Semester-Long Internship Report

On

Spoken Tutorial's Moodle Migration

Submitted by

Nihal Shetty,
NMIMS Mukesh Patel School of Technology Management & Engineering, Mumbai.

Under the guidance of

Prof. Kannan M. Moudgalya

Chemical Engineering Department

IIT Bombay

July 8, 2022.

Acknowledgment

First, I want to thank Mrs. Nancy Varkey, Senior Project Manager, Spoken Tutorial, IIT-Bombay, for giving me the opportunity to work with the Spoken Tutorial team for the internship.

I would also like to thank Mrs. Kirti Ambre, Assistant Project Manager, and Mr. Saurabh Adhikary, Assistant Project Manager, Spoken Tutorial, IIT-Bombay, for their continuous mentorship during my time at the organization. With their patience and openness, I benefited from an enjoyable working environment.

It is indeed with a profound sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I am grateful to FOSSEE for giving me the opportunity under the Semester Long Internship Program 2022, in the organization.

Last but not the least, I wholeheartedly thank all of FOSSEE & Spoken Tutorial's employees working in different projects for helping me evolve better with their critical advice.

With Regards.

Nihal Shetty.

Contents

1. Introduction

The Spoken Tutorial project is the initiative of the 'Talk to a Teacher' activity of the National Mission on Education through Information and Communication Technology (ICT), launched by the Ministry of Human Resources and Development, Government of India.

The use of spoken tutorials to popularize software development and its use will be coordinated through this website. (The Spoken Tutorial project is being developed by IIT Bombay for MHRD, Government of India).

The spoken Tutorial Project aims to make spoken tutorials on FOSS available in several Indian languages, for the learner to be able to learn in the language he/she is comfortable in. Our goal is to enable the use of spoken tutorials to teach in any Indian language, and to be taught to learners of all levels of expertise- Beginner, Intermediate or Advanced.

The Spoken Tutorial project provides various tutorials and courses in a wide range of IT topics. Audio and captioning are provided in up to 28 different languages, ranging from English and Hindi to regional languages like Assamese, Oriya, Telugu, Punjabi, Marathi, etc. and also some international languages such as Vietnamese. These courses are available for access at your own convenience for free on their website, https://spoken-tutorial.org/.

During the course of my internship, I was working on a software called Moodle, performing Data Migration of an old Moodle site used for conducting online tests of the courses offered by Spoken Tutorial project. This site had to be upgraded to the latest version of Moodle. Since the old Moodle site was customised to accommodate some information to be passed on from the parent website, the migration of data to the latest version became a difficult task as the default upgrade procedure does not allow for modified code/database contents.

Their courses also have online tests and certificates are awarded to successful candidates. To conduct the online tests, they had implemented Moodle LMS (Learning Management System). It has also been integrated into their portal. The Moodle site is operating on version 2.4.11, which is more than 7-8 years old, and many new features have been introduced since, making it impractical to operate and obsolete. Generally, Moodle software upgradations are a tedious, inconvenient and a lengthy process and the obsolete v2.4 makes the process even more difficult. Therefore, my objective during this internship is to perform upgradation of their Moodle LMS to the latest stable version, working on successful data migration of the database which has been altered during the integration with their portal, and may involve working on bringing new features to their webpage via Moodle Plugins or any other methods deemed necessary/optimal.

2. Moodle

Moodle is an acronym for 'Modular Object-Oriented Dynamic Learning Environment'. It provides a central space on the web where students can access a set of tools, resources, and courses anytime anywhere. Moodle is the most popular and trusted learning management system that caters to all types of organisations, no matter how large. Moodle is a free and open-source Learning Management System (LMS) written in PHP and distributed under the GNU General Public License. Developed on pedagogical principles, Moodle is used for blended learning, distance education, flipped classroom and other e-learning projects in schools, universities, workplaces, and other sectors. With customizable management features, Moodle is used to create private websites with online courses for educators and trainers to achieve learning goals. Moodle allows for extending and tailoring learning environments using community-sourced plugins.

While Moodle presents certain common features in almost all similar e-learning tools, it also provides certain plug-in options. As an e-learning platform, Moodle features: blogs, chats, database activities, glossaries, support systems enabling the functioning in multiple languages, content management, regular examination & assessment, etc.

The current infrastructure facilities adopted by Moodle enable it to support a plethora of plug-in options like graphical themes and content filters, enrolment, and authentication processes as well as resource and question patterns. Any operating system that supports the usage of PHP allows the usage of an e-learning platform like Moodle and some of the systems where Moodle can perform without any alterations include Mac OS X, Windows, Linux, Unix, NetWare etc.

3. Problem Statements

3.1 Problem 1

The Moodle site currently in use had its database and code altered a few years back, to integrate the Spoken Tutorial website to the Moodle site as a result of which, upgrading was no longer possible via the conventional method as mentioned on their documentation: (https://docs.moodle.org/311/en/Upgrading). Therefore, to perform migration, all of the changes need to be identified and analysed as to what purpose they serve.

3.1.1 Problem 1.1

All of the changes performed to the database involve addition/removal of tables, addition/removal of columns within tables, or datatype changes of the columns. These alterations are not concerned with the data, i.e., the individual records stored in the tables. So only the structural changes of the database are needed to be analysed.

Proposed Solution 1.1

A fresh install of Moodle 2.4.11 will provide access to the same database but without alterations. This can then be compared with the database in question to find out which tables and columns were altered. A backup of Spoken Tutorial's database (OTC.schema) was performed and then imported to my system, where I can compare it to the unaltered database (root).

3.1.2 Problem 1.2

There are up to 550+ tables which are to be compared against each other, to find out whether one exists in both databases, and whether they were dropped/added. This is a very tedious and time-consuming task if performed manually.

Proposed Solution 1.2

A python script can be written which finds out the names of all tables in each database and then iteratively compares them to find the status of the database structures. The information_schema database in MySQL is a default table which holds the metadata of all other databases in the system. It provides access to database metadata, information about the MySQL server such as the name of a database or table, the data type of a column, or access privileges. Other terms that are sometimes used for this information are data dictionary and system catalogue. This database can be accessed using the script to identify the changes.

3.1.3 Problem 1.3

The output of the Python script was inaccurate and not corresponding to the changes identified manually. There was some inconsistency in the outputs obtained.

Proposed Solution 1.3

The elements used to compare the tables were table_schema, table_name, column_name, ordinal_position, data_type, and column_type which determined whether there are any changes or not. These elements are a part of the information_schema database. The ordinal_position variable caused the discrepancy here, since it stood for the position of the column (e.g., column no. 12). While comparing tables this attribute is not needed and removal of this element would provide the correct output.

3.1.4 Problem 1.4

Once the list of changes has been obtained, the migration plan has to be determined by identifying what to do further. These changes have to be either completely expunged, or partially based on how much they affect the system. This has to be evaluated to proceed further.

Proposed Solution 1.4

So, it was found that apart from the 28 extra tables and 4 variable type changes in OTC.schema, only 6 column additions were present. Since all of the variable type changes were only size increases, and the extra tables are just won't affect the Moodle upgrade system. Essentially it all comes down to the 6 extra columns. So, the proposed solution to solve the issue was to drop the column additions and then test the migration on a sample site.

3.2 Problem 2

Along with the Moodle Database, the Moodle code, which is composed of PHP files, must have been altered too, in order to accommodate the changes and to incorporate them into the Moodle System. There is a need to identify these code changes to better understand how the Moodle site used by Spoken Tutorial works, and how migration can be performed.

3.2.1 Problem 2.1

Every time a new user, course, assignment, quiz or question bank is created, the Moodle code populates some tables to store these as data. These tables must be of utmost importance to the site, and while migration we can choose to ignore the rest if necessary. Therefore, all code files which perform tasks related to the above-mentioned examples must be carefully studied to find which tables are being used.

Proposed Solution 2.1

The Moodle System makes use of the XMLDB structure, which is Moodle's database abstraction layer - it is the library of code that lets Moodle interact with and access the database. Moodle supports a number of Database Engines, including MySQL, MariaDB, Postgresql, OCI, and MS SQL Server. Each of these has a slightly different format for some of their table creation statements. XMLDB has been created as a standardised format to describe the structure of the database in a human-readable format which the Moodle installer can turn into DDL commands to create the database structure. There are two main sublayers- DDL and DML (Data Definition and Data Manipulation), which have functions defined in them to perform these database operations. These files are stored in /opt/lampp/htdocs/moodle/lib/dml & /ddl. A new script can be implemented using Python that searches for these function calls within the directories of all operations in question (user, course, assignment, quiz).

3.2.2 Problem 2.2

The script written to perform the task mentioned in Problem 3.2 was not able to find out the table names and returned empty lists each time it was executed, especially in the case of DDL functions.

Proposed Solution 2.2

Every table defined in Moodle has a prefix “mdl_” attached to it. In the code, identifying the tables can be an increasingly difficult task, as these tables are not directly mentioned. Therefore, the script can be improvised by adding search terms such as “DML/DDL_function_name(‘table_name’….” so that it returns the exact name of the table, even without the prefix. In the case of DDL tables, another search term was used: “$table = new XMLDB table (‘table_name’….” which returned the correct results.

4. Design Considerations

4.1 Dependencies and Packages Used:

- **XAMPP:** XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

- **Mysqldump** : The mysqldump client utility performs logical backups, producing a set of SQL statements that can be executed to reproduce the original database object definitions and table data. It dumps one or more MySQL databases for backup or transfer to another SQL server.

- **Python Lib/os.py**: Python OS module provides the facility to establish the interaction between the user and the operating system. The Python OS module lets us work with the files and directories.

- **MySQL Cursor**: In MySQL by using a cursor, you can iterate, or step through the results of a query and perform certain operations on each row. The cursor allows you to iterate through the result set and then perform the additional processing only on the rows that require it. A cursor contains the data in a loop.

- **phpMyAdmin**: phpMyAdmin is a free and open-source administration tool for MySQL and MariaDB. As a portable web application written primarily in PHP, it has become one of the most popular MySQL administration tools, especially for web hosting services.

4.2 Technologies Used

- **Python**: High-Level Scripting Language for Back-End coding and for server-side programming. Python features a dynamic types system and automatic memory management and supports multiple programming paradigms.

- **MySQL**: It is the world's most used open-source relational database management that runs as a server providing multi-user access to a number of databases.

- **PHP:** PHP is a general-purpose scripting language geared toward web development. The PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive initialism PHP: Hypertext Preprocessor.. PHP code can also be directly executed from the command line.

5. Tasks in detail

5.1 Output of scripts implemented and utilized:

5.1.1 Script used to identify changes made in OTC.schema (new) database by comparing it with an unaltered (old) Moodle database:

Both databases were obtained from the same Moodle version, i.e., 2.4:
Total number of tables in old: 309
Total number of tables in new: 333
Total number of tables in both old & new (including duplicates): 642
Total number of tables in both old & new (excluding duplicates): 337
Total number of tables in old not present in new: 4
Total number of tables in new not present in old: 28

Tables found with changes:

Name	No. of changes	Description:
mdl_course_completion_aggr_method	1	• Column ‘criteriatype’ was changed from bigint(10) to bigint(20)
mdl_course_completion_criteria	1	• Column ‘criteriatype’ was changed from bigint(10) to bigint(20)
mdl_event	1	• Column 'name’ was changed from longtext to varchar(255)
mdl_external_tokens	1	• Column 'creatorid' was changed from bigint(10) to bigint(20)
mdl_user	7	• Column ‘institution’ was changed from varchar(40) to varchar(100). Column number 22->24 • Column 25 'academic_code' was added • Column 14 ‘agerange’ was added • Column 55 'flag' was added • Column 13 'gender' was added • Column 54 'invigilator' was added • Column 53 'organizer' was added

5.1.2 Script used to find total list of exclusive tables populated in different functions of Moodle found via their PHP files:

This is a summary of all mentions of tables in the various code files checked. The entire list consists of these 22 tables found throughout all the .php code files.

1. mdl_user
2. mdl_user_info_category
3. mdl_User_info_data
4. mdl_user_info_field
5. mdl_course_modules_avail_fields
6. mdl_course_modules
7. mdl_external_tokens
8. mdl_quiz_attempts
9. mdl_quiz
10. mdl_quiz_reports
11. mdl_quiz_feedback
12. mdl_quiz_question_instances
13. mdl_quiz_overrides
14. mdl_quiz_grades
15. mdl_quiz_question_statistics
16. mdl_quiz_question_response_stats
17. mdl_quiz_statistics
18. mdl_quiz_overview_regrades
19. mdl_question_attempt_step_data
20. mdl_question_attempt_steps
21. mdl_question_attempts
22. mdl_question_usages

So, from 5.1.1, it was found that apart from the 28 extra tables and 4 variable type changes in OTC.schema, only 6 column additions were present. Since all of the variable type changes were only size increases, and the extra tables won't affect the Moodle upgrade system. Essentially it all comes down to the 6 extra columns. So, my proposed solution to solve the issue was to drop the column additions and then test the migration on a test site.

5.2 Upgrading Moodle:

5.2.1 Backup:

Firstly, the new table additions were dropped from mdl_user, which makes our database almost similar to the unaltered database and makes it ready for migration. A backup of the system was performed using the following:

There are three areas that should be backed up before any upgrade:

1. Moodle software (For example, everything in lampp/htdocs/moodle): Backing this up is easy, the entire file must be copied and stored somewhere else in a safe location, preferably a different storage drive, along with lampp/htdocs/moodle/ config.php which has special permissions and cannot be copied easily. To bypass this, we can use Linux's file explorer nautilus, with root privileges by entering sudo nautilus on the terminal, after which we can transfer the config.php file.

2. Moodle uploaded files (For example, lampp/moodledata): Similar to Moodle Software, this is easy to backup as well, and there are no files with special permissions in moodledata.
3. Moodle database (For example, your Postgres or MySQL database dump): The following command was entered:
```
mysqldump -u root --skip-password --default-character-set=utf8
-N --routines --single-transaction --column-statistics=0
--skip-triggers --databases OTC.schema > OTCbackup.sql
```

5.2.2 Configuring Git and Upgrading:

So, the upgrade path I have decided on is:

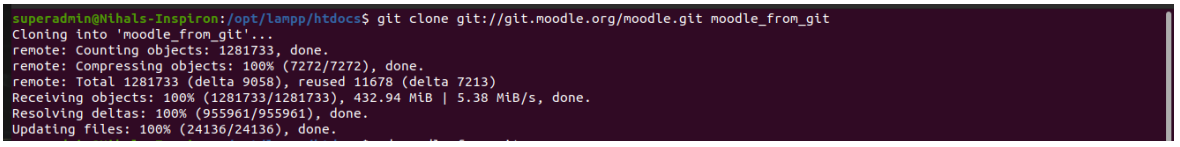
Moodle	2.4	2.6	2.7	3.1	3.5	3.9	4.0
Xampp PHP	5.6	5.6	5.6	7.0	7.0	7.2	7.4
MySQL	5.5.5	Switch to MariaDB	-	-	-	-	-
MariaDB	-	10.1.38	10.1.38	10.1.38	10.1.38	10.2.29	10.2.29

The PHP, MySQL and MariaDB versions are what I have used based on the minimum and maximum software requirements for each version of Moodle.

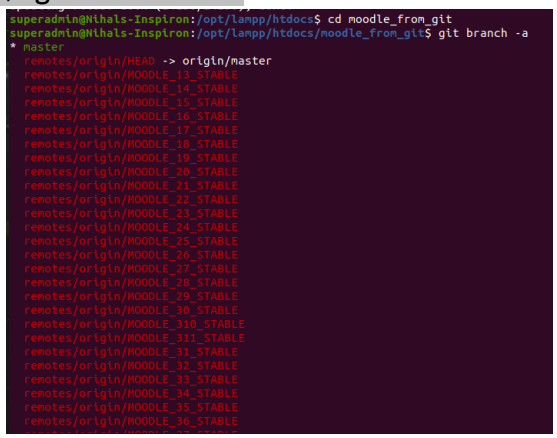
Moodle 2.6 is used, even though it is not an LTS version and can be skipped because: A disadvantage of using Xampp is that the highest compatible version of MySQL with Xampp is 5.5.5.X which is incompatible with Moodle from v2.7 onwards. MariaDB was introduced in Moodle 2.6, so it is used as a transitionary stage to switch the database from MySQL to MariaDB before getting to Moodle 2.7.

Once the data has been backed up, we can proceed with the upgrade.
The most convenient way to do so in my opinion, is through Git. Git is a software which is used for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. To do so we need to first install Git through the following command: `# sudo apt-get install git`.
After this, a clone of the Git repository of Moodle has to be set up (the original repository can be viewed at https://github.com/moodle/moodle). Follow these steps:

1. `$ cd /opt/lampp/htdocs/`
2. `$ git clone git://git.moodle.org/moodle.git moodle_from_git`
   This command initializes the new local repository as a clone of the 'upstream' (i.e., the remote server based) moodle.git repository to a new folder called moodle_from_git. The upstream repository is called 'origin' by default. It creates a new directory named *Moodle*, where it downloads all the files. This operation can take a while as it is actually getting the entire history of all Moodle versions.

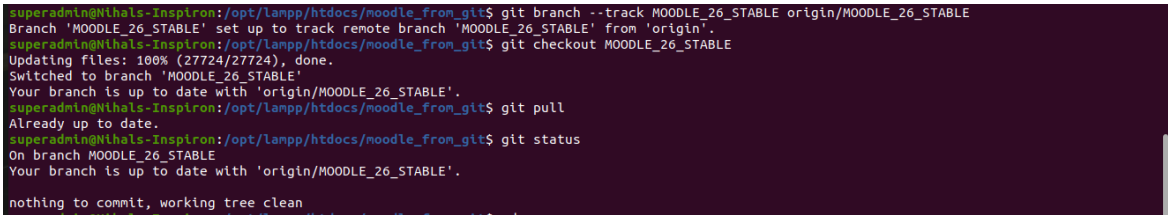


3. `$ cd moodle_from_git`
4. `$ git branch -a`. This command lists all available branches.



5. `$ git branch --track MOODLE_31_STABLE origin/MOODLE_31_STABLE`
   Use this command to create a new local branch called MOODLE_31_STABLE and set it to track the remote branch MOODLE_31_STABLE from the upstream repository. To install any other version, simply enter the name of the file as shown when executing `git branch -a`.

6. `$ git checkout MOODLE_31_STABLE`
   This command actually switches to the newly created local branch.



Now, we have to rename moodle_from_git to moodle and the original moodle as moodleog. Copy the config.php file from moodleog to moodle and then open http://127.0.0.1/Moodle. The site will then guide you through the update as shown below. I have chosen to upgrade to version 2.6, which is the first upgrade from the entire upgrade path to Moodle 4.0.
Now, whenever we need to upgrade to another version, simply enter the commands in steps 5 and 6 and replace the version name and that's it! This is why Git is more convenient, it takes only two commands to upgrade Moodle. Plus, it's even more convenient for registered site owners since it has more features which make the upgrading much less of a hassle.

The above-mentioned procedure is used to install Moodle 2.6 from 2.4. Now, before upgrading to Moodle 2.7, Database settings has to be changed from MySQL to MariaDB. This is done by editing the config.php file stored in /opt/lampp/moodle/. Change the `$CFG->dbtype = "mysqli"` to `$CFG->dbtype = "mariadb"` and save it. Upgrade successful, we can now see that there have been some noticeable changes in the UI.

Again, same procedure is then used to install 2.7 and 3.1. To proceed towards Moodle 3.5, Xampp has to be upgraded to PHP version 7.0 as a minimum. To do so:

1) Stop Xampp,
2) Rename the /opt/lampp folder to /opt/lampp-old (or whatever),
3) Then install the new lampp version.
4) Finally rename the newly installed htdocs folder to htdocs-original or so, and then copy the whole /opt/lampp-old/htdocs folder to the new release (/opt/lampp/htdocs,) and same for the MySQL data, copy the whole /opt/lampp-old/var/mysql to /opt/lampp/var/mysql (and also rename mysql folder from newly installed Xampp before).
5) Be sure to keep the same file permissions and users of the copied folders. Then copy moodledata from lampp-old to /opt/lampp/.

Then give the necessary permissions to the MySQL folder by entering:

```
sudo chmod 777 /opt/lampp/var/
sudo chown -R mysql:mysql /opt/lampp/var/mysql/
```

Note that after doing so, the Git upgrade will stop working. A new clone of the repository has to be initialized (Start from step 1 of the Git upgrade method again). This is another disadvantage of using Xampp since Xampp does not provide update procedures, you have to uninstall the old release and install the new release. This has to be repeated each time Xampp is upgraded during the Moodle upgrade path. Some errors which can be encountered during the Xampp upgrade are discussed in the Error Diagnosing (5.2.3) section.

Now, upgrading to Moodle 3.9. Xampp will have to be upgraded to 7.2 using the steps mentioned earlier to do so. When completed: the site will have some more changes to the UI. Now that we have been able to get to 3.9, we can proceed to the final and latest stable release of Moodle, i.e., Moodle 4.0. Xampp will have to be upgraded to 7.4 using the steps mentioned earlier to do so. After upgradation, all of the original data will be intact, as evidenced in the Outputs section.

5.2.3 Error Diagnosing:

There were some errors which I repeatedly encountered while upgrading Moodle and Xampp. The following mentions how to diagnose them:

1. Got error 2002: Can't connect to local MySQL server through socket '/var/run/mysqld/mysqld.sock':
   This error was shown when I tried to back up my MySQL database. To fix this error, I tried to find where mysqld.sock exists using `sudo find / -type s`. This command shows all socket files present. It was found to be present at /opt/lampp/var/mysql/mysql.sock so, I created a symbolic link to that file using Ubuntu's file explorer at /var/run/mysqld/ and named it mysqld.sock which fixed the error.

2. Opcache.enable issue in Moodle Environment:
   When Xampp is freshly installed/upgraded, Moodle will show a missing opcache warning each time you try to upgrade Moodle software. While this can be ignored since it's just a warning, opcache improves performance and I still tried diagnosing it. It can be installed by entering:

```
sudo apt-get install apache2 libapache2-mod-php php php-cli php-opcache php-mysql php-zip php-gd php-mbstring php-curl php-xml -y
```
This will install opcache and then, we can search where it is located using `sudo find / -name 'opcache.so'`. Then paste the following in /opt/lampp/etc/php.ini:

```
zend_extension=/opt/lampp/lib/php/extensions/no-debug-non-zts-20190902/opcache.so

[opcache]
opcache.enable = 1
opcache.memory_consumption = 128
opcache.max_accelerated_files = 10000
opcache.revalidate_freq = 60

; Required for Moodle
opcache.use_cwd = 1
opcache.validate_timestamps = 1
opcache.save_comments = 1
opcache.enable_file_override = 0
```

Now restart Xampp using `sudo /opt/lampp/lampp restart` and check Moodle environment, it will be fixed. Zend OPCache is now present in http://127.0.0.1/info.php when checked

3. These errors are encountered each time Xampp is upgraded:
mysqli_real_connect(): (HY000/2002): No such file or directory phpMyAdmin tried to connect to the MySQL server, and the server rejected the connection. You should check the host, username and password in your configuration and make sure that they correspond to the information given by the administrator of the MySQL server. Enter the following:
```
sudo chmod -R 777 /opt/lampp
sudo chown -hR nobody /opt/lampp
sudo chmod -R 755 /opt/lampp
sudo service mysql stop
sudo /opt/lampp/lampp restart
```

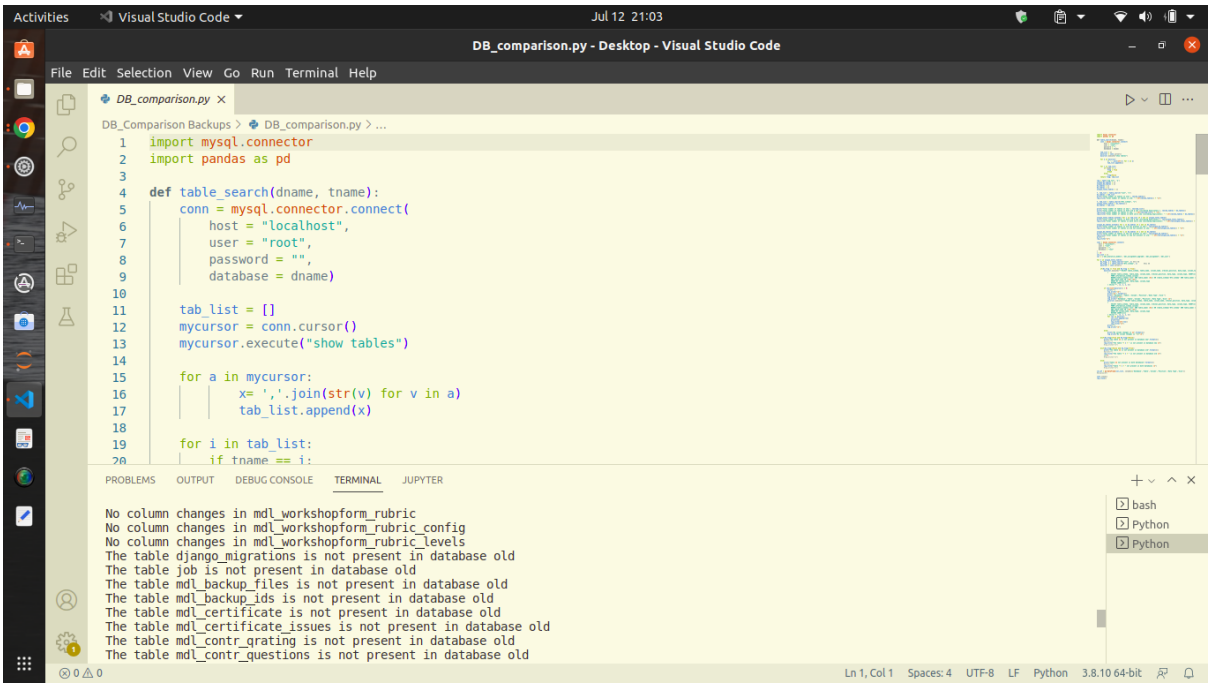
Can't create/write to file '/opt/lampp/var/mysql/ (Errcode: 13 "Permission denied")
```
sudo chown mysql:mysql -R /opt/lampp/var/mysql
```

XAMPP: Another web server daemon is already running and
```
sudo rm /opt/lampp/logs/httpd.pid
sudo netstat -nap | grep :80
```
note the PID (next to LISTEN),
```
sudo kill <PID>
```

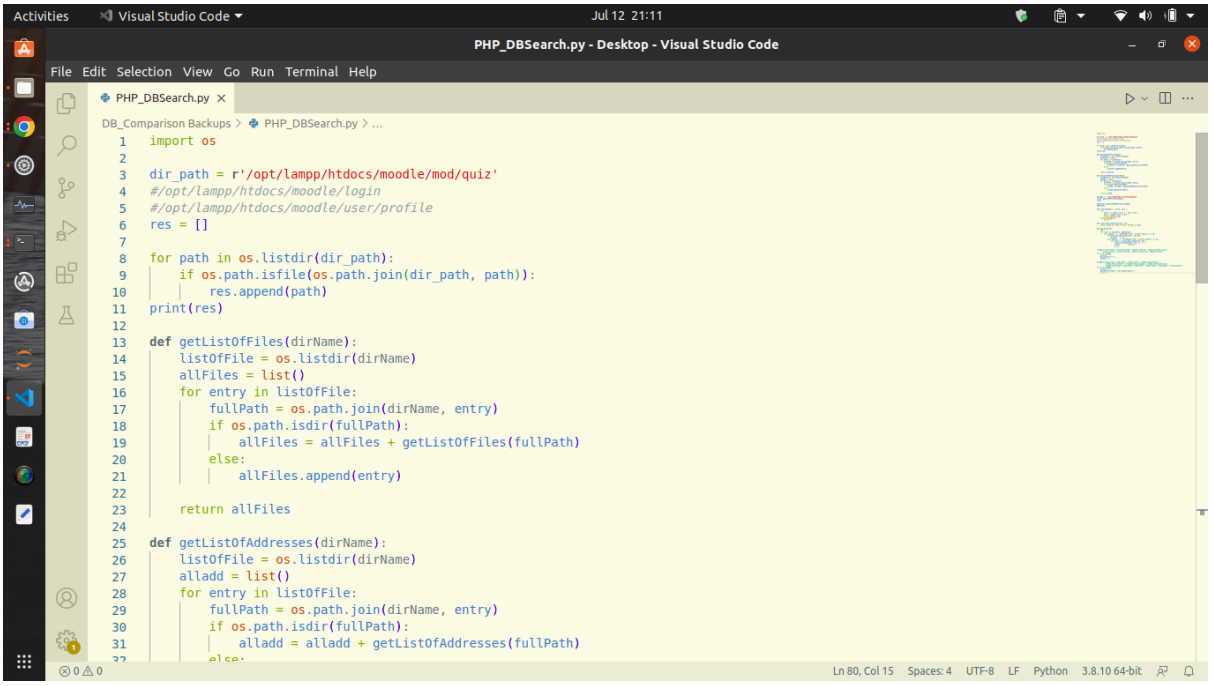
XAMPP: Another FTP daemon is already running.
```
netstat -peanutl | grep :21
Port <Port_number>
```

And then restart Xampp. The errors will have been fixed.

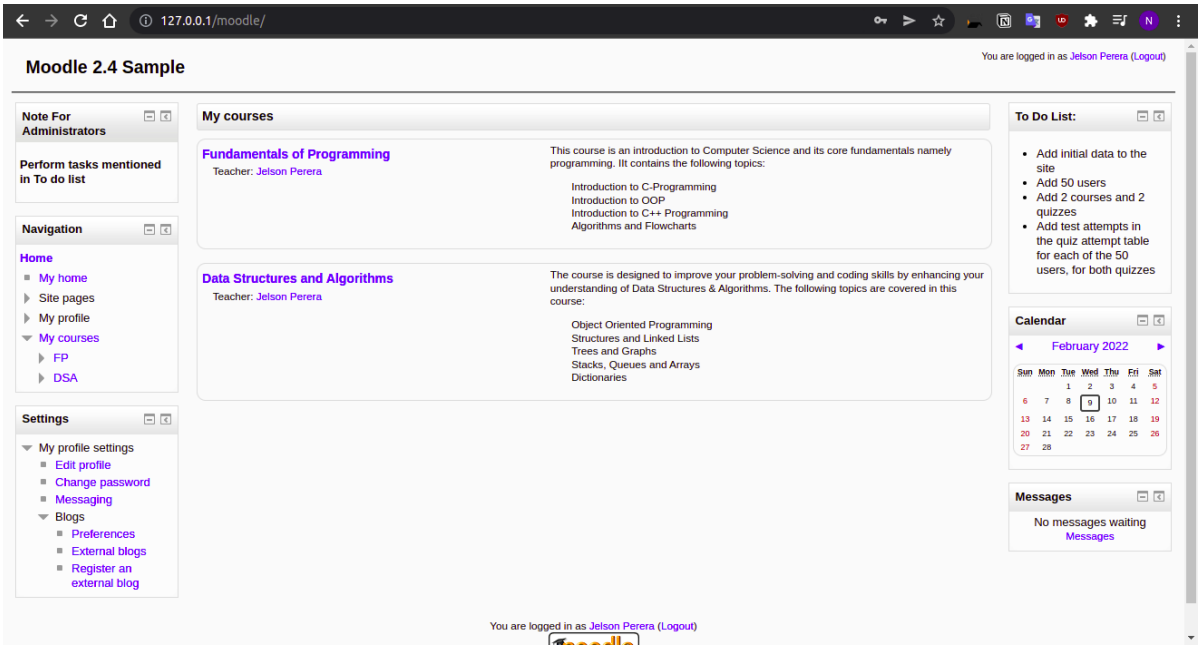
6. Outputs and Screenshots:



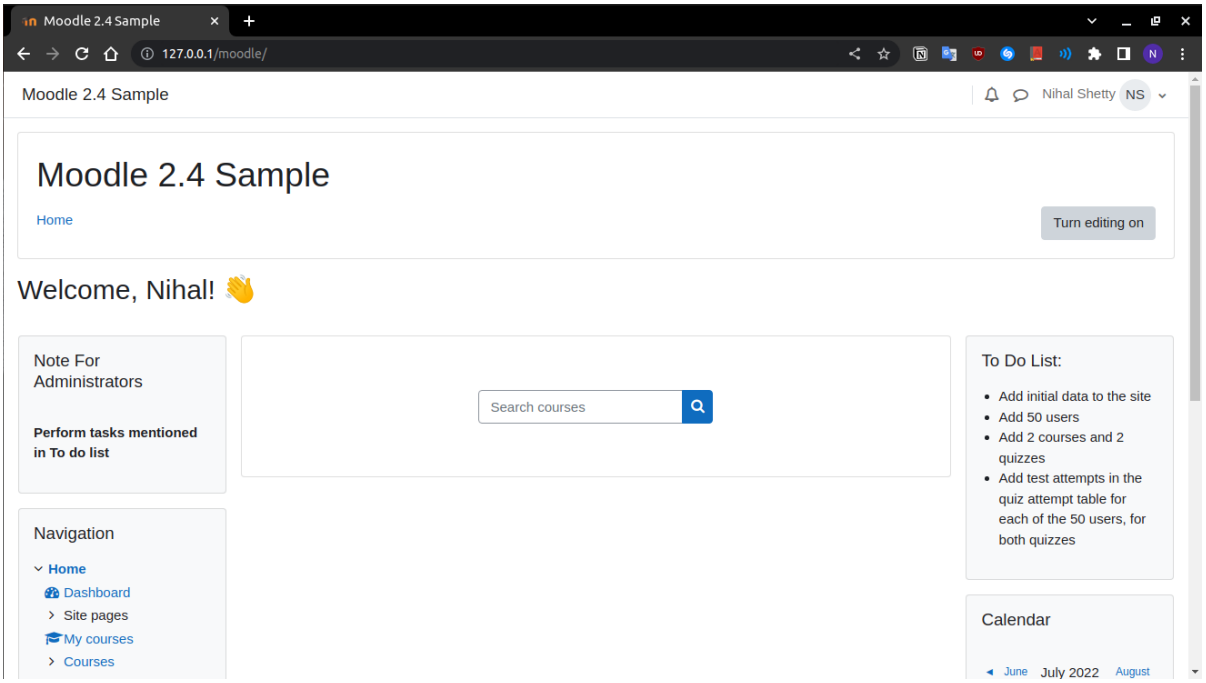
Snippet of the script used to identify database alterations:



Snippet of the script used to find out tables being populated in certain Moodle functions:



Moodle 2.4.11, the initial site from where the migration was performed



Moodle 4.0, the site after successful migration

7. References:

- https://moodle.org/
- https://moodle.org/mod/forum/
- https://www.youtube.com/channel/UCtubrWLY7-zm2RwenKh-bmQ/videos
- https://en.wikipedia.org/wiki/Moodle
- http://www.syndrega.ch/blog/
- https://docs.moodle.org/dev/Moodle_2.4_release_notes
- https://spoken-tutorial.org/stinternship2022/moodlemigration/
- https://moodledev.io/docs/apis/core/dml/ddl
- https://moodledev.io/docs/apis/core/dml/
- https://community.apachefriends.org/
- https://docs.moodle.org/dev/Overview_of_the_Moodle_question_engine#Database_tables
- https://docs.moodle.org/dev/Question_Engine_2
- https://docs.moodle.org/311/en/Verify_Database_Schema