|  |  |  |
| --- | --- | --- |
| **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 resides 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. Control: 8](#_Toc482642162)

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

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

# I. General Project Summary:

**The DIY Energy Saver project is an application written in Java by The Pirates. The project aims to help users to complete DIY 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 a few choices over others.**

**A. Java as a 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 team to improve in JavaScript, which members lightly used in the past. Java is an advanced language, and all members (and class) have taken Java classes. There is also easy access to Java troubleshooting problems through the internet.**

**B. GUI implementation from AWT: .awt is a Java library that consists of many user interface classes. 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 custom buttons, labels, menus, and pages that would fit with the purposes of the final product.**

**C. The GUI and control separation: both GUI and control are separated into two different packages. This helped us to be more organized and gave us the freedom upon debugging the code.**

**D. Using interfaces/abstract classes in UML: having interfaces and abstract classes in UML makes the code clean and comfortable 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.**

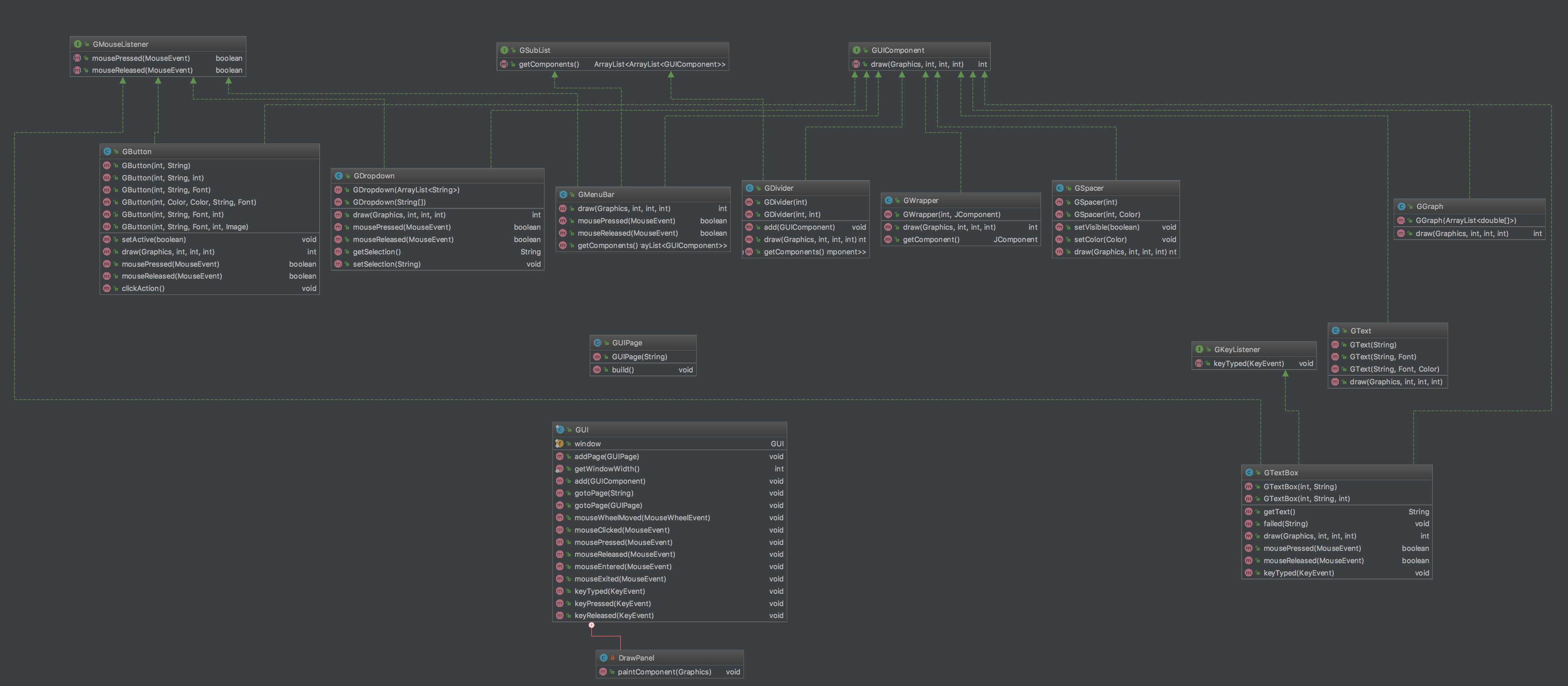
**E. Separation of GUI classes: we have separated each GUI component into a class: buttons have GButton, Pages have GUIPage. This allows programmers to change those components in a neat way; no need to destroy the whole GUI to change the color of just one label. Also, each project type will have its own class. Previously, all pages were created in the 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.**

**F. Future addition, more classes: after hearing from different groups, the group decided to add more classes to ease the users’ interfaces and make the application more efficient. For example, the group will add a class a for energy calculator, using data from Washington state website, to create a more reliable project.**

# II. 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:



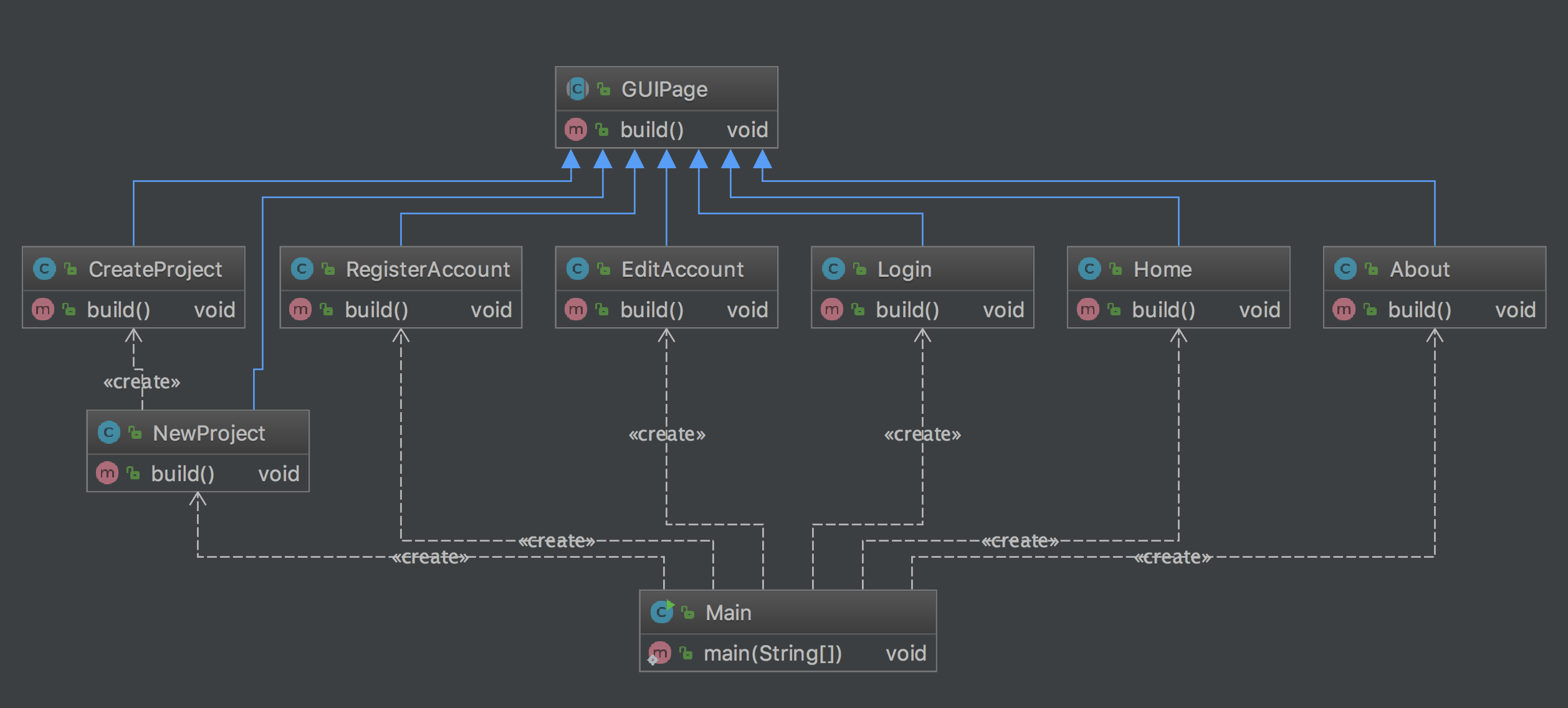
### Full resolution image since Word kills the quality: <https://imgur.com/a/MI1Ll>

Since we wanted our program to both look and feel like a mobile application we made the decision to make most of the GUI components ourselves rather than use the desktop oriented Swing components.

We decided to design a framework that would have components that would both automatically scale elements and reorient them to both look good on a small smartphone screen and a big TV.

Along with that, since we are drawing the components manually, we can add fancy animations and other visual effects that would not be as using working around Java’s Swing framework.

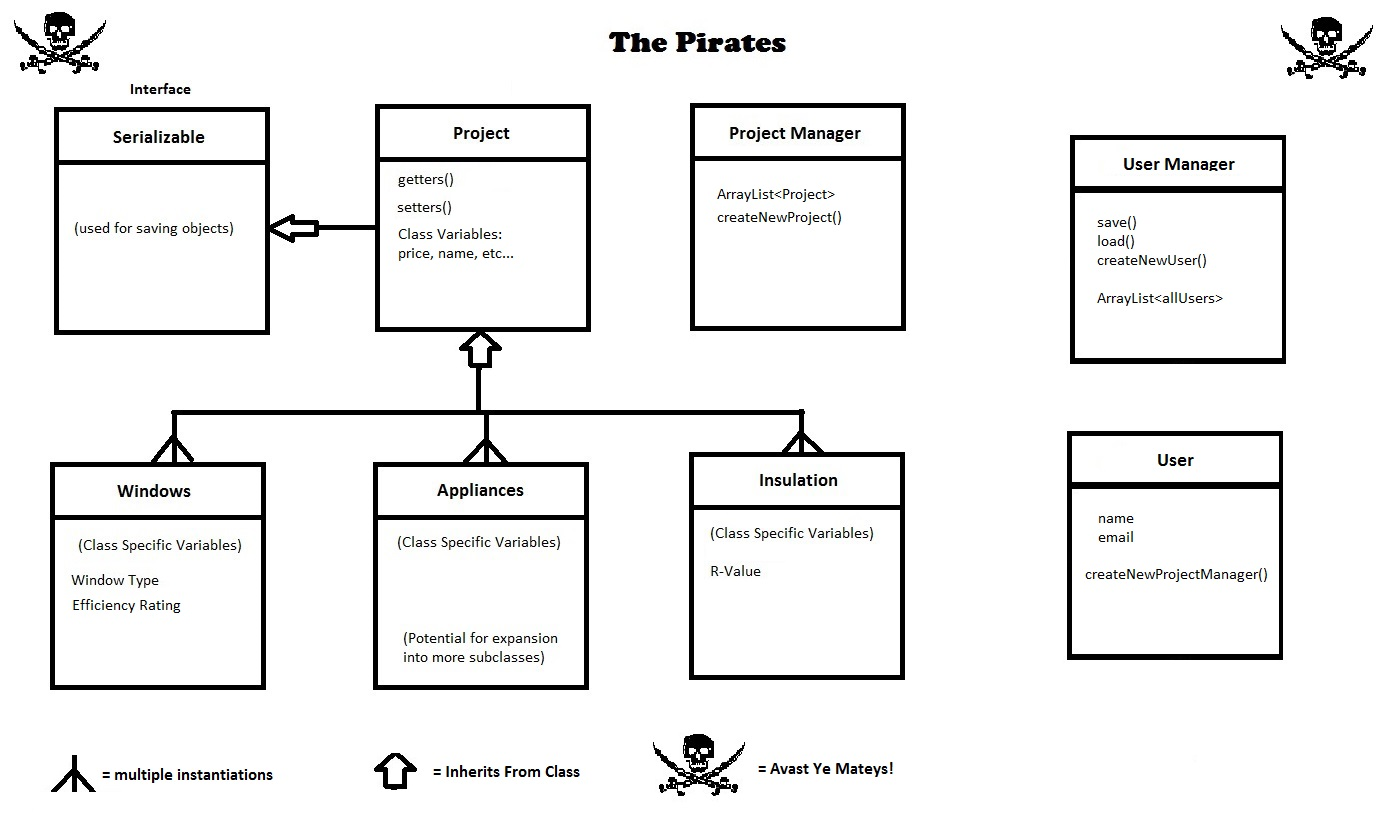
Pages:



**Our GUI system uses pages to differentiate what the user is looking at. Instead of using multiple windows, our application only uses one, dynamically switching pages to change what is on the screen.**

**A page defines what will be shown on the screen and defines what methods will be called when, for example, the user taps on a button. Pages act like the bridge between the GUI itself and the control package. When comparing to the MVC model, GUI would be the view, pages would be the model, and control is the control.**

## Control:



### Project.java

### InsulationProject.java

### WindowsProject.java

### AppliancesProject.java

### User.java

### UserManager.java

### ProjectManager.java

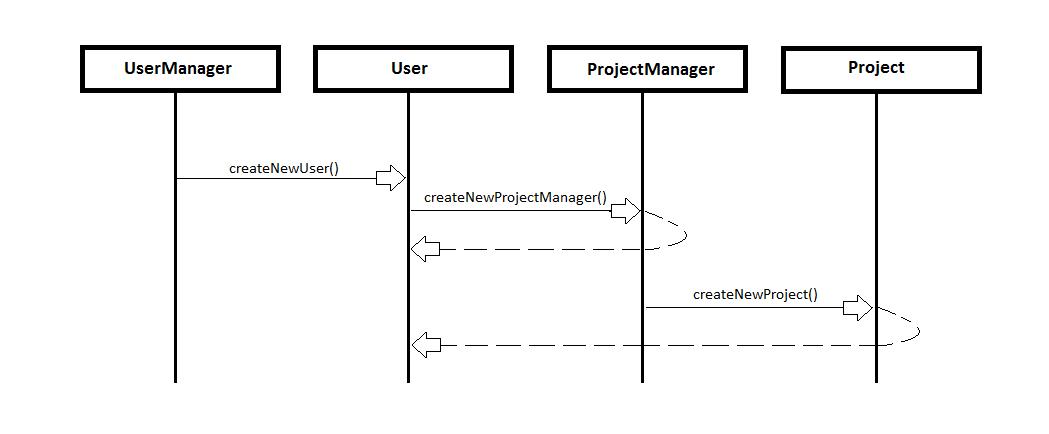
#### 

#### III. User Story Sequence Diagrams**:**

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

* 1. As a user, I'd like for the program to give me the option to create an account with the capability of managing multiple projects at once.
  2. **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.**
  3. **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.**
  4. **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.**
  5. **As a user, I’d like for the program to be portable and usable on multiple devices for easy access on the go.**
  6. **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.**
  7. **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.**
  8. **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.**

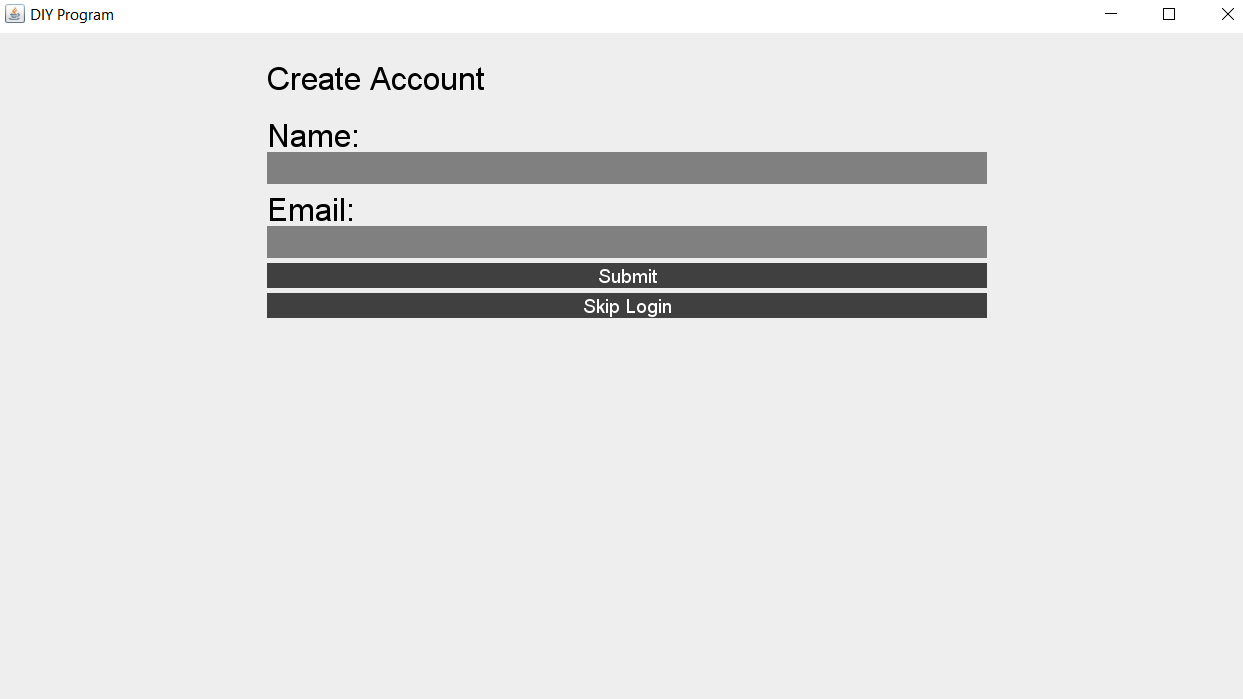
**User Story A: Sequence Diagram**



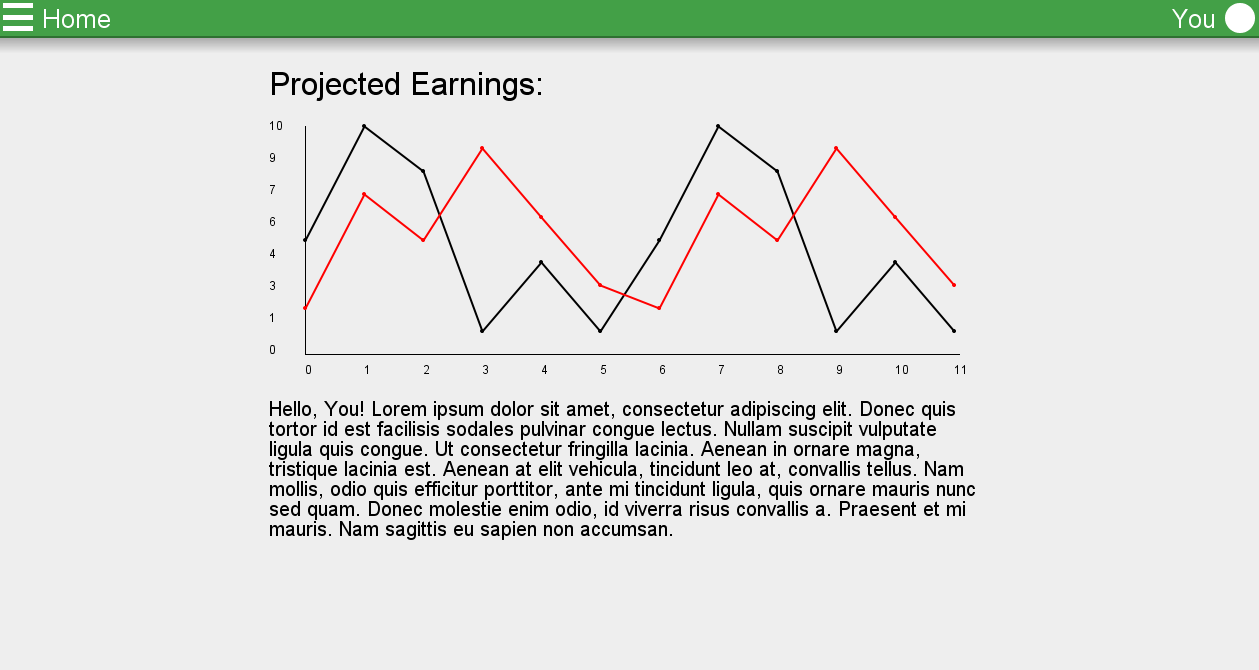
For this user story the program is creating a new user from the UserManager class. After this step has been completed, the user may then prompt the program to make a new project, which in turn triggers the internal ProjectManger class to instantiate a new project object. This new project object is stored inside the ProjectManager object which is a part of the User class itself and can be used for later purposes as well as exported for future use.

Start Up Sequence Diagram:

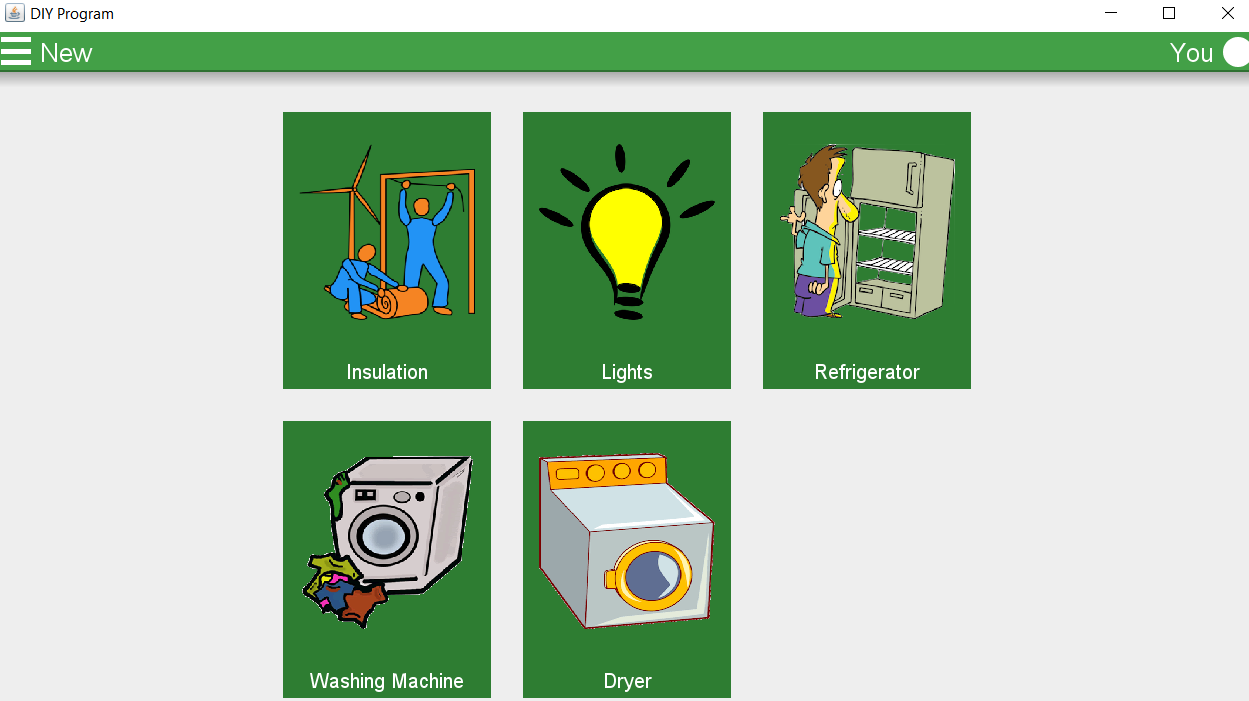
We currently have six high priority user stories that we are trying to achieve, most importantly is to have the program as user friendly as possible, which is mentioned above. The project, as of its current state, has the bones of each user story; the developers have a GUI application that starts up 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 purposes:

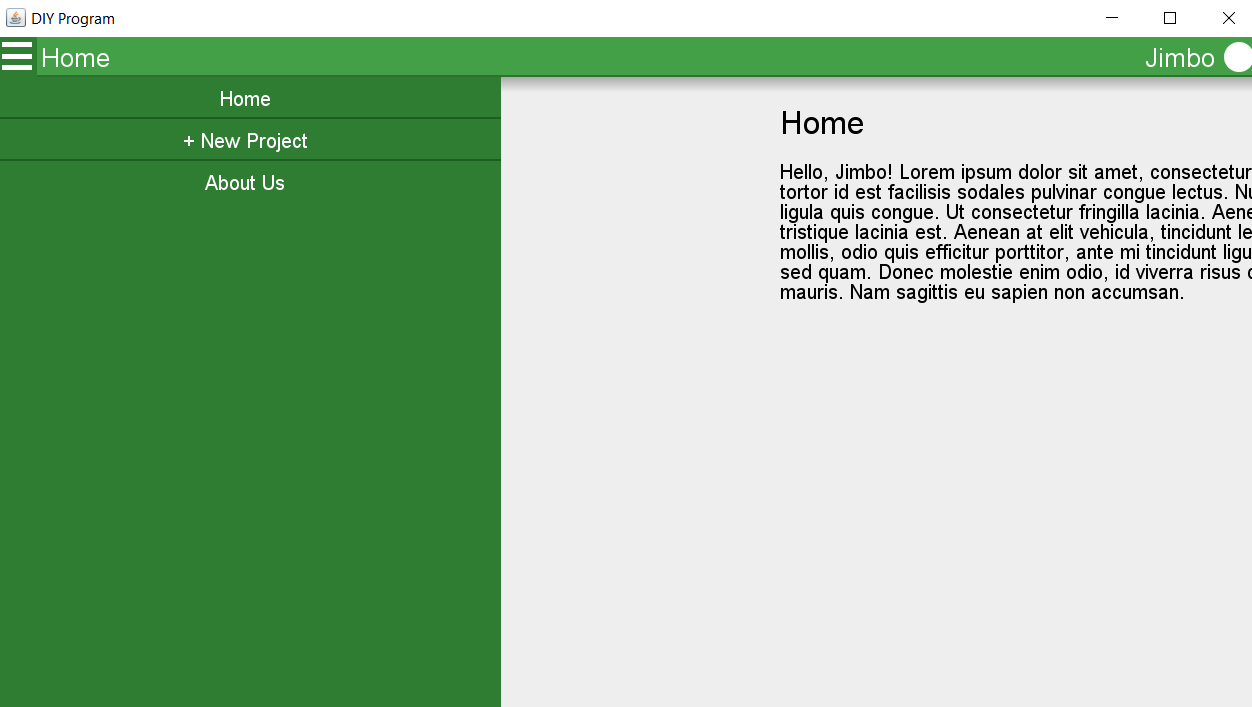


The project also contains different small types of project for users to select from:



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

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.