3.1.2 Unit Descriptions

The following table lists each class in the subsystem and includes a short description of its purpose, as well as notes or reminders useful for the team:

|  |  |  |  |
| --- | --- | --- | --- |
| classes in the subsystem (and its attributes and functions) | short description of its purpose and useful reminders for the team | | |
|  | | |  |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (2).jpg | html/javascript | | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (1).jpg | html/javascript | | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (3).jpg | html/javascript | | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (4).jpg | html/javascript | | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (5).jpg | This is the central/main module of the project, because it handles every request from the client and redirects each request to the appropriate module that can handle the request. Furthermore, this is the module that responds directly the client’s | | |
| Reminder for the team: | | We should figure out whether this design might have any performance issues in the case where many users are connected to a single server.  We should make sure that Ajax is being used correctly from the client’s side to avoid sending unnecessary information to the server. | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (9).jpg | This class is responsible for abstracting the database and controlling access. The Query() function receives the SQL query from other classes and it sends it to the mysql database that’s being used for our project. This is good coding practice, since by not communicating directly with our database; we create an abstraction that is very efficient in the case where we might decide to change anything about our communication with the database, since all information is stored in one class. | | |
| Reminder for the team: | | This class has to be created as soon as possible since most of the other classes need this class to communicate. It is not difficult to write code without it, but it becomes impossible to test our implementation.  As long as our RDBMS is SQL-Compliant we don’t have to change anything in the database module after a first release of that class. | |

|  |  |  |
| --- | --- | --- |
|  | |  |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (7).jpg | | This module is responsible for authenticating, creating or modifying users (accounts).  This class starts a temporary session, during which, the browser that the client is using, will receive a unique ID which associates his session. Then, this module verifies client’s credentials and if they are correct, then the appropriate authentication level is set (0 = Administrator, 1 = Teacher, 2 = Student) and the user gets access to our web app. When a user logs out, the session ends and communication is terminated. |
| Reminder for the team: | We should try to create a simple SSL/TLS certificate, which will be used to avoid man-in-the-middle attacks where a hacker might be able to steal a client’s session by stealing his unique ID when a session is started.  Furthermore, we need to check every input from the client for potential security vulnerabilities, such as SQL injections, which enable a user to potentially manipulate the statements performed on the database. | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (8).jpg | | This module is responsible for obtaining or modifying student record information. This class calls frequently the database module to register for a course or get a student’s schedule, in which case, a schedule will be generated and converted to a Json format, which is the content format that our front-end calendar application understands and communicates. |
| Reminder for the team: | We need more functions for this class that will give more choice and flexibility to the end user. | |
| C:\Users\Andreas\Desktop\New folder (3)\class_diagram - Copy (6).jpg | | This module is responsible for obtaining course information, verifying any potential course dependencies, and generating possible schedules in function of given constraints. |
| Reminder for the team: | To implement +CheckDependecies(), we have to find an efficient algorithm that checks dependencies without wasting too much CPU time on the server. | |