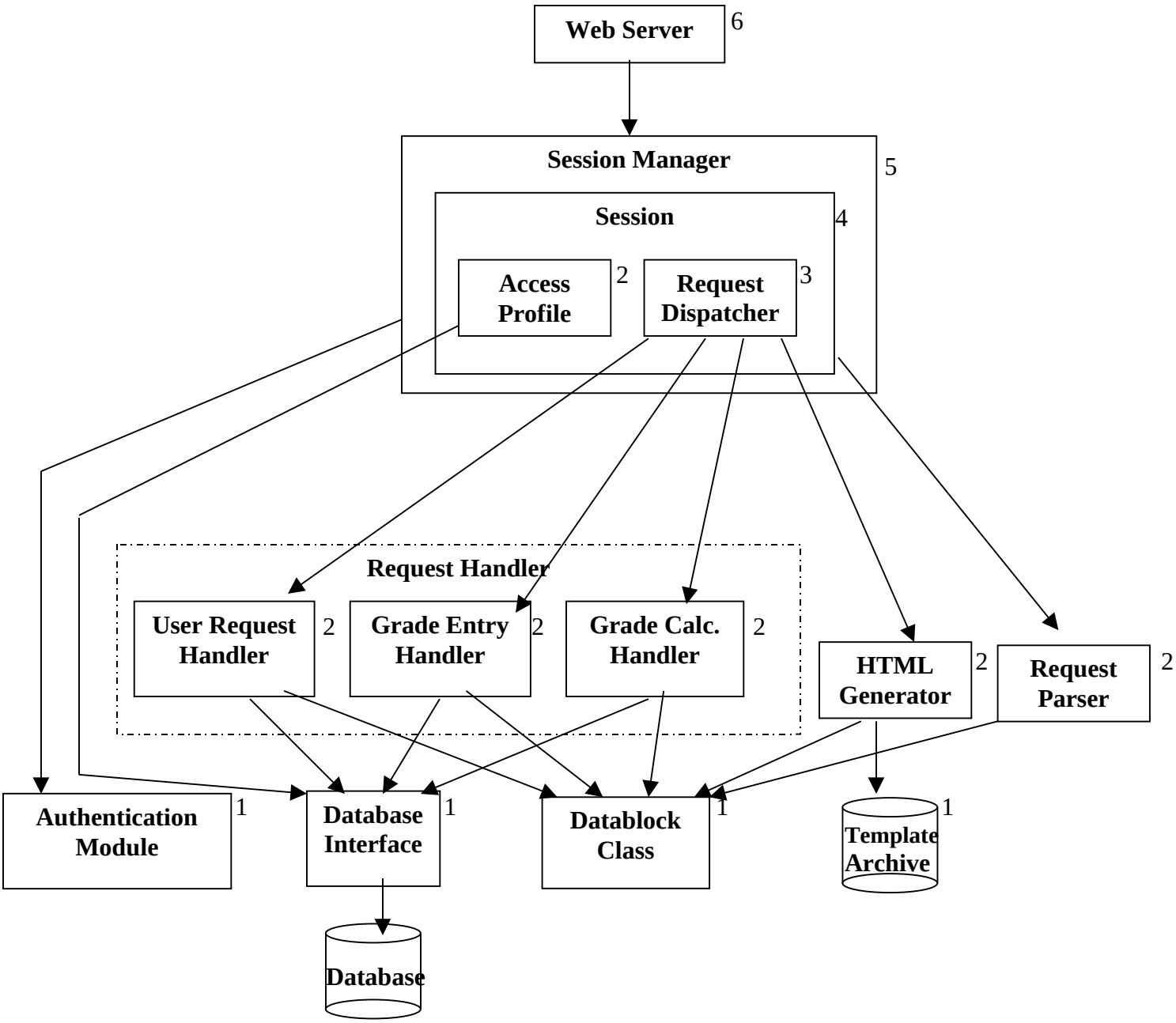
System Diagram

Each component is labeled with its level rank.



The Walkthrough

The web server gets a request over the https port, and hands the returned CGI string over to the Session Manager. The SM peeks at the message to see if there is a sessionID already established, or if this is a login request.

If it is a login request, the SM checks with the Authentication Module (Kerberos, or a password file), to make sure the username-password combo is correct. If not, it sends the user a nasty-gram. If so, it creates a Session object for that user. The session object automatically uses its Access Profile object to get the list of classes and class access from the database for the user. The Request Dispatcher object is also created and it sends the user the opening page. The session object is then put in the SM's session list.

If the incoming string is not a login request, but has a sessionID, the SM searches the session list for the sessionID. If that sessionID is not found, or if the session has expired, the user gets the requisite nasty-gram. If it is found, the string is passed to that Session's Request Dispatcher object. The RD first sends the request to the Request Parser which breaks up the string into a parameterPair list. This is a list where the CGI string is broken down into name-value string pairs. This list is returned to the Request Dispatcher with an OpCode. This integer OpCode, imbedded in the submit button CGI, tells the RD which function should be run with this data. The RD checks the Access Profile to make sure that the requested OpCode is legal for the specified class for this user. IF not, nasty-gram. If so, the RD, with the parameterPair list, creates a Datablock object and calls the proper function in one of the Request Handler modules with the parameterPair List and the Datablock object.

The Datablock object is a structure which will be used to package data to be sent to the HTML generator. It contains a sessionID, an OpCode, a pointer to each of these four standard types (course, assignment, person, assignmentType), and pointers to lists of each of these standard types (course, assignment, person, grade, assignmentType). The first pointers are those which each thing in the list pointers have in common. For instance, if we wanted a list of all the assignments in a particular course, the course pointer would have data in it, and the assignment list pointers would be a list of all the assignments for that class. Each of the objects in the list also contains a pointer to the entry in the AssignmentType list, which is the type of this assignment. If this is confusing, we'll talk about it in class on Monday.

The called module in the Request Handler take in the parameterPair list and does what it needs to do with it – calling the database Interface module when needed. It places into the received, empty, Datablock, all the information that needs to be passed to the HTML Generator. (not all the pointers will have information in them – only the ones that are relevant for this page) It then returns.

The Request Dispatcher then passes the block to the HTML generator which, seeing the OpCode in the Datablock, knows which page it needs to generate, and does so with the rest of the data in the Datablock. The Template Archive has all the pages written with the template language that both it and the HTML generator understand. The HTML generator returns the string representation of the page to the Request Dispatcher, and the string continues to spiral up through the Web server and eventually to the user.

Note: In the diagram, I have moved the Request Parser up one level in order to use the Datablock Class incase we decided that the returned item form the RP could be a Datablock instead instead of the parameterPairs. That will be decided by Monday.