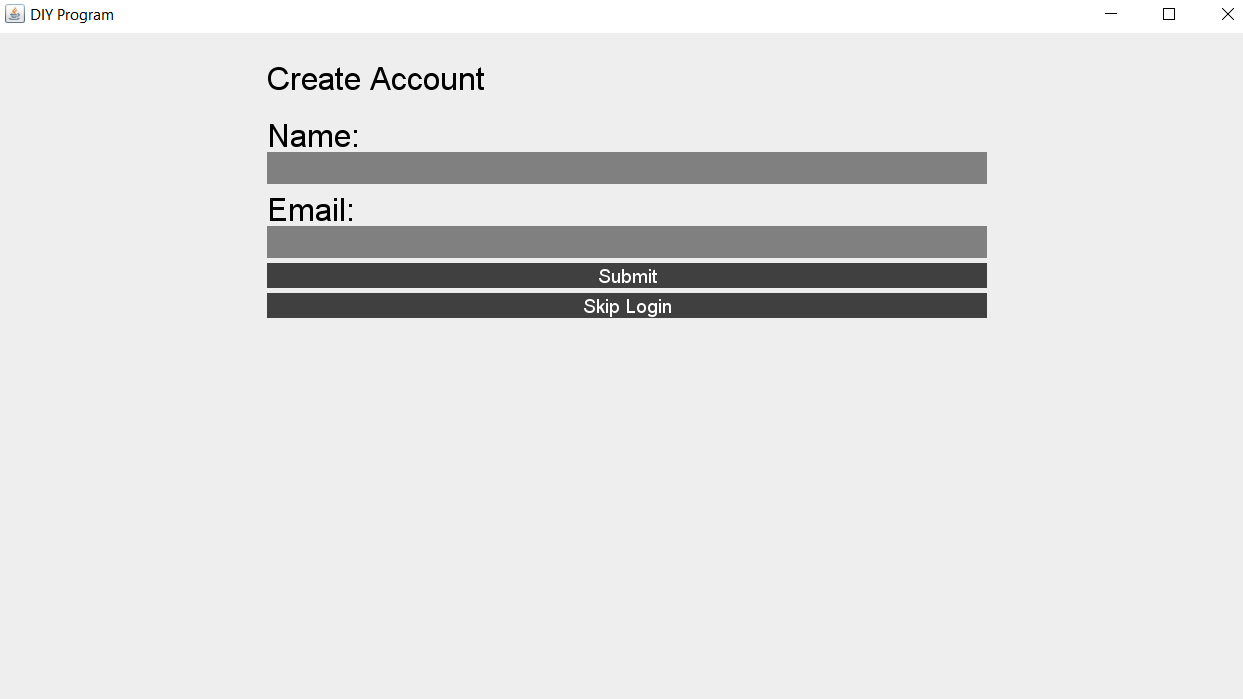
## **As a user, I’d like for the program to be portable and usable on multiple devices for easy access on the go.**

## **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.**

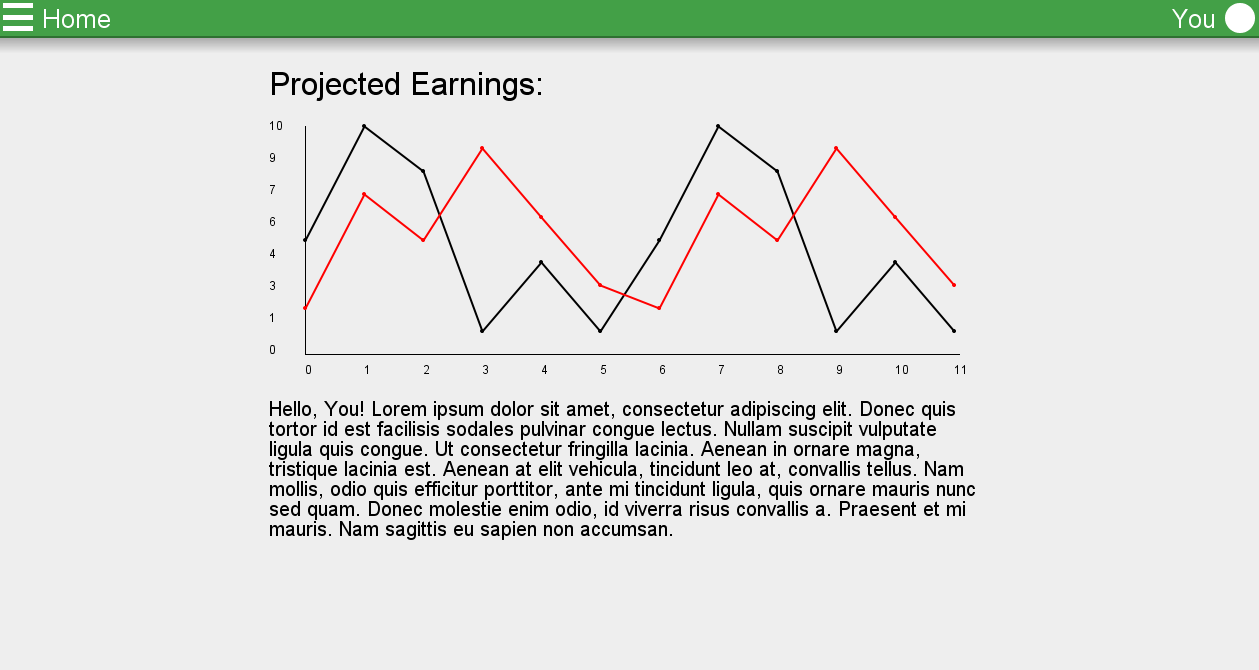
## **As a user, I’d like for the application to be accessible without a data connection while using it from a mobile device for ease of access.**

## **As a user, I’d like for the program to able to let me edit current projects for small changes or errors I could have made in the initial setup.**

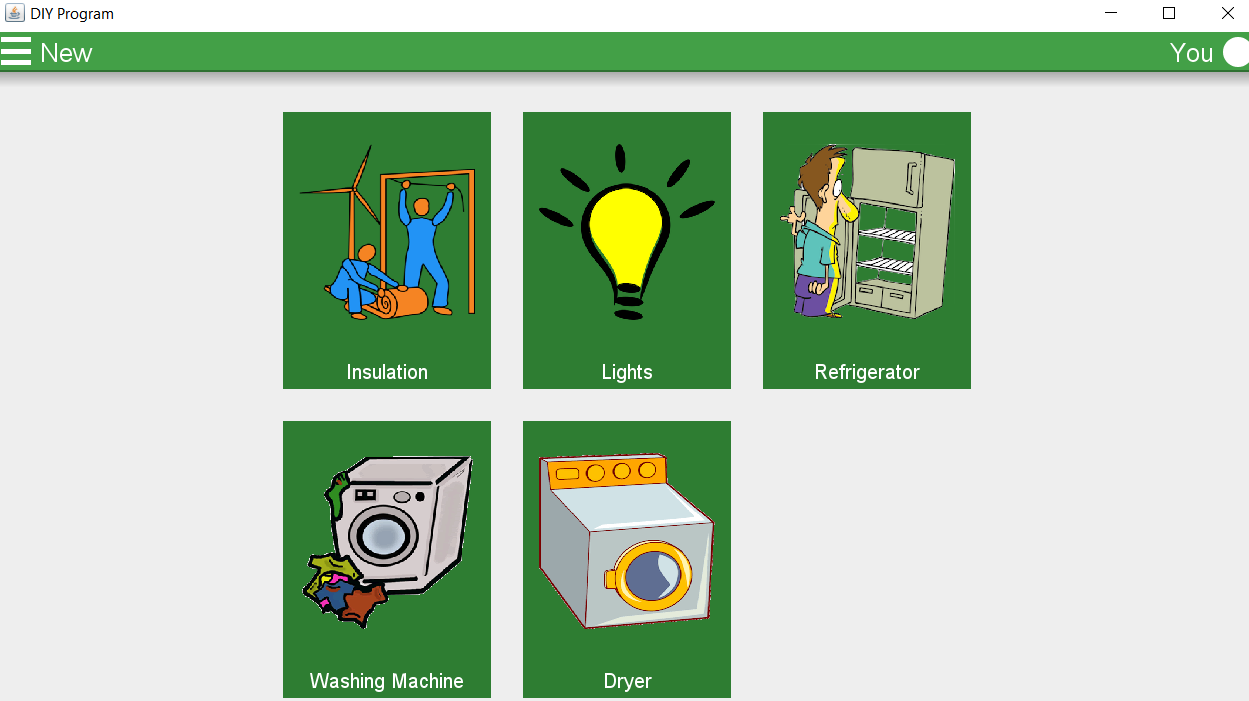
We currently have six high priority user stories that we are trying to achieve, most importantly is to have user friendly that achieve the mentioned above. The project, as of its current state, has the bones of each user story; the developers have a GUI application that starts up with user info, user name and email. The user email has to have the sign ‘@’ and ‘.com’ included, see below:

****

We also implemented a diagram for bill costs trend. Currently we have random points for demonstration puposes:



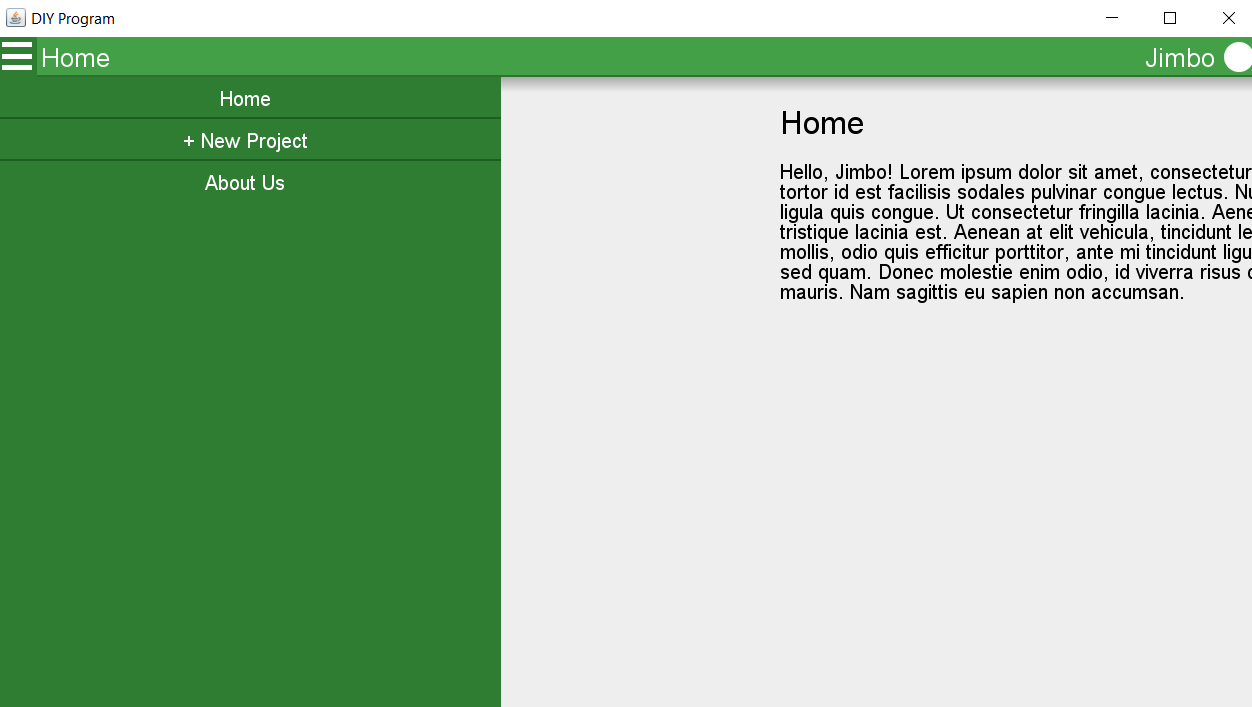
the project contains different small types of project for users to select from:



Future additions: the group plans to have layouts and implementations of each type of project.

# **System Sequence Diagram for startup:**

The user logins in using user name and email, which has to be legit email containing ‘@’ and .com. The user has a choice of skipping the registration process and create projects. Then the registered user should have previous projects’ costs listed in a trend diagram. On the left side of the screen, the menu displays few other options: Home, New Project, About.

****

In New Projects, the users have an option to choose from the following project types:

## Insulation

## Lights

## Refrigerator

## Washing Machine

## Dryer

Then enter the appropriate inputs that are restricted by the developers’ implementation (currently only bones have been implemented, no logic yet). The About page has information about the group and the members contacts.

Future additions: implementation a History page that obtains the old project for registered users only.