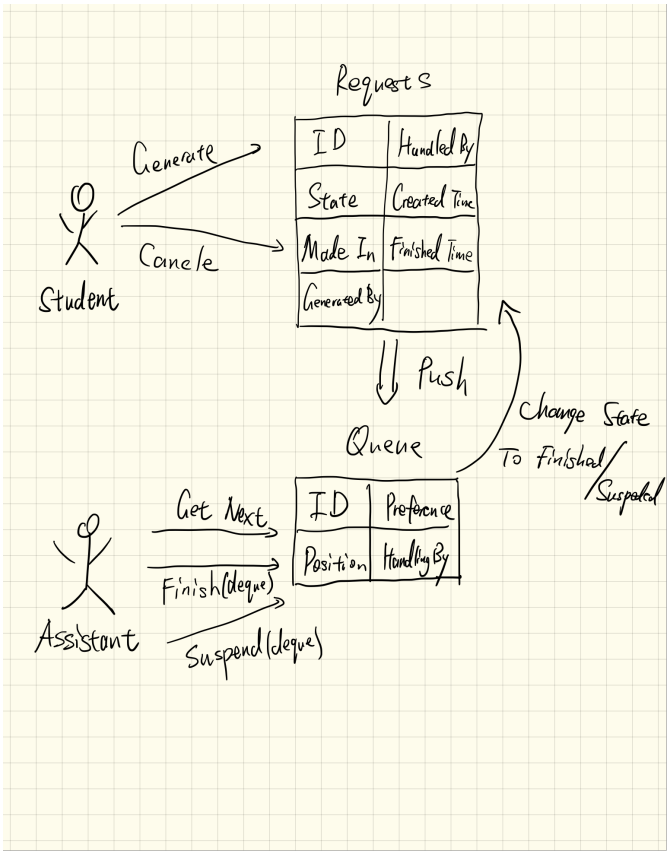
Document for Web-based Lab-Bot System

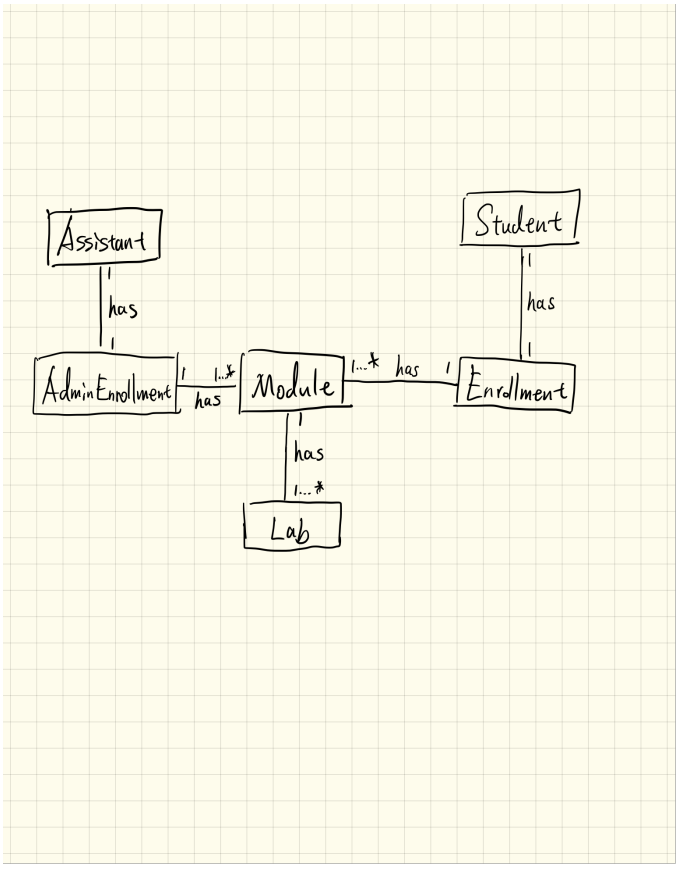
How to set up:

Because all the files are running locally, you should install and run the MySQL system in your computer before testing.

1. Import 'Lab-bot.qsl' file into your MySQL, which includes all the databases and some testing data.
2. Make sure credentials are correct in 'credentials.php', matching your own database.
3. 'studentLogin.php' is for student users to login.
4. 'adminLogin.php' is for assistant users to login.

Diagram:


Requests
ID
Handled By
State
Created Time
Made In
Finished Time
Generated By
Generate
Cancle
Student
Push
Queue
Change State
To Finished/
Suspeted
ID
Preference
Position
Handling By
Get Next
Finish(deque)
Suspend(deque)
Assistant


Assistant
has
AdminEnrollment
1
1...*
has
Module
1...*
has
1
Enrollment
Student
has
1
has
1...*
Lab

Database Explanations:

1) Users: Students' emails and passwords details.
2) Admins: Assistants' emails and passwords details.
3) Enrollment: Students module enrollment.
4) AdminEnrollment: Assistants module enrollment.
5) Modules: All module codes.
6) Labs: Module labs timetables.
7) Requests: All requests with ID. There are 4 states for a request naming 'Waiting', 'Finished', 'Canceled' and 'Suspended'.
8) Queue: The request queue.

Code Explanations:

Main Pages:

1) **"admin.php"** : The main page for the assistant. First part of PHP code is for session validation checking. Second part of PHP code is to get request history from the database to fill the table. Third part of PHP code is to get suspended requests which the assistant can handle. The JavaScript code at the bottom is to use several PHP queries to check next request state, queue size, current lab and next lab. The query interval is set as 5 seconds.
2) **"student.php"** : Similar to "admin.php", the main page for students. The middle big bulk of PHP code is to query available labs that the student can

request help for. It will check the ongoing lab, last lab within 15 minutes and the next lab within 10 minutes if there are any, and query the database to check whether the student can make request on these labs (10 and 15 can be easily changed in the code).

3) **"adminLogin.php"** : The page for assistance login, quite straightforward.
4) **"studentLogin.php"** : The page for student login.

Ajax Query PHP:

1) **"nextRequest.php"** : Check if the assistant is handling a request, if yes, echo the request student email back to main page, if not, query the next request in the queue. There is a **preference mechanism** here: when a student raises a request, if he has a request that was finished by an assistant within 10 minutes (can be modified in code) ago, the request he makes will have a preference which is the assistant that solved his request just now. Then this new request can only be handled by that specific assistant unless the request has been made for more than 10 minutes (can be modified in code), then any assistants for the lab can get this request.
2) **"requestSize.php"** : Check how many waiting requests.
3) **"currentLab.php"** : Check current lab.
4) **"nextLab.php"** : Check next lab today.
5) **"queryPosition.php"** : Check whether the student has raised a request, and query the state and the position of the request.

6) **“requestedLab.php”** : Get the lab that he student requests.

Action PHP:

1) **“finishiOneRequest.php”** : Finish the request being handled.
2) **“suspendOneRequest.php”** : Suspend the request being handled. (The assistant can suspend a request then handle another one. If the student’ s request gets suspended, the student cannot make another request until the request is solved by an assistant.)
3) **“finishiSuspendedRequest.php”** : Finish a suspended request with a specific ID (in GET).
4) **“getNextRequest.php”** : Get next available request that the assistant can handle.
5) **“makeARequest.php”** : Make a request and insert it to database.
6) **“onCancelling.php”** : Cancel the request.

Checking PHP:

1) **“onAdminLogin.php”** : Check if the assistant login valid.
2) **“onStudentLogin.php”** : Check if the student login valid.

Testing PHP:

1) **“loadTest.php”** : Test if the database can handle the Ajax query and request made by 100 students and 5 assistants (one request made and being handled per

minute).

2) **“requestInsertionAndFinishingTest.php”** : Simulate the process of making and finishing a request.

3) **“Testing.php”** : No actual use, only for demoing code.

RESTful API PHP:

1) **“Site.php”** : Main REST file, RESTful Webservice, indicating what kind of data is included in JSON.

2) **“RestController.php”** : To handle REST request.

3) **“SimpleRest.php”** : To get http statues message.

4) **“SiteRestHandler.php”** : Decode data into JSON.

Other PHPs:

1) **“credentials.php”** : The info of database.

2) **“functionSet.php”** : All the helper functions.

Q&A:

1) Q: How to change the time periods defining the last and next labs can be requested?

A: There are several PHP files should be modified. In “student.php”, at line 173, where defines the next lab time is within 10 minutes. Line 242 defines the last lab time is within 15 minutes (10 - 25 == -15). Same modifications should also

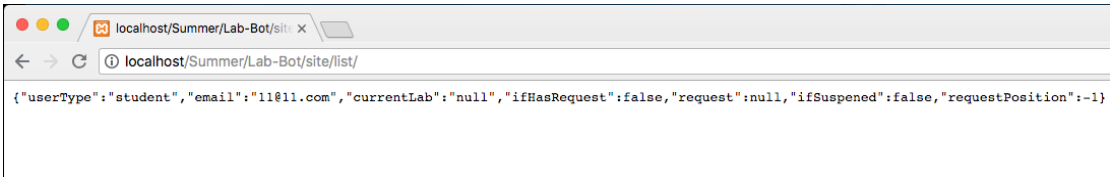
made in “functionSet.php”, at line 161 and line 200.

2) Q: How to change the time of preference mechanism?

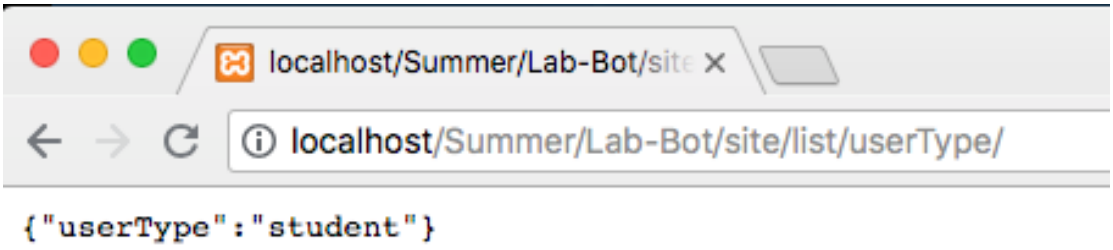
A: There are several lines of codes need to be changed. In “getNextRequest.php”, at line 39; In “makeARequest.php”, at line 120; In “nextRequest.php”, at line 74;

RESTful API Explain:

To use RESTful API from web browser, you should login first. Go to url: http://localhost/.../Lab-Bot/site/list/ (... is the prefix address), then you can see JSON data similar to below:



You can also get a specific JSON item by indicating in the URL, for example, if you go to URL: http://localhost/.../Lab-Bot/site/list/userType/



Please note the file “.htaccess” is essential to REST, keep it with other PHP files in the same folder.