# 5 TEST

## 5.1 Test Overview and Test Plan

### 5.1.1 Test plan

After the implementation of the whole functional system, we design a detailed test plan to validate and check its reliability and robustness. The platform mainly has five sub-systems in total, and we make great efforts to design a set of test cases to cover all kinds of different possibilities. The test plan mainly covers two parts, one is for modular testing, and the other part is for integrated testing. Black box and white box testing methodogies are both adopted, with the purpose to cover all statements in the code and remove all potential bugs the system may encounter. Besides, the assumption behind the test is that we treat the user as a beginner and know nothing about our platform. Therefore, some random input context may be expected during the user’s operation. The goal of the test is to simulate different situations that the user may have with some unexpected behaviors. In the best case, our system should pass all test cases designed and be able to handle any potential situations.

### 5.1.2 Test Tools

*pytest-flask* is a powerful tool designed for automatic testing. We utilize it to test the back-end for routing flow check. Since we implement the back-end of the system in a python *flask* frame, this tool is able to provide us a relatively easy way to check the correctness.

Apart from the automatic testing, we also manually check the front-end flow of the website by following all predefined test cases. Users’ operations are simulated to guarantee the connection between the front-end and back-end is free of problems and bugs. We try to design the minimum set of test cases to reduce such human-intensive works.

### 5.1.3 Test Environment

* Front-end working flow is tested in both browsers, namely Chrome and Safari
* Back-end routing flow and functions are partially tested under *pytest-flask*
* Database are tested in Mysql WorkBench and python *pymysql* package
* All testing cases are conducted in macOS Mojave 10.14.6
* The IDE used for coding and testing is PyCharm and VS code

### 5.1.4 Features to be tested

Overall, we have one integrated system and five subsystems to be tested, including Account System, Course System, Question System, Answer System and Coupon System. Testing details will be covered in Section 5.2. The corresponding databases to be tested are Account, Enrollment, Course, Question, Answer, and Coupon. In order to have a clearer overview of all tested features, we list all subsystems and their corresponding functions and related databases in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Subsystem** | **Feature** | **Function** | **Database** |
| Account | User Management | load\_user(user\_email) | N/A |
| login() / logout() | Account |
| signup() | Account |
| Email Verification | send\_async\_email(app, msg) | N/A |
| send\_email(to, subject, template, \*\*kwargs) | N/A |
| email\_check() | N/A |
| confirm(token) / unconfirmed | Account |
| Landing Page | teacher\_main() / student\_main() | Account |
| Course | Course Management | teacher\_create\_class(course\_code, course\_name, course\_instructor, course\_token) | Course / Enrollment |
| student\_get\_class(course\_token) | Course / Enrollment |
| Question Management | teacher\_view() | Question / Answer |
| student\_within\_course(course\_id) | Question / Answer |
| Question | teacher\_add\_question(course\_id) | Question |
| teacher\_view\_question(question\_id) | Question |
| Answer | Answer Management | teacher\_view\_answer(question\_id) | Question / Answer |
| teacher\_collect\_answer(question\_id) | Question / Answer |
| update\_answer(q\_id) | Answer |
| Coupon | Coupon Management | reward\_coupon(user\_id, q\_id, a\_id) | Coupon |
| use\_coupon(student\_id, course\_id) | Coupon |

## 5.2 Case - Account

### 5.2.1 Purpose

The purpose of testing on Account System is to validate whether it is a well-defined system for the user to login, logout, signup, email check and account activation. The system should be able to pass all test cases designed to handle all valid operations. Any invaild operations should be detected automatically and well-managed. Also, the account system should be able to protect user’s privacy and their personal information.

### 5.2.2 Inputs

|  |  |  |
| --- | --- | --- |
| **Case** | **Description** | **Input** |
| 1 | Directly access the main page without login by hacking the website address. | http://127.0.0.1:5000/student\_main\_page?messages=22 |
| 2 | Check signup page and login page are able to jump to each other. | Click the text “Already have an account? Login” or “No account yet? Sign up” |
| 3 | Correct signup for a student. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': 'xuxiangcuhk@qq.com',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '1234'  } |
| 4 | Correct signup for an instructor. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': 'xuxiangcuhk@qq.com',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '1234',  'is\_instructor': True  } |
| 5 | Check whether the system can detect invalid SID. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '115510',  'email': 'xuxiangcuhk@qq.com',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '1234'  } |
| 6 | Check whether the system can detect the invalid email address. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': 'xuxiangcuhk',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '1234'  } |
| 7 | Check whether the system can detect the invalid user name. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': 'xuxiangcuhk@qq.com',  'user\_name': 'XiaoYU^^',  'password': '1234',  'confirmed\_password': '1234'  } |
| 8 | Check whether the system can detect the unmatched password. | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': 'xuxiangcuhk@qq.com',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '12'  } |
| 9 | Check whether the system can detect the registered email (the email 1155107785@link.cuhk.edu.hk has already been in the database). | 'input\_data': {  'first\_name': 'Steven',  'last\_name': 'Yu',  'SID': '1155101234',  'email': '1155107785@link.cuhk.edu.hk',  'user\_name': 'XiaoYU',  'password': '1234',  'confirmed\_password': '1234'  } |
| 10 | Correct login for a student. | 'input\_data': {  'email': '1155107785@link.cuhk.edu.hk',  'password': 'wecoupon'  } |
| 11 | Check whether the system can detect the wrong email or password. | 'input\_data': {  'email': '1155107785@link.cuhk.edu.hk',  'password': 'we1234'  } |
| 12 | Test whether the system can successfully send out email. | Successful signup required |
| 13 | Test whether the account activation link works. | Click the activation link: <http://127.0.0.1:5000/confirm/76dacb79b8fb1c828e301a6cdb1c4356> |
| 14 | Check whether the user can log out smoothly. | Click the button “logout” |

### 5.2.3 Expected Outputs & Pass/Fail Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Output** | **Pass** | **Fail** |
| 1 | The action is not allowed. The system shall inform the user “login required”. | Return back to Login page | Successful login |
| 2 | Signup page and login page are able to jump to each other. | Click “Already have an account? Login” , go to Login page; click “No account yet? Sign up”, go to Signup page | After clicking the text, there is no page transition or wrong redirection |
| 3 | Correct signup. Go to the email checking procedure. | The page is redirected to email checking | Output error message or no further process |
| 4 | Correct signup. Go to the email checking procedure. | The page is redirected to email checking | Output error message or no further process |
| 5 | The system shall inform the user that “SID requires 10 numbers”. | Output the message to user “SID requires 10 numbers”, and ask user to input the field again | Successful signup |
| 6 | The system shall inform the user that “The email address is not valid”. | Output the message to user “The email address is not valid”, and ask user to input the field again | Successful signup |
| 7 | The system shall inform the user that “Usernames must have only letters, numbers, dots or underscores  ”. | Output the message to user “Usernames must have only letters, numbers, dots or underscores”, and ask user to input the field again | Successful signup |
| 8 | The system shall inform the user that “Passwords must match”. | Output the message to user “Passwords must match”, and ask user to input the field again | Successful signup |
| 9 | The system shall inform the user that “The email has already been registered”. | Output the message to user “The email has already been registered”, and ask user to input the field again | Successful signup |
| 10 | Correct login. Go to the main page of the corresponding student. | Go to the main page of the corresponding student. | Unsuccessful login |
| 11 | The system shall ask the user to input the password again. | Remain at Login page and ask the user to input the password again | Successful login |
| 12 | The system shall send out the verification email and redirect to the email check page. | Send out the verification email and redirect to the email check page | No email received or the page remains the same |
| 13 | The link shall direct the user to the main page without login any more. | Go to main page without login; the status of the account should be activated | No response after the click or the link directs to the login page |
| 14 | The system shall log out successfully. | Go to Login page. | Cannot log out. |

## 5.3 Case - Course

### 5.3.1 Purpose

The purpose of testing on Course System is to validate whether it is a well-defined system for the instructor to create a course and for the student to register a course via token. The display of all registered courses should also be free of errors. The system should be able to pass all test cases designed to handle all valid operations. Any invaild operations should be detected automatically and well-managed.

### 5.3.2 Inputs

|  |  |  |
| --- | --- | --- |
| **Case** | **Description** | **Input** |
| 1 | Correct course creation. | 'input\_data': {  'course\_code': 'ESTR1003',  'course\_title': 'Java',  'course\_instructor': 'Prof. Michael R. Lyu',  'course\_token': 'question'  } |
| 2 | Check whether the system can detect the token has already been occupied (the token ‘software’ has already been used by others). | 'input\_data': {  'course\_code': 'ESTR1003',  'course\_title': 'Java',  'course\_instructor': 'Prof. Michael R. Lyu',  'course\_token': 'software'  } |
| 3 | Correct course registration. | 'input\_data': {  'course\_token': 'ESTR1002'  } |
| 4 | Check whether the system can detect the wrong token. | 'input\_data': {  'course\_token': 'wrong token'  } |
| 5 | Check the display of the list of courses | Account login required |
| 6 | Check the link behind each course code can direct user to the corresponding course page | Click the course code “ESTR1002” |

### 5.3.3 Expected Outputs & Pass/Fail Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Output** | **Pass** | **Fail** |
| 1 | The system shall allow the correct creation of a new course. | Create a new course and display it in the main page | Cannot create the corresponding course |
| 2 | The duplicated token shall be rejected by the system. The system shall inform the user that “The token is occupied by others”. | Output the message to user “The token is occupied by others”, and ask user to input the field again | Create the course successfully |
| 3 | The system shall allow the correct registration of a new course. | Enroll in a new course and display it in the main page | Cannot enroll in the corresponding course |
| 4 | The wrong token shall be rejected by the system. The system shall inform the user that “The token is not available”. | Output the message to user “The token is not available”, and ask user to input the field again | Enroll in the course successfully |
| 5 | The system shall display all registered courses with correct course code, title and instructor name. | The display of the list of courses in the main page is completed and correct | There is some missing courses displayed or errors in the column |
| 6 | The system shall direct the user to the corresponding course page. | Correct page transition to the corresponding course | No page transition or wrong page transition |

## 5.4 Case - Question

### 5.4.1 Purpose

The purpose of testing on Question System is to validate whether it is a well-defined system for the instructor to add a new question, edit the question, start a (random) question, view a question, and for the student to answer the question. The display of all student participations and attempts in different questions should also be free of errors. The system should be able to pass all test cases designed to handle all valid operations. Any invaild operations should be detected automatically and well-managed.

### 5.4.2 Inputs

|  |  |  |
| --- | --- | --- |
| **Case** | **Description** | **Input** |
| 1 | Correct question creation. | 'input\_data': {  'question\_title': 'Demo Question',  'question\_content': 'What is the name of this course?',  'suggested\_answer': 'CSCI3100',  } |
| 2 | Create a question without some required data. Check whether the system can handle this case. | 'input\_data': {  'question\_title': 'Demo Question',  'question\_content': 'What is the name of this course?',  'suggested\_answer': '',  } |
| 3 | Check whether the question can be correctly edited (edit on the first test case). | 'input\_data': {  'question\_title': 'D Question',  'question\_content': 'What is the name of the course?',  'suggested\_answer': 'CSCI',  } |
| 4 | Check whether the system can start a question session for the instructor account. | Click “start” button in the question bank |
| 5 | Check whether the system can start a default question session for the instructor account. | Click “Start default question” button in the question bank |
| 6 | Check whether the question which ends will move to the posted question column. | End question operation required |
| 7 | Check all the data displayed in question view page | Click the “View” button in the posted question column (In this case, the first question “Simple Coupon” is selected.) |

### 5.4.3 Expected Outputs & Pass/Fail Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Output** | **Pass** | **Fail** |
| 1 | The system shall allow the correct creation of a new question. | Create a new question and display it in the question bank | Cannot create the corresponding question |
| 2 | The system allows the data missing in ‘suggested\_answer’. It can still create a new question as normal. | Create a new question and display it in the question bank | Cannot create the corresponding question |
| 3 | The system allows editing. The question in the question bank will be changed correspondingly. | The question in the question bank is successfully edited | Unsuccessful edit of a question |
| 4 | The system shall start the question session automatically. The user will in the answer collecting page. | The question starts and the user is redirected to answer collecting page | Unsuccessful start of a question |
| 5 | The system shall start the default question session automatically. The user will in the answer collecting page | The default question starts and the user is redirected to answer collecting page | Unsuccessful start of the default question |
| 6 | The system shall move the ended question to the posted question column for further viewing | The question stays in the posted question column | The question stays in the question bank column |
| 7 | The system shall display all required data as follows:  ‘output\_data’: {  ‘per\_ans’: {'answered': 3, 'not\_answered': 0},  ‘answer\_list’:  [{'answer\_id': 1, 'answer\_userid': 1, 'answer\_user': 'StevenXU', 'answer\_content': 'hard to implement', 'status': 1}, {'answer\_id': 2, 'answer\_userid': 2, 'answer\_user': 'Piper', 'answer\_content': 'explosive states', 'status': 1}, {'answer\_id': 6, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to perform', 'status': 1}, {'answer\_id': 7, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to do', 'status': 0}, {'answer\_id': 8, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to perform', 'status': 0}, {'answer\_id': 9, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to answer', 'status': 0}]  } | ‘per\_ans’ is correctly shown in the pie chart; ‘answer\_list’ is correctly listed in the answer table. The ‘status’ = 0 is labeled as ‘reward’ (which means to be rewarded) and ‘status’ = 1 is labeled as ‘rewarded’ | There is some errors in data display |

## 5.5 Case - Answer

### 5.5.1 Purpose

The purpose of testing on Answer System is to validate whether it is a well-defined system for the instructor to collect answers, and for the student to submit the answer. The two accounts should be able to interact with each other. Specifically speaking, once the student submits the answer, the instructor should be able to see the answer in his/her answer collecting page. Such interaction is the core part of *WeCoupon* platform. The system should be able to pass all test cases designed to handle all valid operations. Any invaild operations should be detected automatically and well-managed.

### 5.5.2 Inputs

|  |  |  |
| --- | --- | --- |
| **Case** | **Description** | **Input** |
| 1 | Submit a valid answer for the existing question. | 'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer',  'a\_time': 1617701891  } |
| 2 | Submit a invalid answer for the question (the question does not exist in the database). | 'input\_data': {  'question\_id': 100,  'answer': 'Demo Answer',  'a\_time': 1617701891  } |
| 3 | Two students submit answers concurrently. Check whether the system can handle such a situation. | 'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 1',  'a\_time': 1617701891  }  'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 2',  'a\_time': 1617701891  } |
| 4 | Check whether the student can submit the answer multiple times. | 'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 1',  'a\_time': 1617701891  }  'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 2',  'a\_time': 1617701895  } |
| 5 | Check whether the answers are always under update in the instructor’s answer collecting page. | 'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 1',  'a\_time': 1617701891  }  'input\_data': {  'question\_id': 1,  'answer': 'Demo Answer 2',  'a\_time': 1617701895  } |
| 6 | Check whether the website terminates updating after stop collection | Click the button “Stop Collection” |
| 7 | Check the order of all displayed answers. The rule should be that the less coupon the user has, the higher priority he/she gets. If the coupon number is the same, then time factor will be considered. The less time the user takes to submit the answer, the higher priority the user gets. | [{'answer\_id': 1, 'answer\_userid': 1, 'answer\_user': 'StevenXU', 'answer\_content': 'hard to implement', 'status': 1}, {'answer\_id': 2, 'answer\_userid': 2, 'answer\_user': 'Piper', 'answer\_content': 'explosive states', 'status': 1}, {'answer\_id': 6, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to perform', 'status': 1}, {'answer\_id': 7, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to do', 'status': 0}, {'answer\_id': 8, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to perform', 'status': 0}, {'answer\_id': 9, 'answer\_userid': 3, 'answer\_user': 'Prof\_Michael', 'answer\_content': 'hard to answer', 'status': 0}] |

### 5.5.3 Expected Outputs & Pass/Fail Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Output** | **Pass** | **Fail** |
| 1 | The system shall allow the correct submission of an answer. | The answer should be submitted successfully and be displayed in the instructor’s answer collecting page | The answer cannot be submitted or it cannot be displayed in the instructor’s answer collecting page |
| 2 | The system shall disallow the invalid submission of an answer. | The answer cannot be submitted and it will not be displayed in the instructor’s answer collecting page | The answer is submitted successfully and displayed in the instructor’s answer collecting page |
| 3 | The system shall record the 'a\_time' as the same number in the database. | The system can still process under this case and the records are correctly kept in the database | The system outputs error message and the records cannot go into the database |
| 4 | The system shall allow the user to submit the answer multiple times. There will not be any error message. | The system can still process under this case and the records are correctly kept in the database | The system outputs error message and the records cannot go into the database |
| 5 | The system shall refresh the answer collecting page every 1 second | The answers are always the most updated | There is more than 1 second delay in displaying answers in answer collecting page |
| 6 | The system shall stop refreshing the page and display all answers in the viewing page | The user is redirected to the viewing page | The website is still under answer collecting status |
| 7 | The order of the answers listed shall follow the rules. | The order of the users should be: 'StevenXU', 'Piper' and 'Prof\_Michael' (from higher priority to lower) | The order of the users is different from the pass case |

## 5.6 Case - Coupon

### 5.6.1 Purpose

The purpose of testing on Coupon System is to validate whether it is a well-defined system for the instructor to reward coupons, use coupons, and for the student to view current coupon status. The system should be able to pass all test cases designed to handle all valid operations. Any invaild operations should be detected automatically and well-managed.

### 5.6.2 Inputs

|  |  |  |
| --- | --- | --- |
| **Case** | **Description** | **Input** |
| 1 | Check whether the instructor can successfully reward a coupon for the student | 'input\_data': {  'student\_id': 1,  'course\_id': 3  }  Click the button “Reward” |
| 2 | Check whether the instructor can successfully use a coupon for the student | 'input\_data': {  'student\_id': 1,  'course\_id': 3  }  Click the button “Use” |
| 3 | Check whether the system is able to detect the coupon is already used up | Click the “Use” button multiple times to run out of all the coupons available |

### 5.6.3 Expected Outputs & Pass/Fail Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Expected Output** | **Pass** | **Fail** |
| 1 | The system shall allow rewarding a coupon to the student. | The “Reward” button should be disabled and labeled as “Rewarded”; the corresponding record in the database changes | The “Reward” button is not disabled or the database is not correspondingly changed |
| 2 | The system shall allow using a coupon for the student. | The used coupon record should increment by 1 | There is no change in the used coupon number |
| 3 | The system shall disallow any further use of coupon as soon as the available number is 0 | The “Use” button should be disabled and labeled as “N/A” | There is no change happened on the “Use” button |