After analyzing a handful of design patterns, we were able to come up with a few that occur within our project. We have a mediator, our database.js file. Chain-of-responsibility, the functions within some javascript files. Iterators in ejs files, and an observer-like process in the database.

A mediator is something that handles requests or other events and manages any process or activities. Our database.js file will take SQL queries and send them to MySQL and retrieve information or set and save values. All this makes the Database.js file, and therefore the database a mediator of sorts.

Chain-of-responsibility is a design pattern where, for example, function after function is called to get the end result. Each preceding function passes on its result to the next for its needed responsibilities. All of them relying on each other to get the job done, passing the responsibility of the desired result down the line. An example of this is the functions used in the routes for the project, or the javascript files. Not all routes may use this design, but some do. In the reservations file there are two functions, one that gets the user_id needed for insert. That function then calls the next function which will call the insert_reservation procedure.

The least surprising recurring design pattern would have to be iterators. Iterators can be so versatile for checking just about anything. In this scenario, iterators are used to go through the list of retrieved results such as records to the employee reports. This helps create a flexible, functional way to display the records when the number of rows in the report could change at any moment.

Lastly, we have the Observer. We have implemented an Event which, when running properly, will check the sites and reservation table to set those reservations that are active to be not available repeatedly at an interval. Meaning unavailable sites will generate in the Occupancy report, or on the other side, generate in the vacancy report. Should this method not work, we would use a trigger that would update the availability before or after an action happens.