**How to run the application?**

Method #1

Open the project in PyCharm, run main.py. You will then enter the login page through <http://127.0.0.1:5001>. This method is mainly used for application development as one computer play two roles – server and client.

Method #2

Add the computer’s local IP address in the last line of main.py, e.g.

app.run(debug=True, host='192.168.1.69', port=5001)

Other computers (i.e. clients) on the local network can then enter the login page by visiting the server’s IP address, e.g. <http://192.168.1.69:5001>

This method makes use of Flask’s built-in WSGI server. (The MC Test application was written in Python, based on the Flask web framework.) Because of the limited performance and security of Flask’s built-in server, this method is not recommended for production use.

Method #3

Set up a WSGI server (e.g. gunicorn) and a web server (e.g. nginx) on Mac to host the Flask application.

* gunicorn --config gunicorn\_config.py main:app
* nginx -c /usr/local/etc/nginx/nginx.conf

The example gunicorn and nginx configuration files can be found in the project folder.

With the WSGI and web servers, multiple clients can access the application by visiting the Mac’s IP address, e.g. http://192.168.1.134:5001

The port number can be changed from 5001 to another by modifying the nginx configuration file.

Initially, I tried to run the application with WSGI server only. Things looked fine until it was discovered that images could not be shown with the questions. After adding the web server, everything became correct.

Method #4

Deploy the application to a cloud platform which supports Flask applications. I have tried this method successfully on render.com with a free account. This method can support heavy loads as long as you are willing to pay for the server resources. Anyway, there is one concern – you may not want to upload the question banks to an external service provider.

**Project File Structure**

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |

1. Project’s root directory

Python files (\*.py) and the library/package requirements file (requirements.txt).

nginx’s configuration file is actually stored in /usr/local/etc/nginx/nginx.conf. A copy is kept here for easy reference.

1. Folder “instance”

The SQlite database file (data.db) stores candidates’ personal data (e.g. candidate no., name, phone no.) as well as their test data (e.g. drawn questions, saved answers, datetime of last update). Its contents are filled in and retrieved by Python codes. Manual modifications are not required.

The table structure is shown below.

A screenshot of a computer

Description automatically generated

1. Folder “static”

Prior to each test session, the test administrator should prepare candidates’ personal information in an Excel file called candidates.xlsx and upload to this folder through the test administrator’s web page. Once uploaded, the application’s Python code will randomly draw test questions for the candidates. The candidates’ personal data, together with the drawn question numbers and answers will then be saved in the database file.

Below is an example of candidates.xlsx. “trade” can represent the industry sector of a trade test. Or, it can be a course/module code for a school test. Instead of specifying when and where a test will be conducted, a batch no. is assigned by the test administrator to identify the test.

A screenshot of a computer

Description automatically generated

1. Folder “static/image”

Stores picture files of the question banks. They can be uploaded through the test administrator’s web page. At present, removal of files needs to be done by the server’s IT administrator.

1. Folder “static/questions”

Stores the questions bank Excel files. Like the picture files, these Excel files can be uploaded but not removed by the test administrator. Two examples of question bank are shown below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Folder “static/scores”

Stores the test results of each test session. The files can be downloaded but not removed by the test administrator.

1. Folder “static/test\_config”

Stores the test configuration files of different trades. The files can be uploaded through the test administrator’s web page. Two examples of test configuration file are given below.

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated | A screenshot of a computer  Description automatically generated |

In the left example (i.e. trade: VM), each test paper contains a total of 28 questions. Category A, B, C, D contribute 3 questions each while Category E, F, G, H contribute 4 questions each. In the VM question bank, the questions are numbered as 01A, 02A, …, 01H, 02H, …. In case the question bank contains one category of questions only, then the question numbers should be 01A, 02A, … 99A, 100A, …. In the test configuration, both “first category” and “last category” should be “A” and a single number (i.e. the total number of questions in test paper) should be entered in “question per category”.

In the right example (i.e. trade: EL), the situation is more complicated. For each category (i.e. A to H) of questions, they are further divided into 4 groups (M, N, O, P). When drawing questions from each category, either group M or group N, plus either group O or group P will be selected. In other words, questions from group M and group N will not appear together in each category of questions and the same principle applies to group O and group P. The question bank assigns numbers to the questions in the form of M01A, M02A, … M33H, N01A, N02A, … N21H, … etc. This peculiar arrangement was designed to support the Electrician Trade Test offered by the Hong Kong VTC.

**User Interface - Candidates**

1. Log-in

A screenshot of a computer

Description automatically generated

For test administrator, enter “admin” in “Candidate No.” and “39076670” as password. The password can be changed by modifying Python file main.py

1. During test

A screenshot of a computer

Description automatically generated

After candidate’s information is validated with the database records, he/she will enter the multiple choice test. During the test, candidates can move forward or backward among the questions.

To avoid any unforeseeable technically problems during the test, candidates are advised to press the “Save” button from time to time, to save their answers in the server. In case a candidate’s computer hangs during the test, he/she can reboot the computer and log in again. The saved answers will appear with questions so the candidate needs not to answer those questions again.

A screenshot of a computer

Description automatically generated

At present, the application does not time the test, the test administrator is required to announce when the test ends. When time is up, candidates should press the “Finish” button. Their answers will be saved and the final scores will be calculated.

A screenshot of a computer

Description automatically generated

If you do not want candidates know their scores at this stage, please modify main.py

*…*

*def finish():*

*…*

*message = f"測驗結束。 ~~你的總分是 {result}~~.<br>Test finished. ~~Your score is {result}.~~"*

Once the “Finish” button is pressed, candidates will not be able to save their answers. in case some candidates do not follow test administrator’s instruction to end the test, administrator can press the “Finish” button for them via the administrator web page.

A screenshot of a phone

Description automatically generated

**User Interface – Administrator**

To visit the test administration page, one should enter “admin” and “39076670” as the candidate no. and password on the log-in page.

A screenshot of a computer

Description automatically generated

Function 2

The test administrator should have uploaded the candidates’ information file (candidates.xlsx) onto the server. With the trade and batch no. specified in the input boxes, pressing this button instructs the application to create records (i.e. personal data + question numbers and answers) in the database, for the corresponding candidates. If the records already exist, action will be skipped.

Function 3

Show a summary of candidates’ data for a specified test session.

Function 4

End a test session. Candidates will not be able to change their answers.

Function 5

If a candidate presses the “Finish” button to end the test, his/her score will be calculated and saved in the database. If the test was terminated by the test administrator (i.e. by means of Function 4), Function 5 should be used to calculate and save the final scores.

Function 6

If a candidate presses the “Finish” button by mistake. Test administrator can use this function to cancel the “Finish” action and the candidate can continue the test by logging in again.

Function 1

This function allows the test administrator upload candidates’ information, question bank, question pictures, and test configuration to the server.

To avoid confusions, there are some restrictions on the filenames:

Candidates’ information – the filename must be “candidates.xlsx”

Question bank – the filename must be “questions\_??.xlsx” where ?? (1 to 5 characters long) is the trade code.

Question picture – file type must be bmp, jpg, png or gif

Test configuration – the filename must be “config\_??.xlsx” where ?? (1 to 5 characters long) is the trade code.

A screenshot of a car

Description automatically generated