Specifications Document

Team 2 - CS374 - Fall 2014

In this document, we describe the specifications for our Schedule Conflict Calculator application.

Table of Contents

1 Requirements

Our Schedule Conflict Calculator Web application will function as an extension to other academic administration technologies. Given a set of inputs related to moving a course from one time block to another, the application determines the number of student scheduling conflicts that would arise in some number of move scenarios. The application handles a number of different move scenarios, allowing the user to decide which scenario to test.

2 Use Cases and Scenarios

The user is allowed multiple scenarios for calculating schedule conflicts. For example, sometimes the user has a single *section-change description* in mind and would like to see scheduling conflicts. In other instances, a user might know that they need to move a section to another time and/or room, yet would like to know which room out of a set of possible candidates has the least number of conflicts.

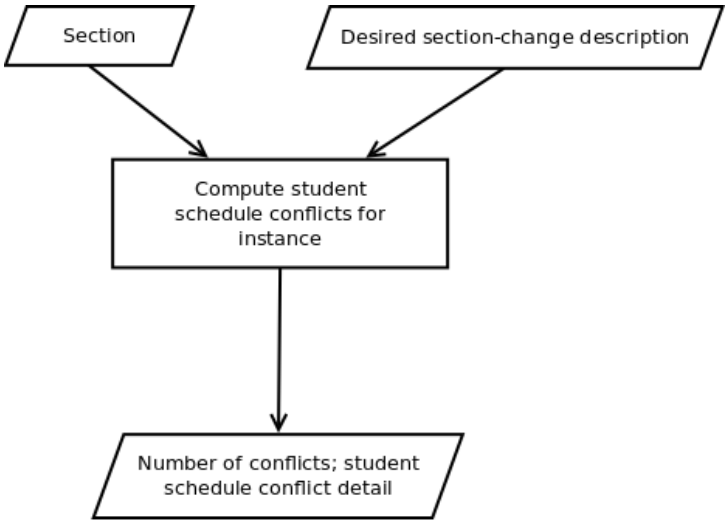

Figure 1: Describes a simple scenario for finding conflicts given a single section-change description

In the simple case, we consider a single section-change description supplied with a desired input section. Output will consist of the number of conflicts as well as a detailed look at individual conflicts. If another section is already *associated* with the section-change description, the user is informed of the conflict but is still provided the normal conflict output as if there weren't a course currently associated with the description. Other input parameters might be needed to provide a more customized operation tailored to meet user needs.

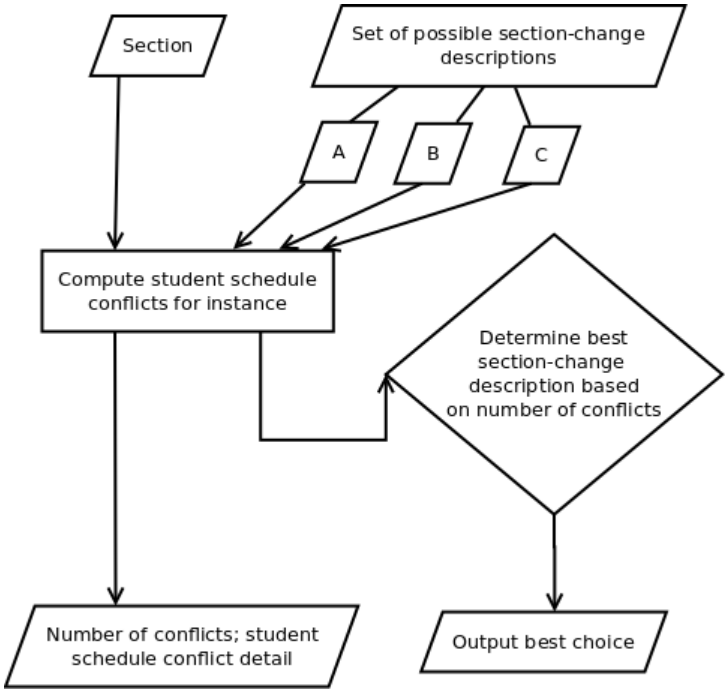

Figure 2: Describes a more complex scenario for finding conflicts given multiple section-change descriptions

Other scenarios are needed for more complex operations. A user might wish to provide a set of section-change

descriptions to apply to the input section in order to find the best change description. This set of descriptions may encompass multiple rooms and buildings and also take into consideration moving courses already associated with section-change descriptions. This introduces added complexity.

To deal with this complexity, we introduce the terms *locked-section* and *floating-section* (see glossary). In the proposed scenario (Figure 2), a set of section-change descriptions is provided into which the input section may be moved. In a simple case, the scenario given in Figure 1 can be applied to each description. The description(s) with the fewest conflicts are then presented as output. However, sometimes more than one section can be moved within the sections already associated with each section-change description. Each one of these sections is either marked as a *locked-section* or a *floating-section*. A locked-section is one that cannot be moved. This automatically disqualifies the description to which the locked-section is associated (conflicts are still computed as in the Figure 1 scenario). A floating-section may be moved within the set of section-change descriptions.

3 Data Flow

This section describes the interaction of various modules within our application. We map the flow of data through the different modules of the application, demonstrating how it is used and transformed.

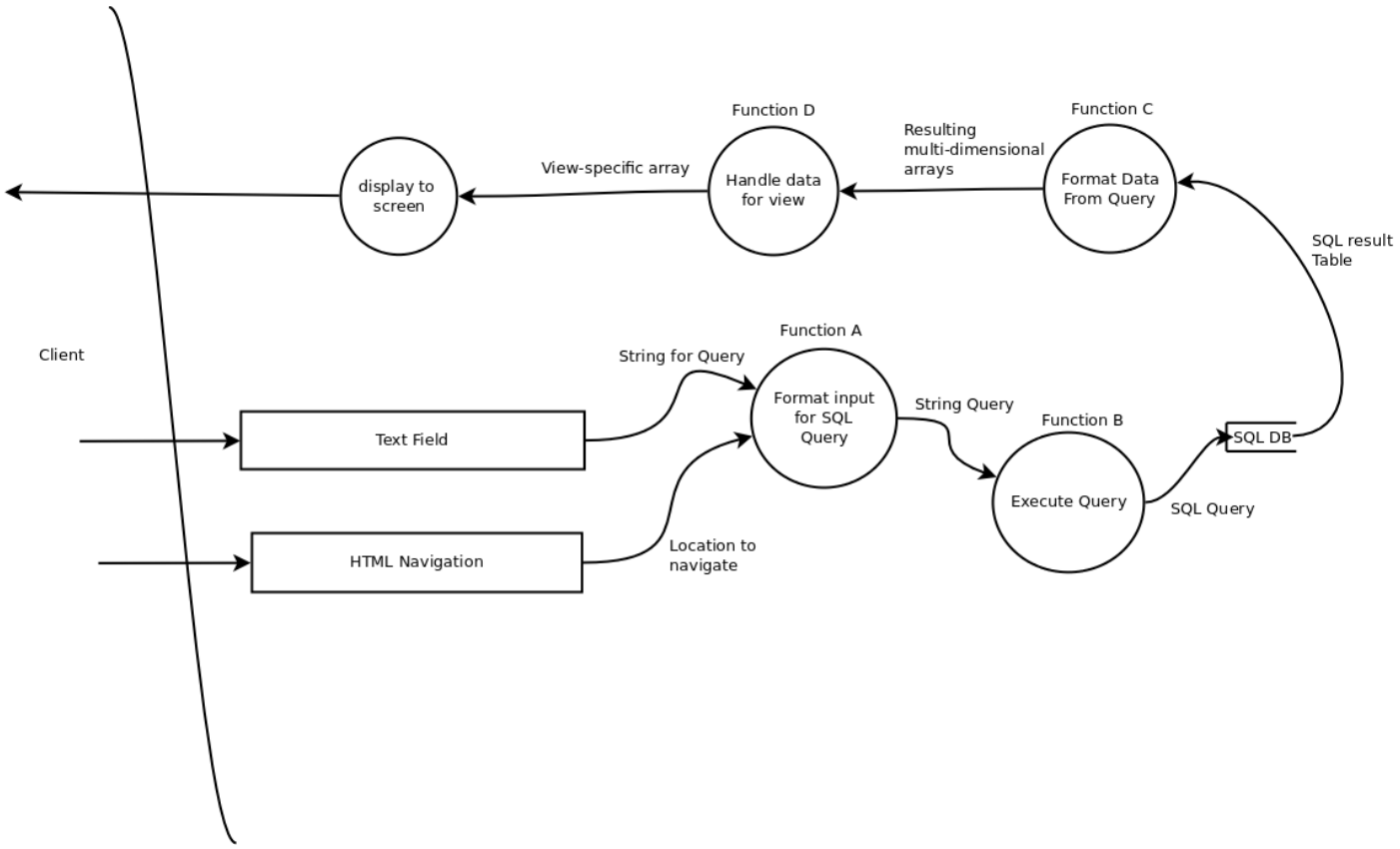

Figure 3: Data flow diagram for main application operation

In each proposed scenario, a user provides input through an HTML form which is passed to a data handler. The data handler uses an SQL database connection to issue a query and retrieve the data needed to execute the task. The handler is responsible for producing the desired result. Different handlers will require different amounts and/or types of data and thus will make different queries. Handlers will be implemented using a class hierarchy in PHP. Each derived class will implement a different handler-kind which executes a distinct scenario.

Though each handler performs a different operation, it still functions according to the same interface. Figure 3 graphically models data as it flows from the client, is transformed and goes back to the client as output. Figures 4-6 provide in more detail the functions presented in Figure 3.

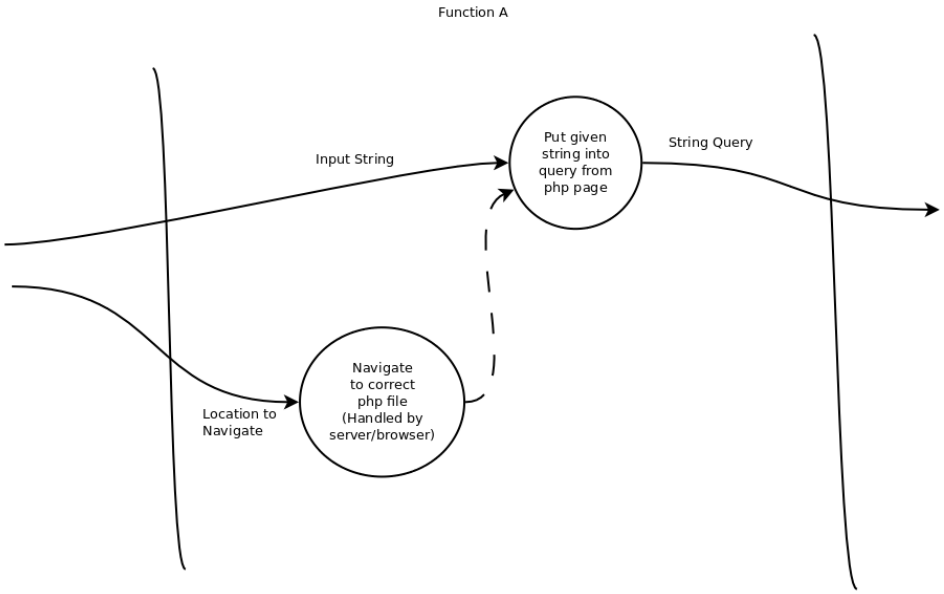

Figure 4: Details function A in Figure 3

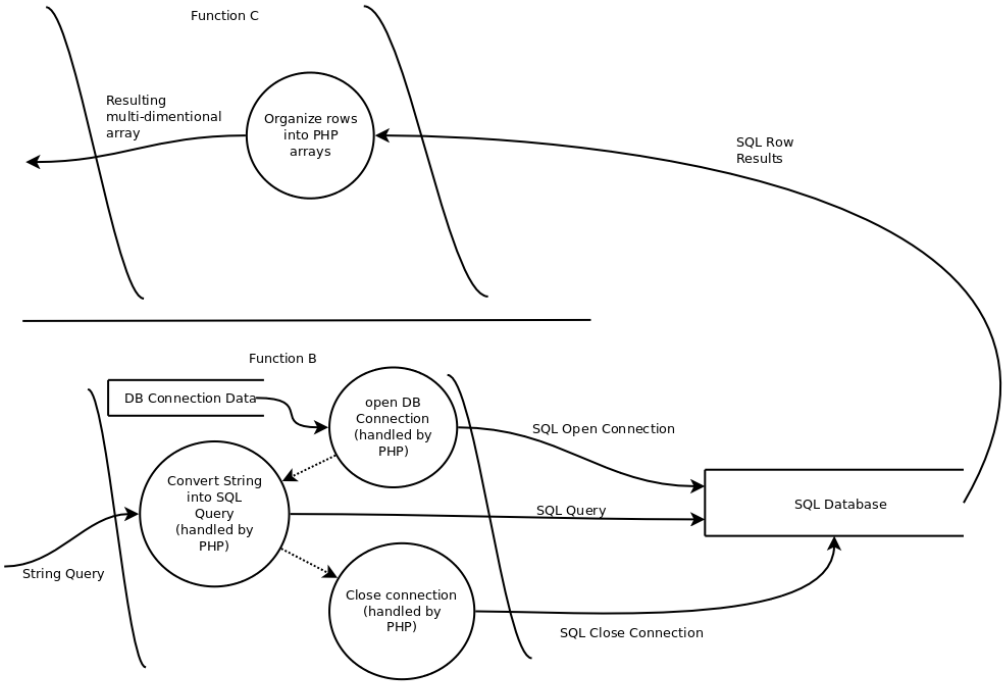

Figure 5: Details function B and C in Figure 3

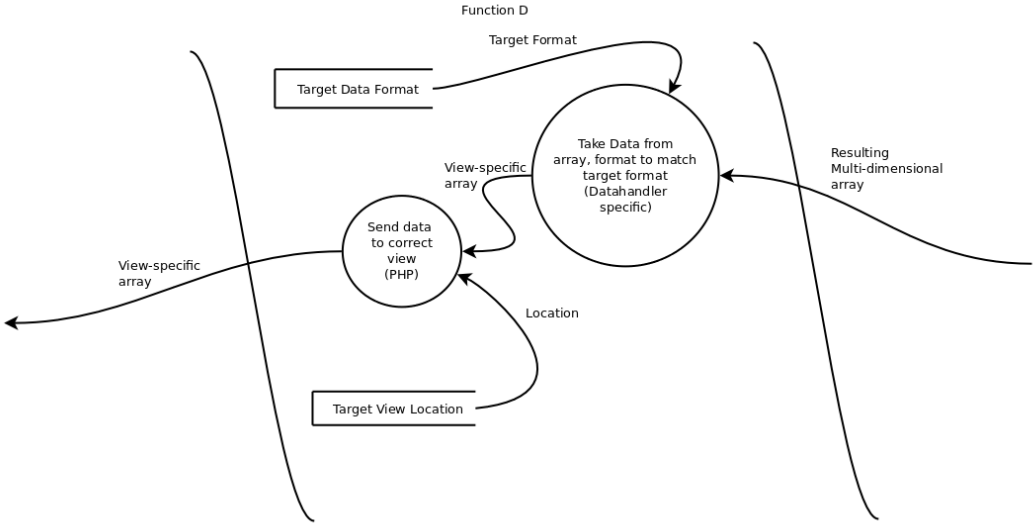

Figure 6: Details function D in Figure 3

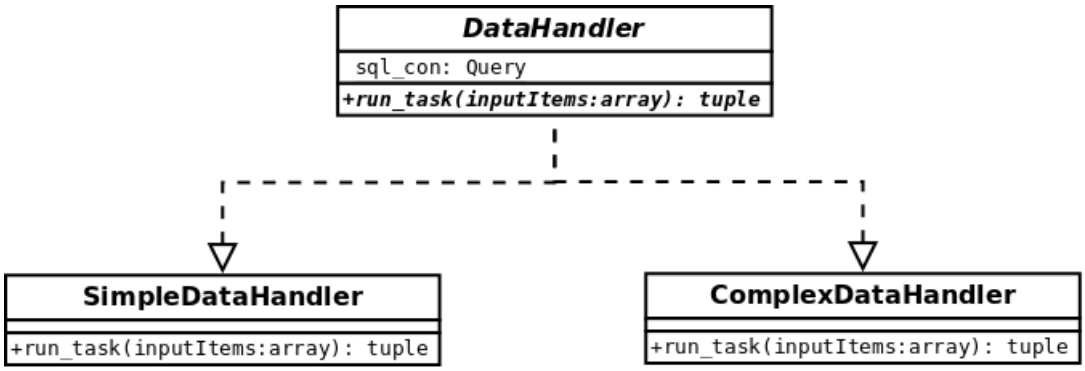

Figure 7: Simple UML concept for data-handler class-hierarchy

4 Glossary

association The set of courses whose location matches and time duration overlaps a section-change description

building A collection of rooms

course A general description of a class of sections

data-handler An object that executes a scenario using data obtained from the database

floating-section A section that may be potentially moved from either its current time, location or both

locked-section A section that cannot be moved from its current time and location

professor An instructor of a course; there is one professor per section

office-hours A time block associated with a professor; a potential conflict item to consider

room The meeting place of a section

scenario A section change operation (e.g. moving a section from one time to another)

section An instantiation of a course

section-change description A tuple that specifies a desired change to a section's time and location

student A participant in a section