Detailed Design Model and Patterns

Advisely, An Education Support Application

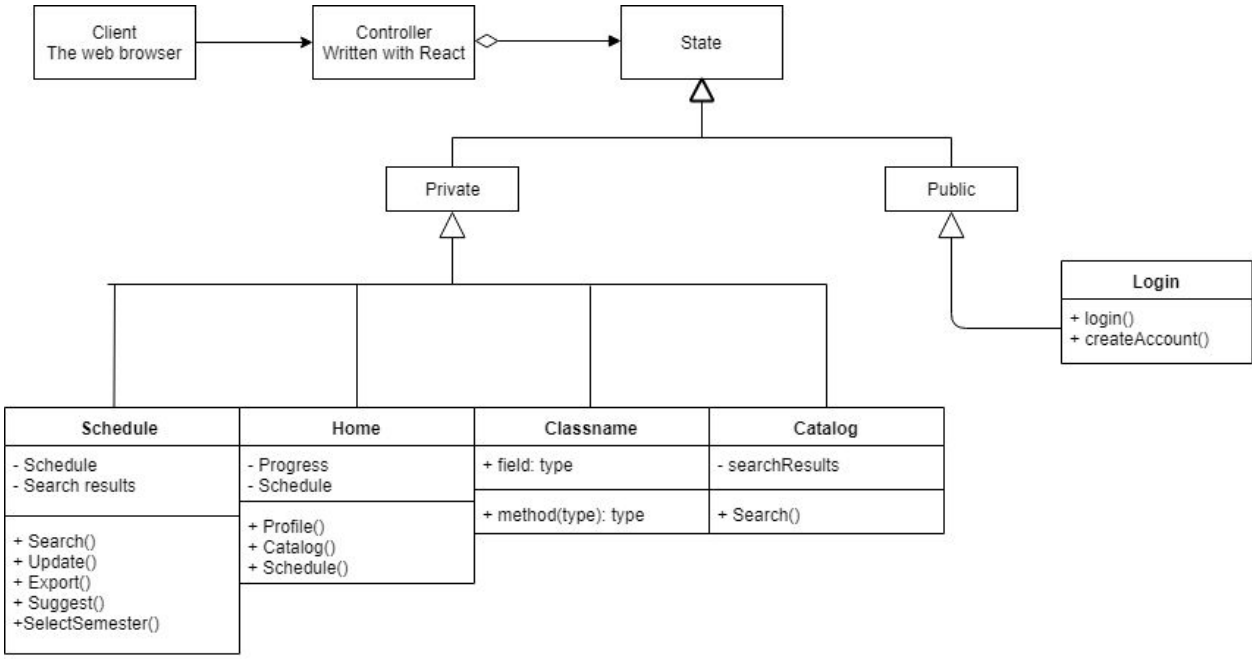
Big Muscle Boys

Sam Elliott, Jens Hansen, Connor Langlois, Eli Legere, Eric Schessler, Spencer Ward

COS 420 UMaine Spring 2019

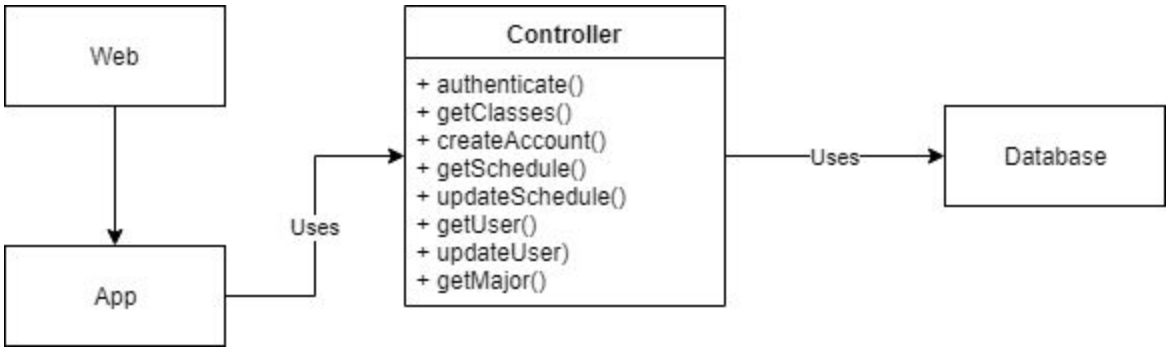
Professor: Sepideh Ghanavati

State Design Pattern



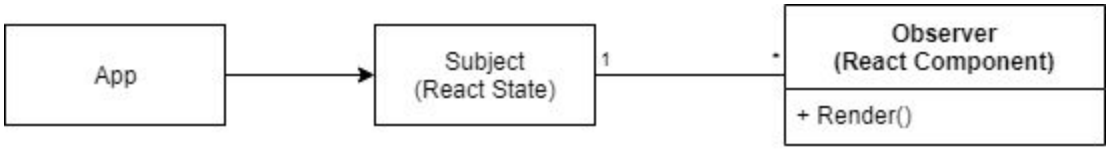
The state design pattern is used when the object alters its behavior based on the state of the program. This state design pattern works well with the React-Router-DOM in which the single page website alters what the functionality of the page is based on the state of the application. We will be using the state design pattern to account for routing between component and allow the user to access all the functionality of the application on a single page. Each state will correspond to each page that the application will include, login, schedule, home, profile, and catalog.

Front Controller Design Pattern



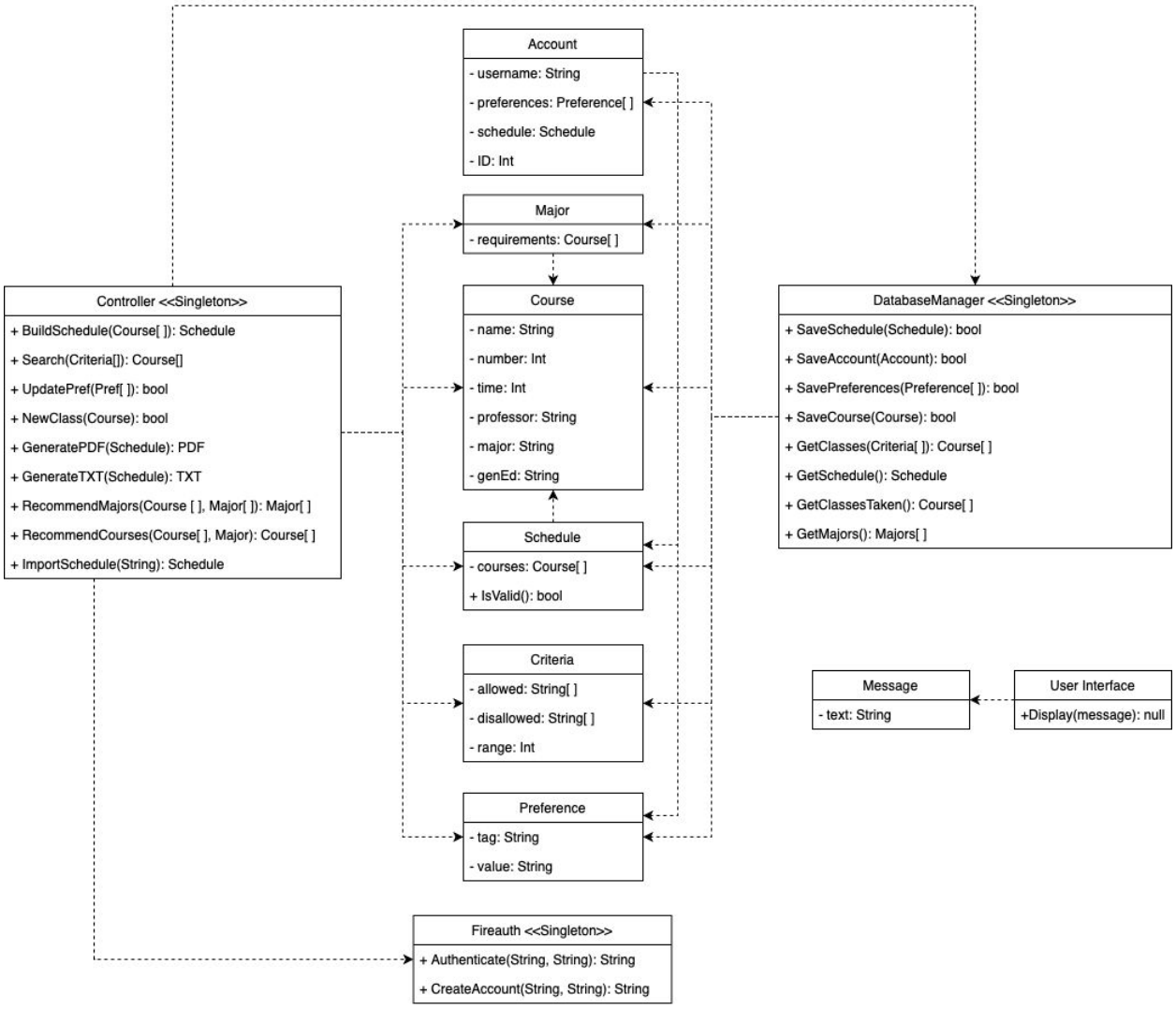
The front controller design pattern allows us to handle all incoming requests in a centralized place which can then provide the necessary components of our application with the required information and make the corresponding function calls. The application will take user input and send all input to the controller where it will be processed and distributed.

Observer Design Pattern



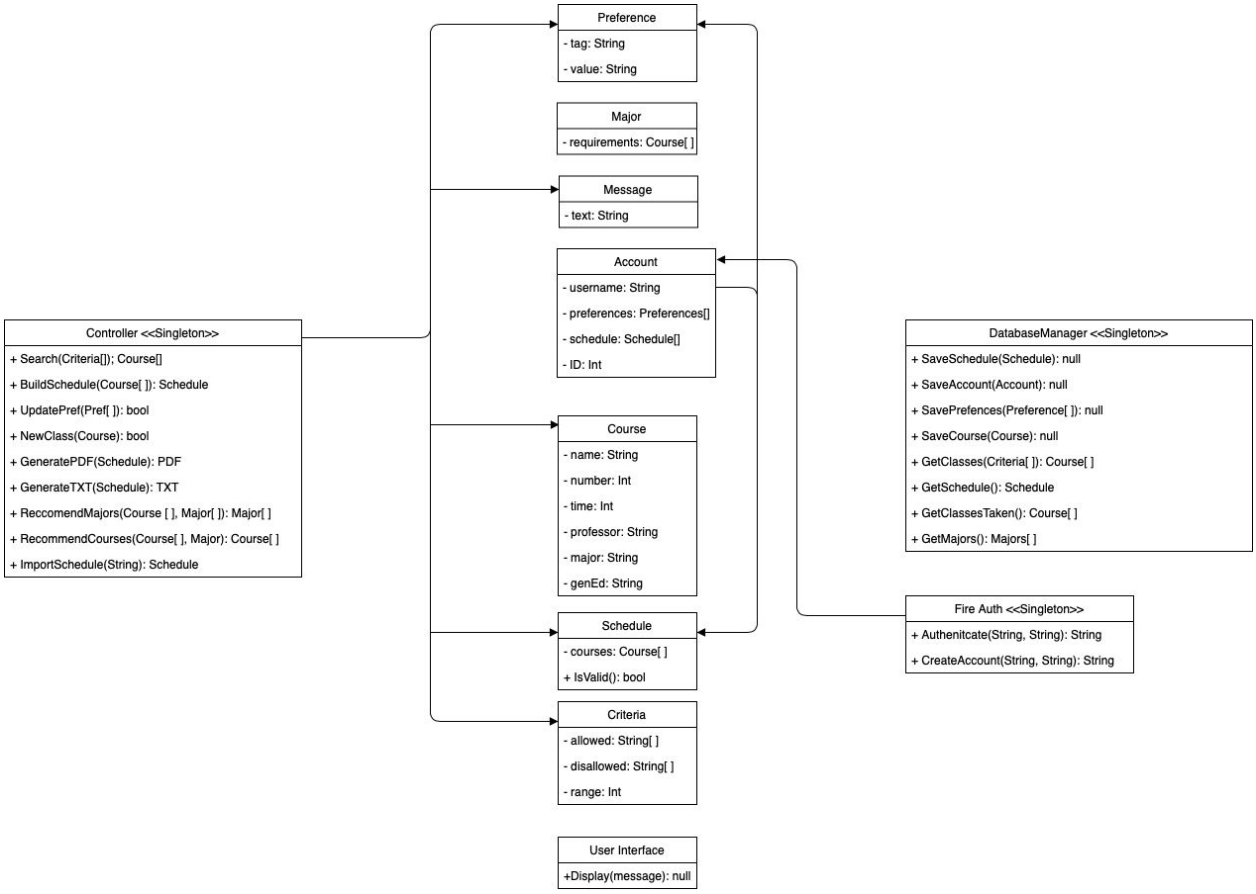
The observer pattern is a relationship between many objects (observers) with a single object (subject) in which the observers will be automatically notified if a change is made to the subject. This design pattern works well with the react framework we are using to implement our application due to the nature of react state. In react whenever the state is updated all components are notified and their render() functions are called. This allows us to have a centralized source of information with which all components can observe and update themselves whenever that information changes.

Design Class Diagram with Uses



The diagram above shows detailed class design and which classes they use. The use relationship is shown by a dotted line with the arrow pointing from the using class to the used class. The three classes with functions associated with them are Singletons. This is because our application doesn't need more than one instance of each, they have no attributes that would change with different instances. As shown in the diagram, the Database uses every object that it stores. This can be seen in the functions used by the Database. Where ever returned object has been found inside the Database. The Controller's most significant uses are the use of the Database and the FireAuth. Without the use of FireAuth none of the functions used by the Controller would be accessible and the same is true for the Database. Every object used by the Controller are passed in as parameters for certain functions.

Design Class Diagram with Creates



The Design Class Diagram with Creates shows all the classes and which objects they create. In the instance of Controller - which utilizes the Singleton pattern - creates the majority of the objects. This is because the Database will end up storing most generated objects; every function in the Database either stores or retrieves an object or a list of objects. Many of the Controller functions create the objects which will be passed into the database object. The functions *UpdatePref(Pref[]), NewClass(Course),* are examples of that. There are functions such as *Search(Criteria[])* which take in input and then pass that along to the Database, the output are the results generated from the Database retrieval. With the exception of FireAuth's *CreateAccount(String, String)* and that the creation of an Account has the creation of a Schedule and a list of Preferences, these are the two types of functions shown in this diagram.