Design Document

Dirty Water

University of Washington Tacoma
Jeffrey Weiss
TCSS 360

https://dirtywater.github.io
dirtywater365@gmail.com
https://discord.gg/GM3xgYg

TEAM:
Michelle Brown
Jim Phan
David Guerrero
Caleb Wheeler

Introduction

The Software Design Document is a document to provide documentation, which will be used to aid in software development. The documentation provides sufficient information to proceed with an understanding of what is to be built and how it is expected to built, by providing the details for how the software should be built. Within the Software Design Document is a design rationale, graphical documentation of the software design for the project including a class diagram, and various sequence diagrams. This Software Design is focused on the base level system and only critical parts of the system.

Table of Contents

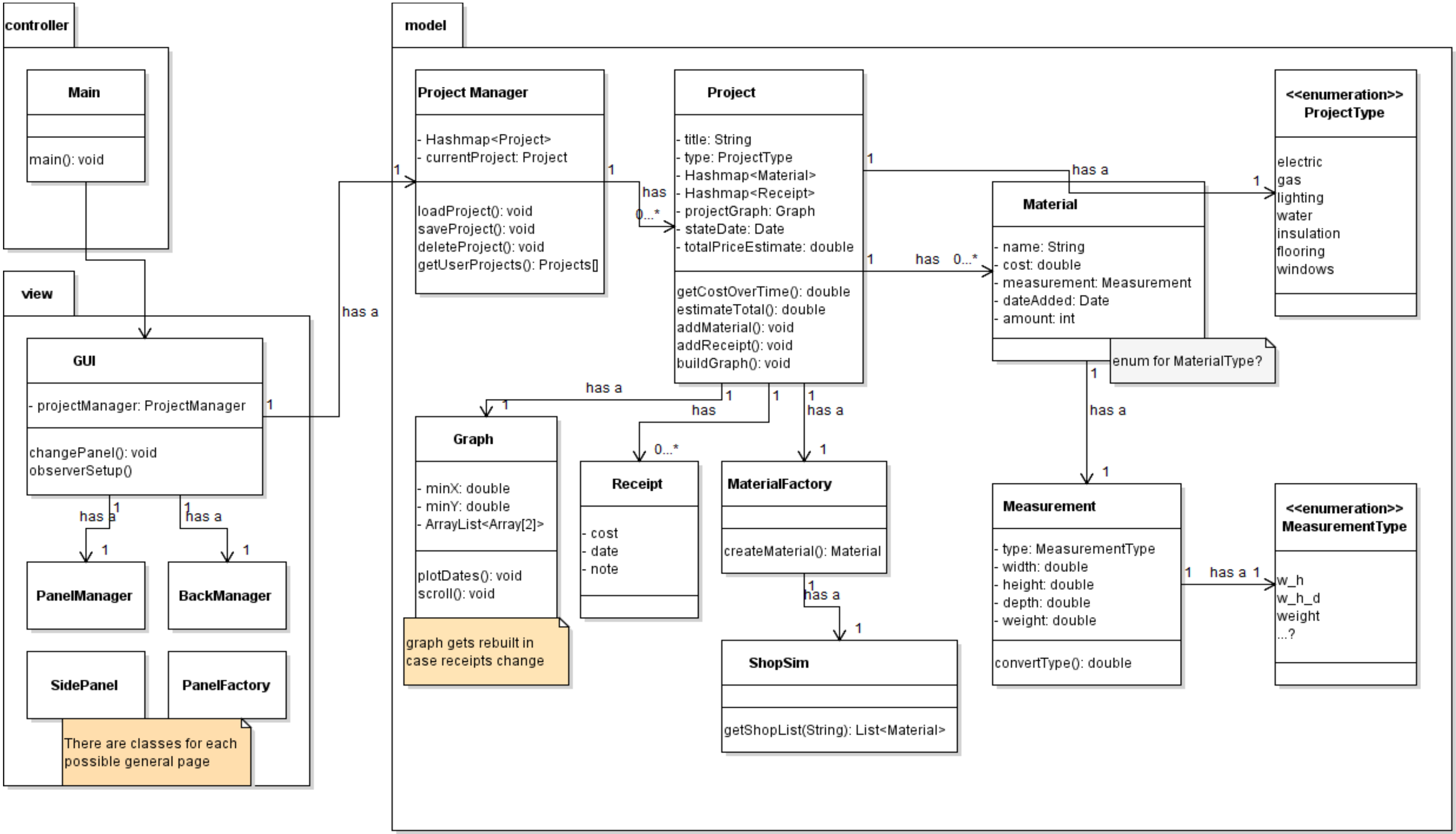
Rationale Summary

Our design for this project follows the MVC pattern. It's the easiest form to follow and allows us to separate the classes into different sections based on what we need them to perform. The project's main focus is to have data be stored in the Project class which is managed by the ProjectManager. This is used to allow us a gathered location of data the user inputs and customizes as the user will need to store that data into a text file in between sessions. The Project Manager exists as to allow the user to control multiple projects at one time.

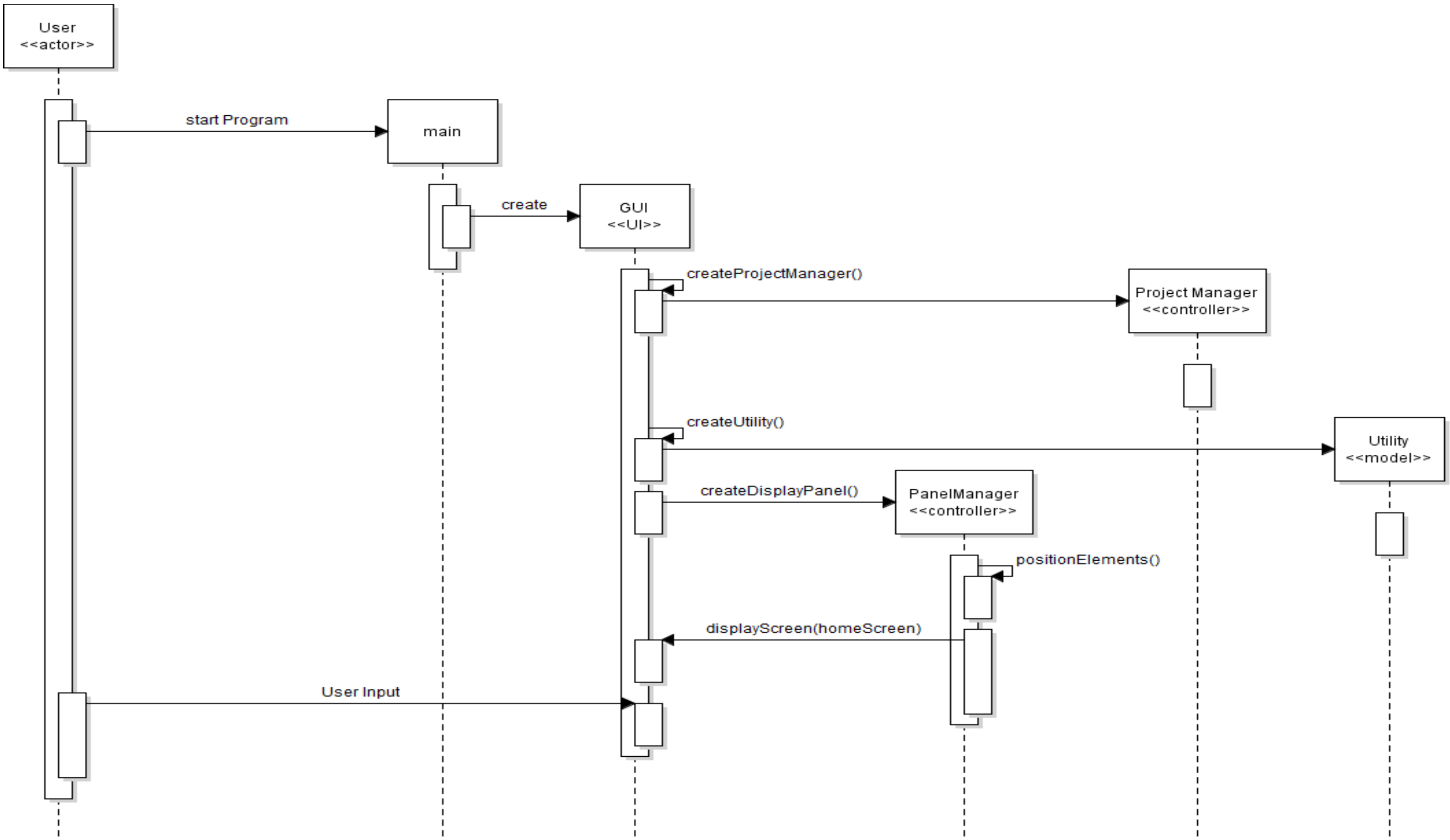
When applying our design to the Object-Oriented Programming Heuristics, we can examine if our design follows these heuristics. Examining if the project follows uniformed work, we can say most of the system does so. The class that may perform more work would be the Project class as it is used by other classes due to its nature as being the class that stores the user's data at run time. Examining the second heustritic design, we separated the classes as not to create god classes however we did design classes with 'Manager' in their names. This is a design choice as the Manager classes are necessary in order to allow for the changing of the panels the user can view and for the multiple data the user can access.

The majority of this design follows the ten heuristics given to us. While some can be suspect such as our naming conventions of the 'Manager' classes, the rest keep true to our MVC pattern. We have had to remove some classes as other classes would allow for a simplified version of this project. We haven't come into issues with having to remove other classes but the design seems to hold on its own. Overall, the project's design keeps true to the ten heuristics we hold if not a few suspect issues.

Class Diagram

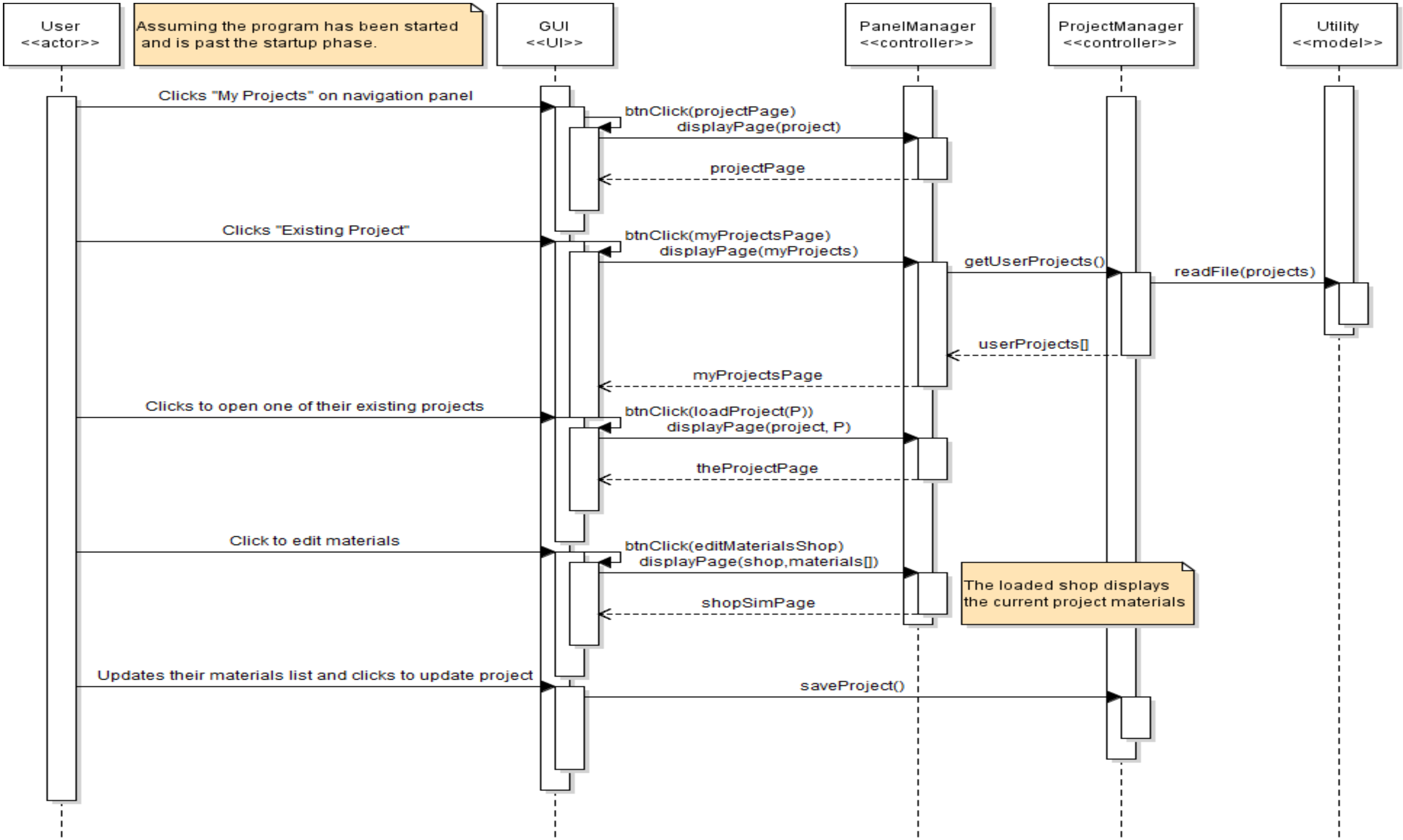


Startup Sequence

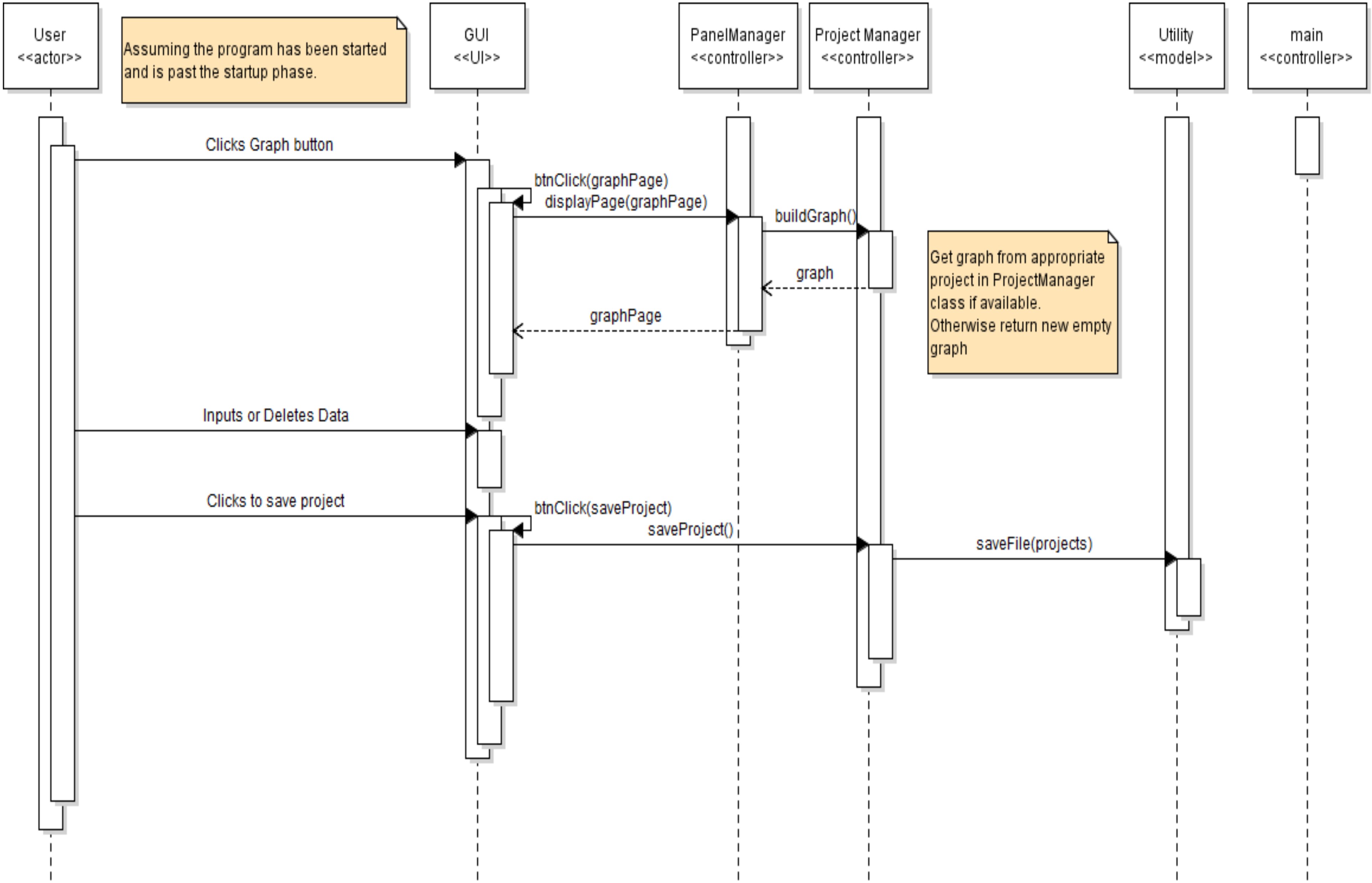


User Story Sequence Diagrams

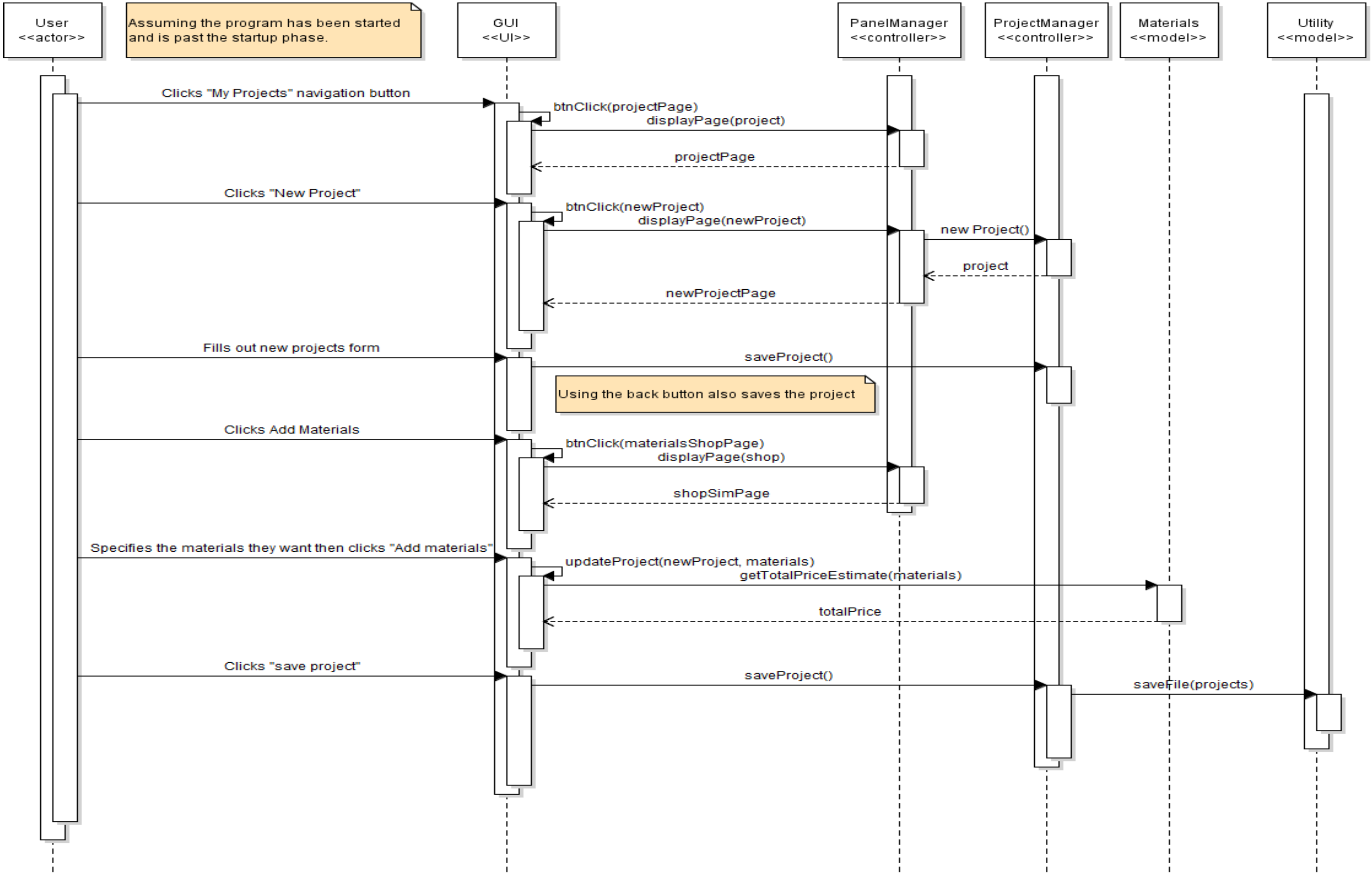
US01. As a DIYer, I want to store data and measurements for my DIY projects so I can refer to them any time.



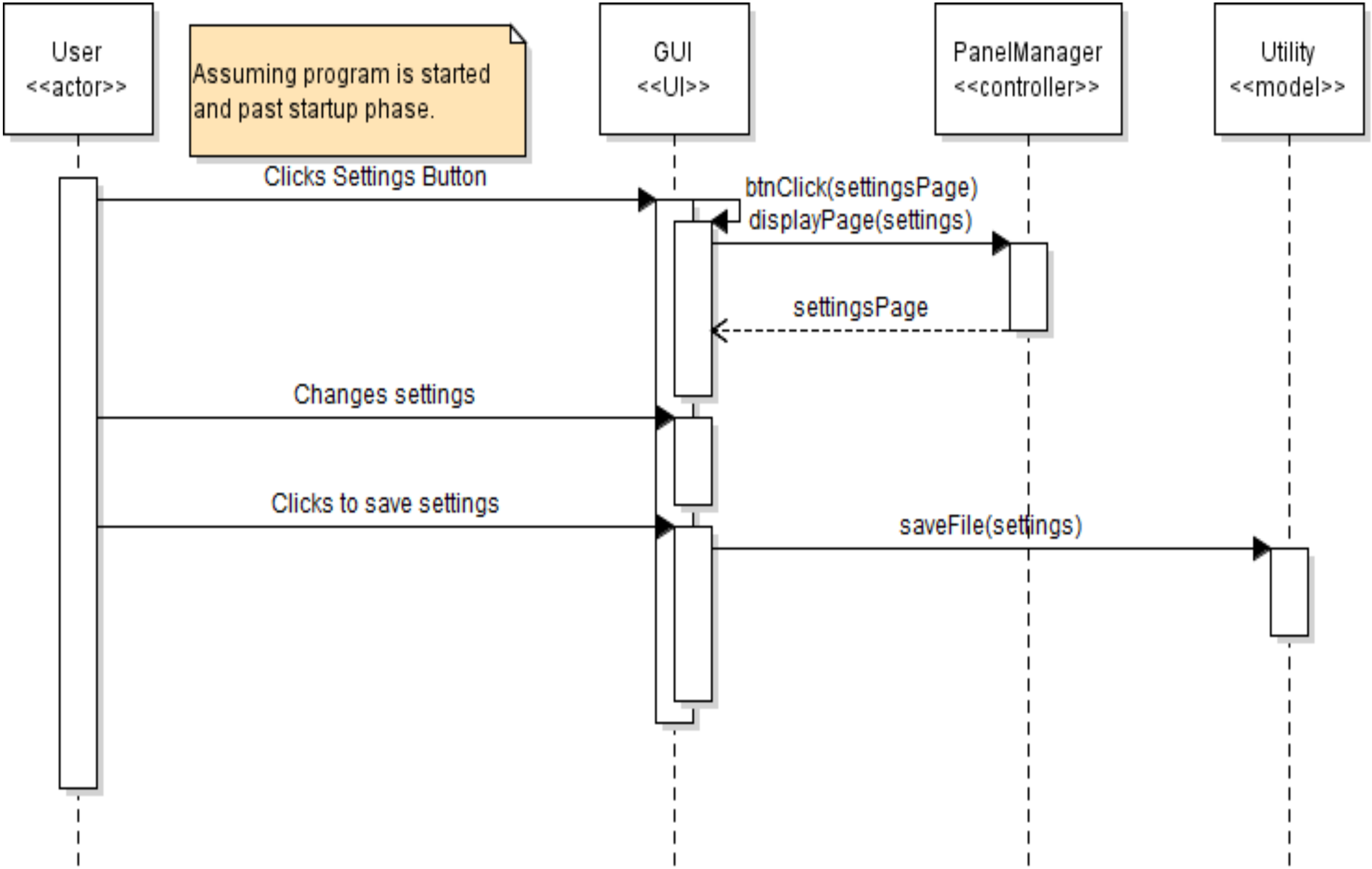
US02. As a homeowner, I want to document my home's energy efficiency.



US03. As a user, I want to make price calculations based on the data I assemble so I can estimate project costs.



Extra. As a user I want to export settings for synchronization to other devices.


User
<<actor>>
Assuming program is started and past startup phase.
GUI
<<UI>>
PanelManager
<<controller>>
Utility
<<model>>
Clicks Settings Button
btnClick(settingsPage)
displayPage(settings)
settingsPage
Changes settings
Clicks to save settings
saveFile(settings)