Design

Study Calendar Application

Pierre Drego 301301848 pdrego@sfu.ca
Gurkiran Kaur Brar 301274688 gkbrar@sfu.ca
Tyler Bailey 301346936 tjbailey@sfu.ca

Table of Contents

1. High Level Architecture

Our high level architecture designs for the calendar app have remained the same throughout development, and mirror the design present in the interim report.

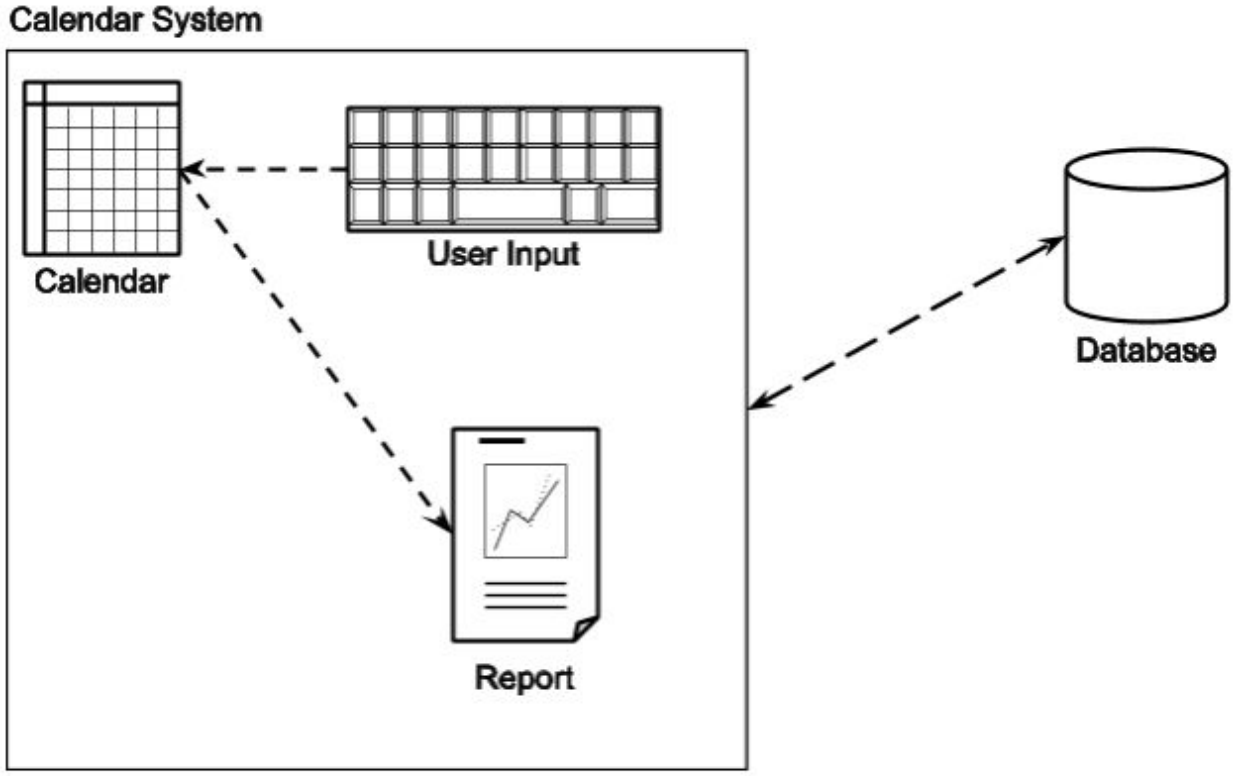
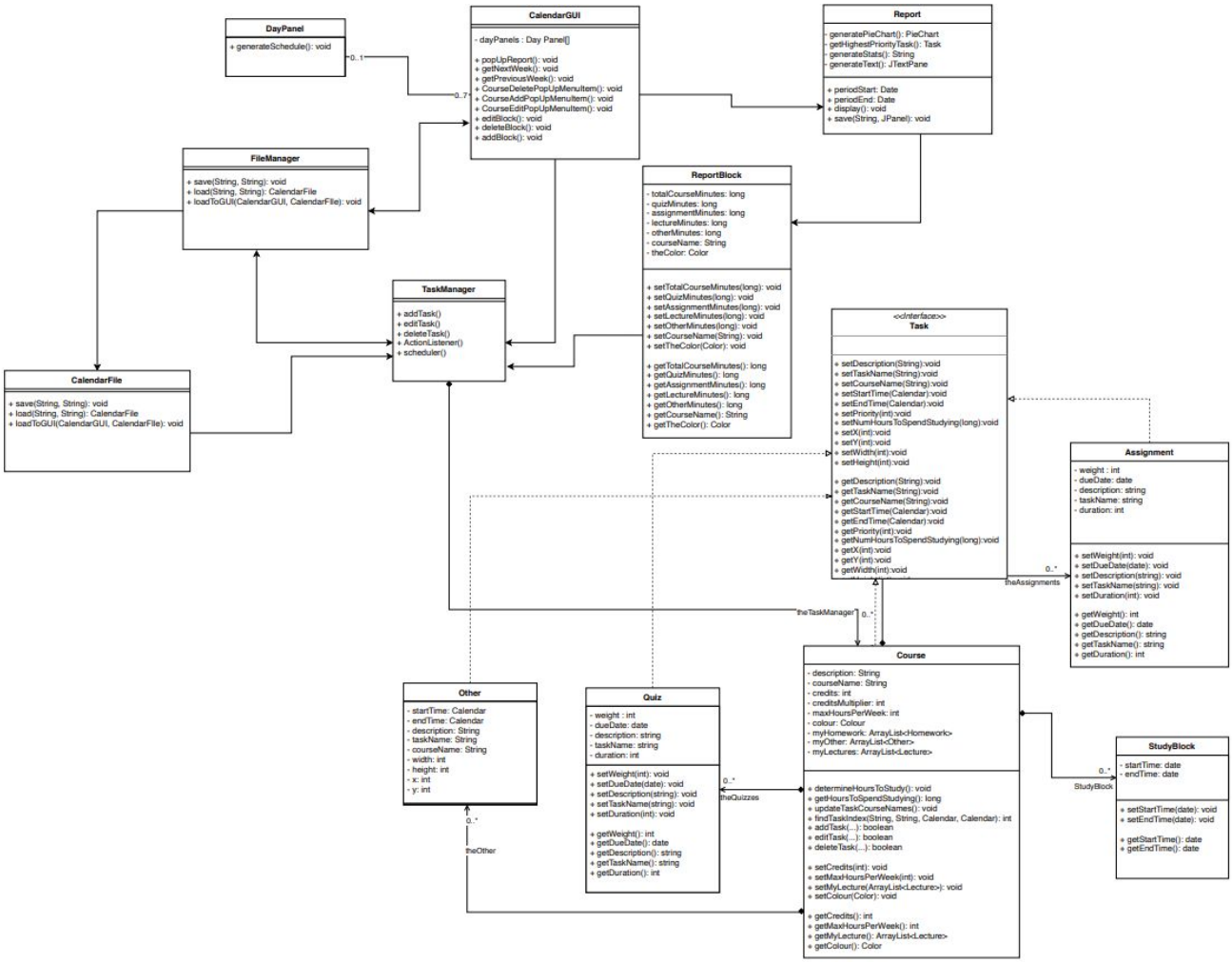


Figure 1: High-level design for study calendar app

2. Detailed Design

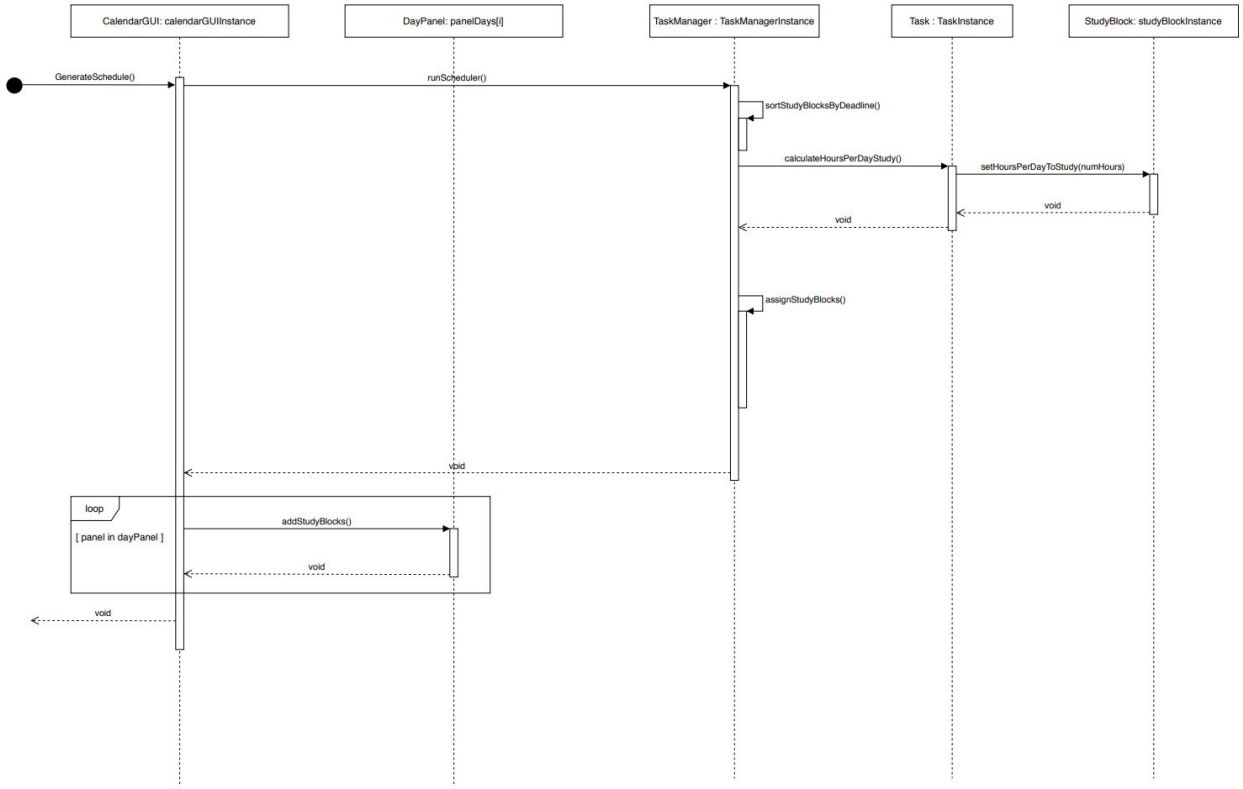
1.1 UML Class Diagram

1.1.1 Class Diagram



1.2 UML Sequence Diagram

1.2.1 Schedule Generation Sequence Diagram



1.3 Design Justification

We chose to implement the calendar in the way we did due to the network of specific objects which access each other constantly, and some which are independant. The basis of our design is to separate our GUI elements for the calendar from the implementation of the more complex features such as scheduling and file saving. In addition, modifying the underlying algorithms which manipulate the objects displayed on the GUI can be simply modified in “TaskManager” without having to worry about these changes impacting the display. This decoupling allowed us to avoid conflicts which may have occurred in alternative implementations of such an application.

1.2.1 Utilized Design Patterns

The only notable design pattern we used in our project is the Façade pattern. This is present in the implementation of “Task” since we have several classes such as “Lecture”, “Homework”, and “Other” implementing this interface, which is created by the “TaskManager” class. We decided to use the Façade pattern for implementing these features, because the task related functions are easier altered by an exterior class due to their complexity when working together. Using the task manager to orchestrate the creation, deletion, and so forth, of the task objects

made managing them easier, and allowed us to mask a majority of the “insides” of the task objects.