StadiumPro-stadium reservation management system-test

V4.0

Test plan specification

Catalogue

1. Introduction [4](#_Toc23226)

[1.1 Objective 4](#_Toc15155)

[1.2 Background 4](#_Toc15155)

[1.3 User Base 4](#_Toc15155)

[1.4 Definition 4](#_Toc15155)

[1.5 Test Object 5](#_Toc15155)

[1.6 Test Stages 5](#_Toc15155)

[1.7 Test Tools 5](#_Toc15155)

[1.8 Reference Materials 5](#_Toc15155)

[2. Test Abstract 6](#_Toc444)

[2.1 Process Review 6](#_Toc16252)

[2.2 Test Execition 6](#_Toc32368)

[2.3 Test Cases 8](#_Toc19929)

[2.3.1 Functional Testing 8](#_Toc19134)

[3. Test Environment 11](#_Toc444)

[3.1 Hardware & Software Configuration 11](#_Toc16252)

[2.2 Network Topology 12](#_Toc32368)

[4. Tests Implementation 12](#_Toc444)

[4.1 Input box tests 12](#_Toc16252)

[4.2 MySQL connection tests 15](#_Toc32368)

[4.3 security tests 15](#_Toc32368)

[4.4 Compatibility tests 17](#_Toc32368)

[4.5 Usability tests 23](#_Toc32368)

[5. Tests Results 30](#_Toc444)

[5.1 Bug Trend chart 30](#_Toc16252)

[5.2 Bug Priority Distribution 31](#_Toc32368)

[5.3 Problem Types Distribution 31](#_Toc32368)

[5.4 Recently submitted defect diagram 32](#_Toc32368)

[6. Test Summary 33](#_Toc31872)

6.1 Usability [33](#_Toc32690)

6.2 Reliability [34](#_Toc32690)

6.3 Compatibility [34](#_Toc32690)

6.4 Security [35](#_Toc32690)

6.5 Functionality [36](#_Toc32690)

[7. Measurement 37](#_Toc31872)

7.1 Resource consumed [37](#_Toc32690)

**1 Introduction**

1.1 Objective

The test summary report is prepared for the following purposes:

* Through the analysis of test results, the evaluation of software quality is obtained.
* Analyze test processes, products, resources and information to provide reference for future test planning.
* Evaluate test compliance with test execution and test plan.
* Analyze the defects in the system and provide suggestions for bug repair and prevention.

1.2 Background

This system is mainly used to facilitate and secure the venue reservation and information management for fitness gym consumers. The overall task is to achieve convenient and secure user reservation and payment systems, as well as standardized and automated user information management. Its main functions include user registration/login, personal information modification, online booking, online/offline payment, and membership management. Due to the complexity of the system's functions and involvement in payment processes, comprehensive project testing becomes a very important task. Project testing can ensure stable system operation, normal function, and user data security, while also identifying and resolving potential issues and vulnerabilities in a timely manner, improving user experience and satisfaction. Our designed testing includes multiple aspects such as functional testing, performance testing, and security testing, where appropriate testing methods and tools are adopted for different testing requirements, and detailed testing plans and reports are developed to ensure system quality and stability.

1.3 User Base

Main readers: StadiumPro project manager, StadiumPro test manager.

Other readers: Personnel related to StadiumPro project (Stakeholders, developers etc.)

1.4 Definition

Severe bug: The following defects occur and the test is defined as a critical bug:

* The system does not respond, is in the state of crash, need other manual repair system can be restored.
* When a menu is clicked, "The page cannot be displayed" is displayed or an exception error is displayed.
* After an operation (such as adding, modifying, or deleting) is performed, The page cannot be displayed or an error message is displayed
* If no required field is entered during verification, The page cannot be displayed or an error message is displayed
* The system defines fields that cannot be repeated. After repeated data is entered, The page cannot be displayed or an error message is displayed

1.5 Test Object

The tested objects in our designed tests includes web pages(HTML, CSS, JavaScript), database(MySQL), web security and web services.

1.6 Test Stages

Our test process can be divided into 4 stages, At the end of each sprint, we summarize the test round, discuss the improvement plan, and make plans for the next test.

* **The first stage:** Finished the unit tests for basic functionalities (Registration and Login) and adjustment for the layout of the web pages to maximize user experience.
* **The second stage:** Completed the test code of other basic functionalities (membership payment, edit information module), we also adjusted the showing sequence of different pages to improve usability. Part of the input box tests had been carried out.
* **The third stage:** Designed the unit tests for some important functionalities(booking, receipt handling and income visualization module), adjusted the layout of app pages, finished all the database connection tests. Complete the tests for all the input box.
* **The last stage:** Finish the security tests for the web project, implemented some completability tests and usability tests.

1.7 Test Tools

* SpotBugs for the security tests.
* Selenium for automated testing.
* JMeter for load and performance testing.
* TestRail for test case management.

These tools have helped us to streamline our testing process and improve the efficiency and accuracy of our test results.

1.8 Reference Material

*Firefly host monitoring requirements specification*

*ISTQB (International Software Testing Qualifications Board) guidelines*

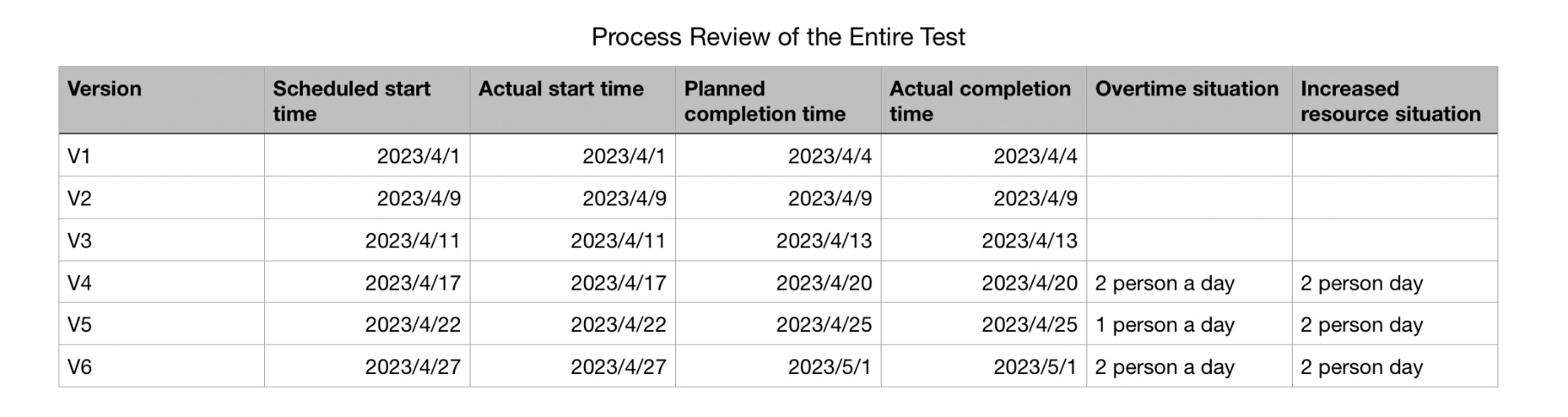
*OWASP (Open Web Application Security Project) top ten web application security risks*

**2 Test Abstract**

The StadiumPro-stadium reservation management system test will last for 33 days from April 1, 2023 to May 3, 2023. During this test, 85 test function points were tested, and a total of 2045 test cases were executed. On average, 24 test cases were executed for each function point. A total of 207 bugs were found in the test, including 32 serious bugs and invalid bugs. There are 14, with an average of 2.4 bugs per test function point.

The system releases a total of 6 test versions, among which V1-V3 is the planned iterative development version (for the baseline identification of the project plan), V4-V5 is the number of unplanned iterative versions, and V6 is the regression test version. Among them, the V1-V3 test progress is completed on time according to the project plan and the report is submitted. V4-V5 is an unplanned regression test version, and the test adds 2 working days of resources to complete the test on time. V6 is a regression test version, and the test adds 2 man-days of resources to complete the test on time.

The StadiumPro-stadium reservation management system test uses the JIRA defect management tool for defect tracking management. Each test version has a detailed bug analysis table and test report.

 2.1 Process Review

2.2 Test Execution

The test was carried out in strict accordance with the project plan and test plan, and the test of the test objects specified in the test plan was completed on time. The test strategy specified in the test plan is reflected in the test execution. During the test execution process, the system is completely tested according to the test plan and test cases.

2.2.1 Test Object

The test object includes all modules of the system, mainly involving venue management, personnel management, booking management, order management, ticket management and other modules. It is necessary to conduct a detailed and comprehensive test on each module.

2.2.2 Test Purpose

StadiumPro-stadium reservation management system is a key information system, which needs to ensure its stability, reliability, security and performance to meet the needs of users. Therefore, the main purpose of this test is to verify the correctness of the underlying code and the compatibility between each module, so as to ensure the integrity of the whole system.

2.2.3 Test Focus

According to the test strategy stipulated in the test plan, the focus of this test is to conduct a comprehensive test of the system's function, performance, security and usability, mainly including the following aspects:

* Functional Test

Verify whether each functional module of the system can operate normally according to the design requirements, including venue management, personnel management, reservation management, order management, ticket management, etc.

* Performance Test

Verify whether the stability and response speed of the system under high concurrency conditions can meet user needs.

* Security Test

Verify whether the security protection measures of the system can effectively protect the user's privacy and data security.

* Usability Test

Verify whether the user interface design of the system complies with general usability principles and the ease and comfort of user operations.

* Compatibility Test

The system needs to be able to work normally on different operating systems, browsers, and devices, and compatibility testing on different platforms is required.

* Reliability Test

In the face of various abnormal situations, the system needs to have good fault tolerance, and various abnormal tests need to be performed to ensure the reliability of the system.

* User-experience Test

The system needs to make users feel that the operation is simple and convenient, and the interface is beautiful and comfortable. User experience testing is required to ensure that users are satisfied with the ease of use and comfort of the system.

* Performance Stability Test

The stability and performance of the system under high load, high concurrency, and long-term operation need to be verified, and the performance bottleneck and performance optimization plan of the system need to be analyzed.

* Maintainability Test

The system should be easy to maintain, so that developers can quickly maintain and improve. It is necessary to evaluate the maintainability of the system and formulate an improvement plan.

2.3 Test Cases

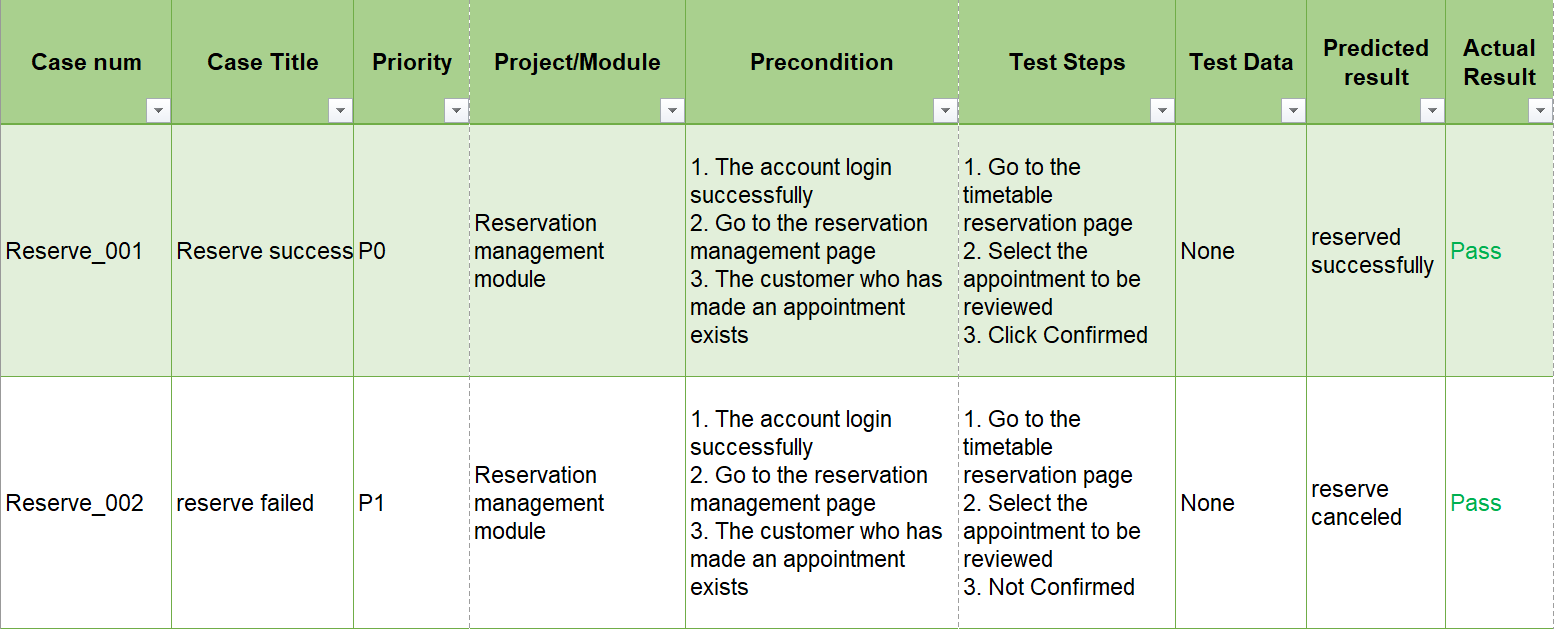
2.3.1 Functional testing

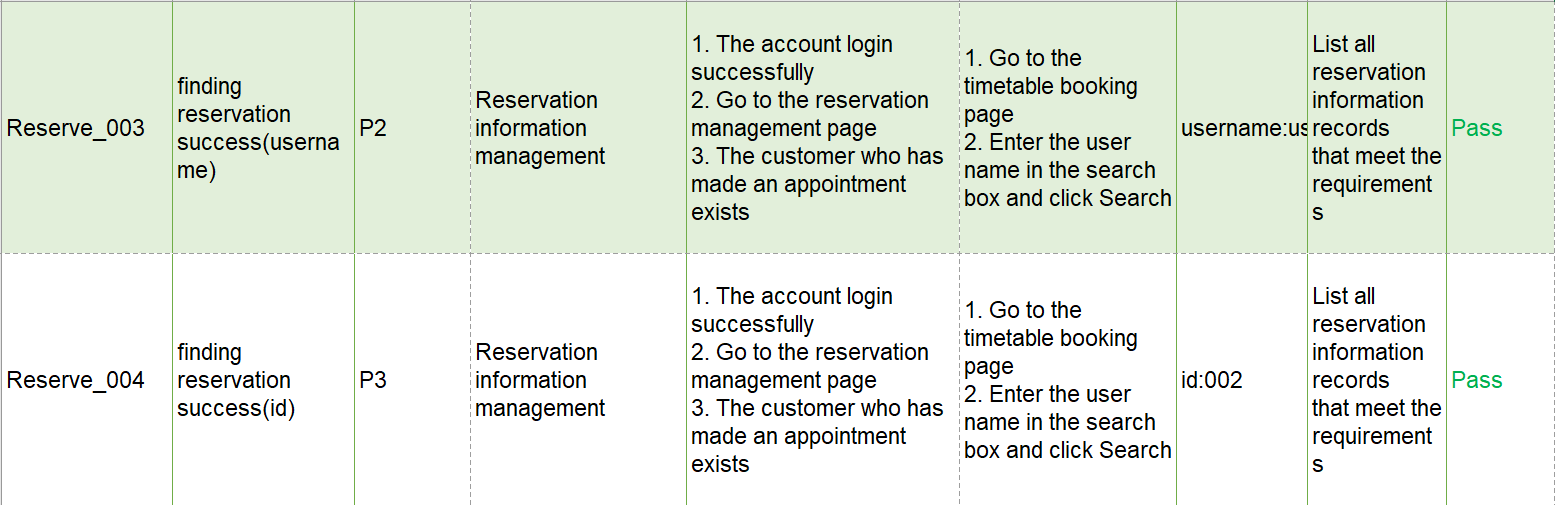
The main functionalities includes query, add, modify and delete.

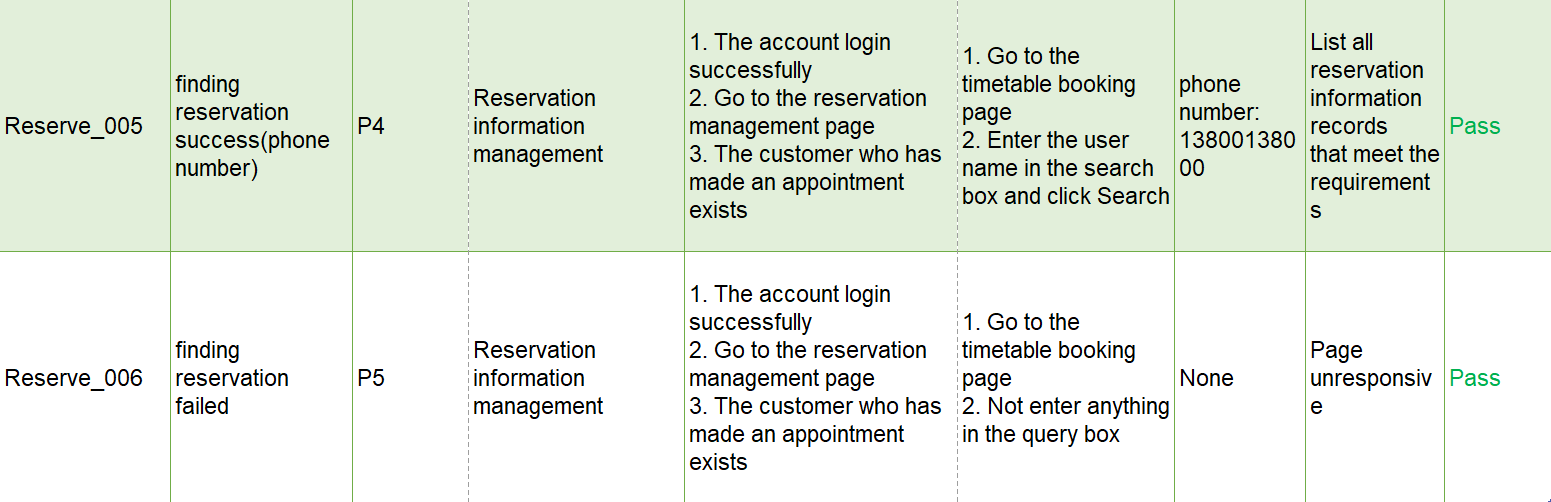
The important functionalities realized by the system include booking for specific activities and facilities, make payment for booking, membership management, make payment for membership.

The secondary functionalities includes receipt handling, visualization of statistics(the income of the stadium etc.)

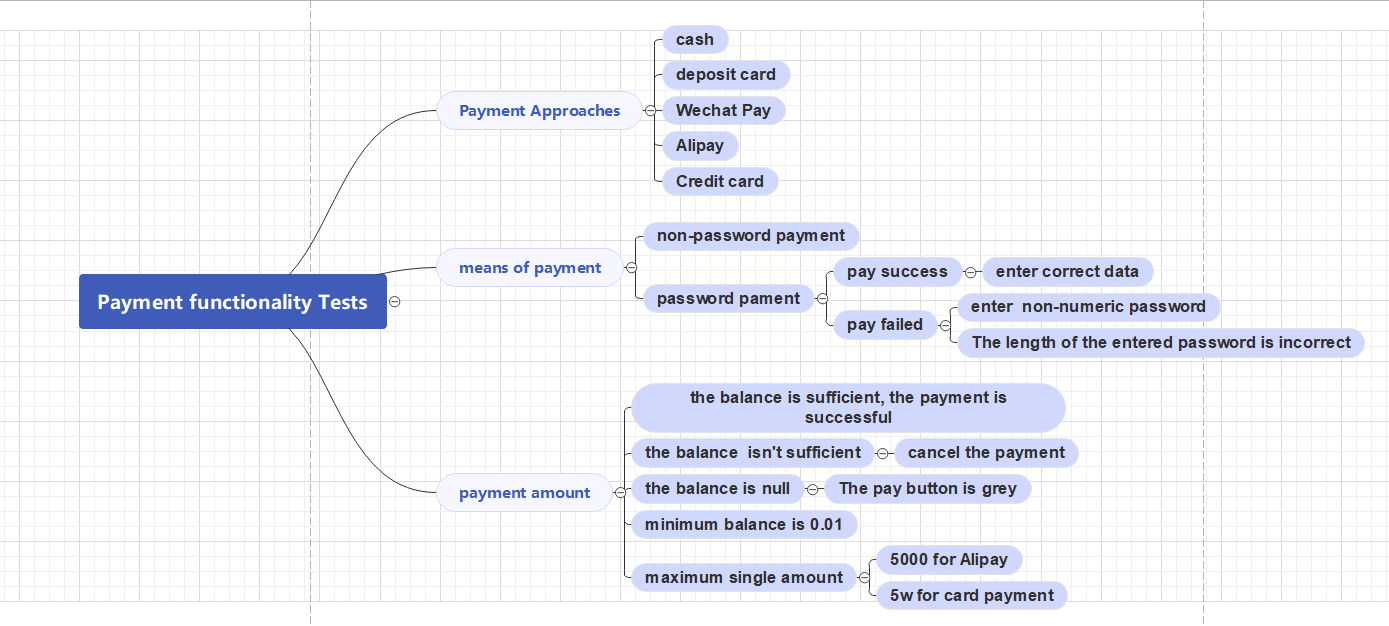
Reservation Module





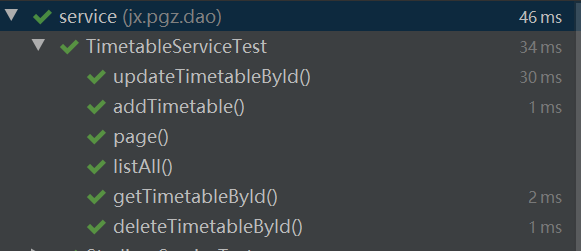


* Payment Module (Test point)

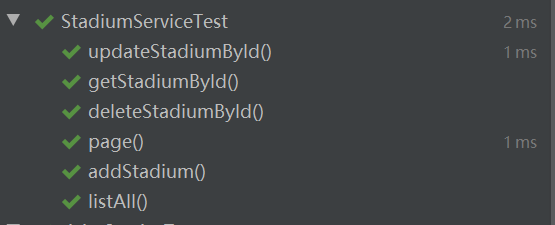


According to the test cases we specified and the mind map for the test points for important module, we then wrote the functionality unit tests for all the functions. Here are the result of our tests.

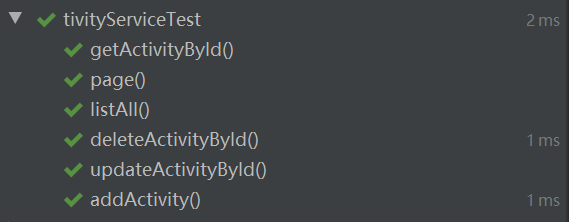
* For the service layer of the Timetable display functionality, we made 6 test cases for this unit test. The result shows that all the tests are passed.



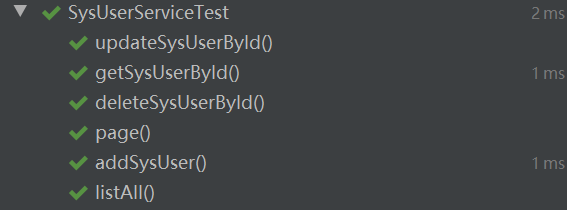
* For the service layer of the Stadium facilities functionality, we made 6 test cases for this unit test. The result shows that all the tests are passed.

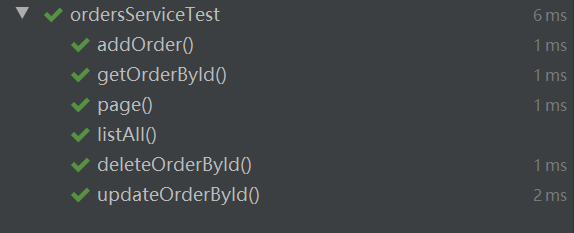


* For the service layer of the activity displaying functionality, we made 6 test cases for this unit test. The result shows that all the tests are passed.

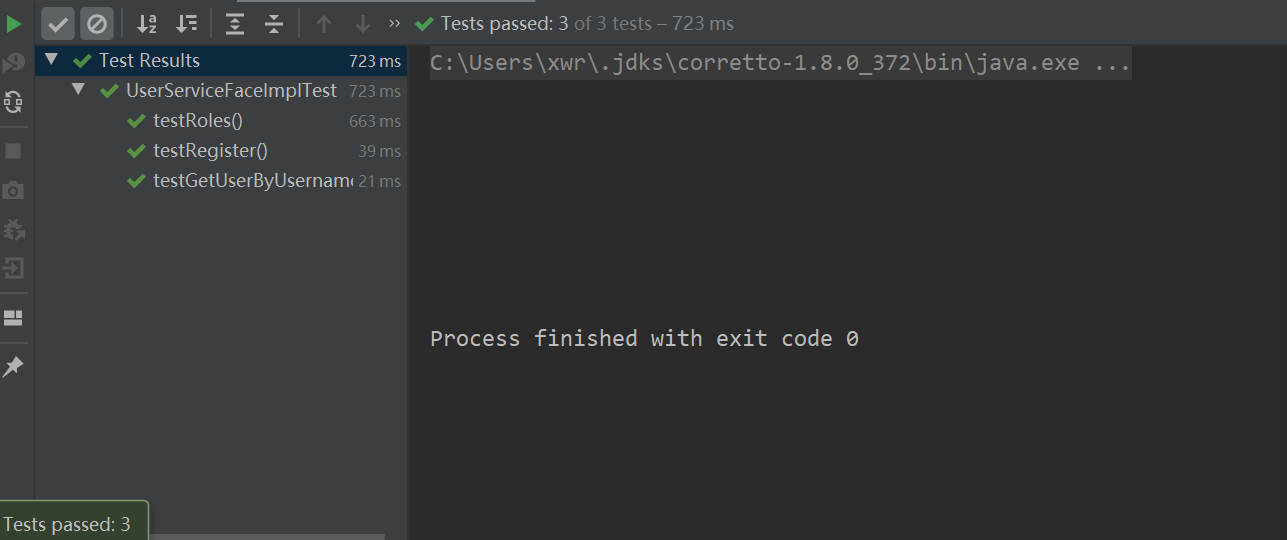


* For the service layer of the user data, we made 6 test cases for this unit test. The result shows that all the tests are passed. The same pattern of unit tests were implemented in the order information module, and all the tests in this module has passed.





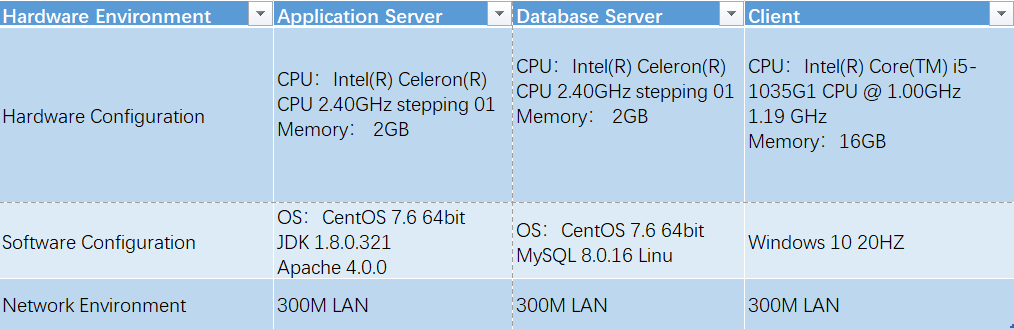
* Lastly, we wrote and run the implementation of the unit tests for user verification module (containing registration, login and roles allocation). The result shows that all the tests are passed.



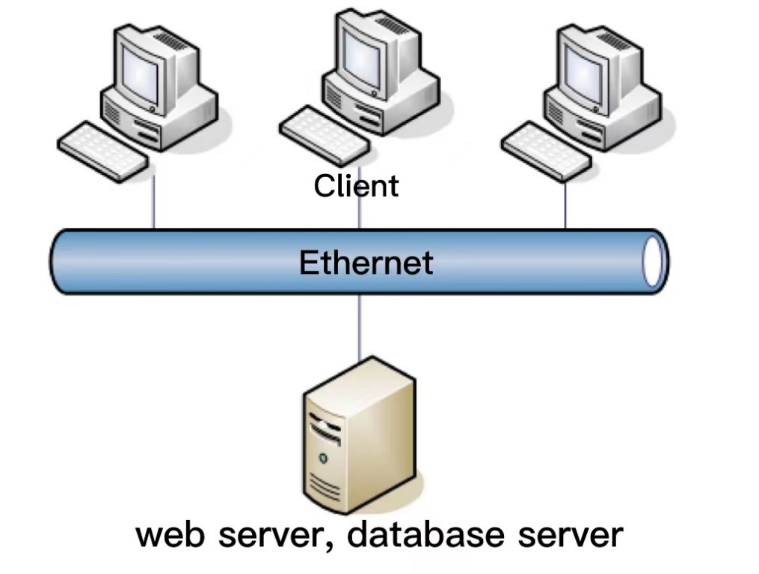
For the functional tests, we wrote a total of 33 unit tests, all of which passed, This is a great achievement and it ensures the stability of the project. It can help our team to identify and solve potential problems more quickly during the development process, and reduce unnecessary iterations and fixes. By writing high-quality unit tests, we can also enhance the readability and maintainability of the code, improve team collaboration efficiency, and facilitate rapid delivery and continuous integration/delivery.

**3 Test Environment**

3.1 hardware & software environment



3.2 Network topology

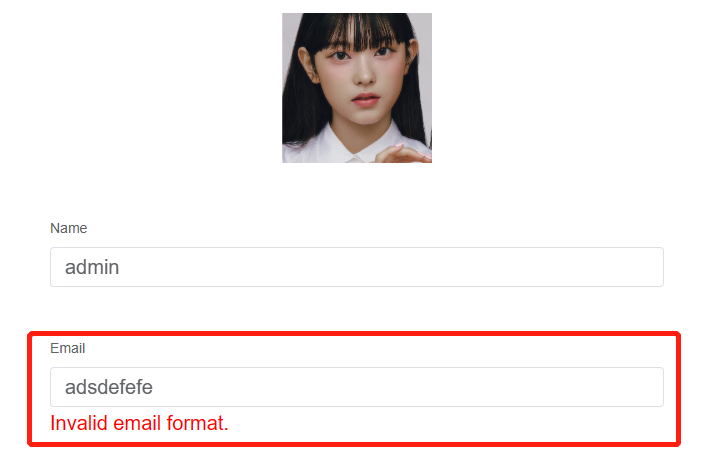


**4** **Test Implementation**

4.1 Input Box Tests

We designed different test cases for input boxed in different data types.

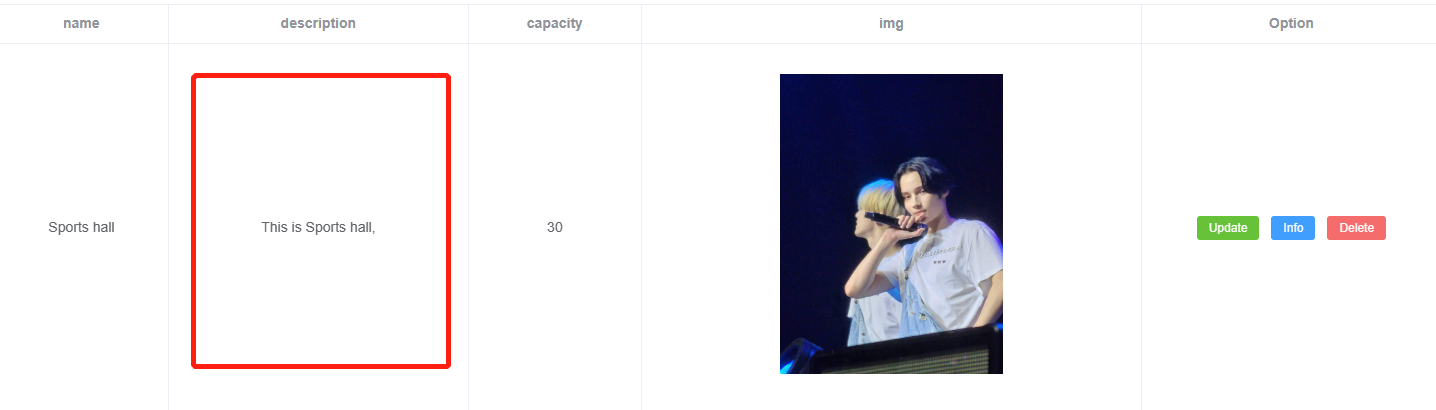
* **String type input** **box:** blank or space, English full Angle, English half Angle, number and special characters "~! &%...... &\*? [] {}".

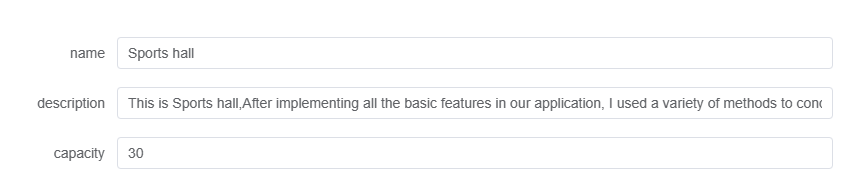


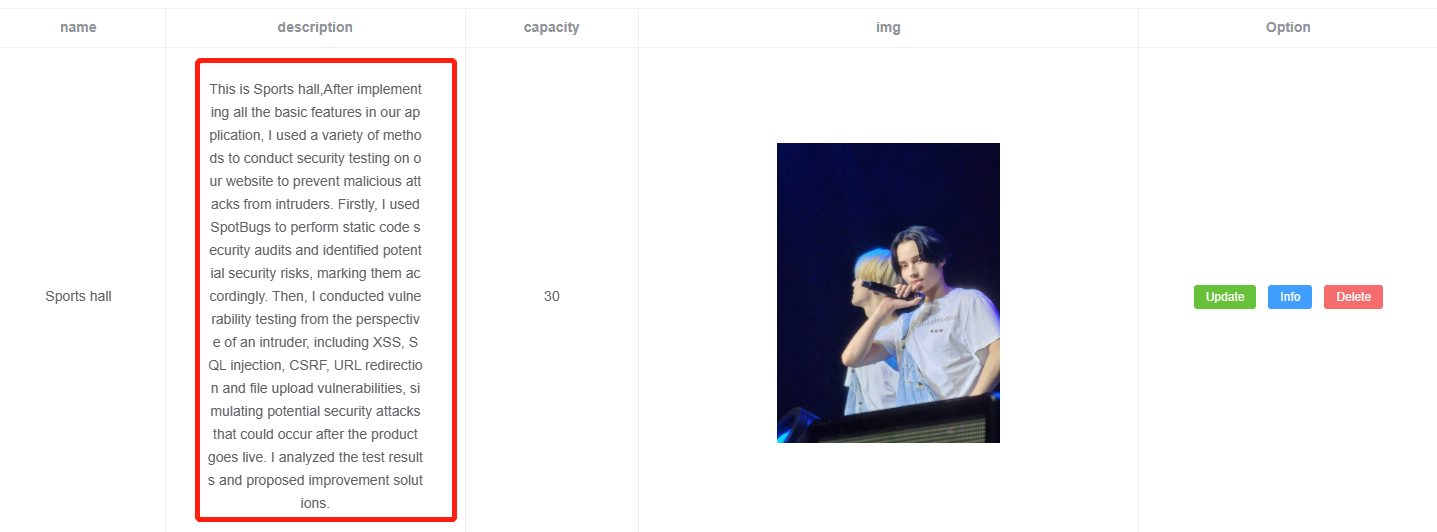
ex. Display error message if the string input is invalid

* **Length check**: minimum length -1, maximum length +1, minimum length, maximum length and input super-long string.
* **Space check:** Spaces before and after the characters, Spaces between the entered characters, The character is preceded by a space and Space after character.
* **Multi-line text box input:** only enter a newline, check whether it can be saved correctly (if yes, check the saved result; if not, check whether there is a normal prompt), whether allow carriage return and line feed,

save and then display the format that can save the input.

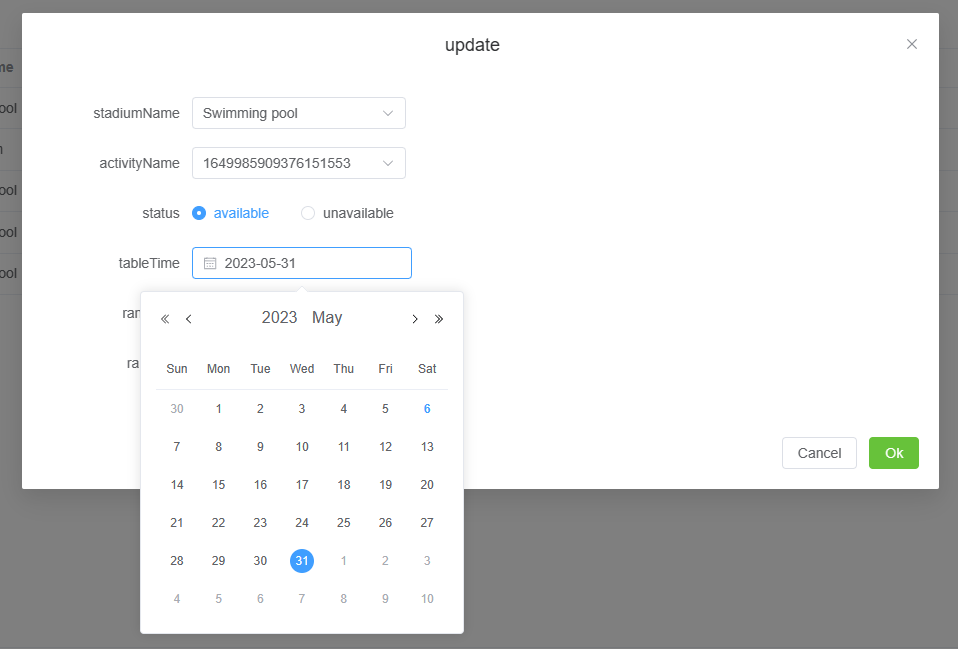


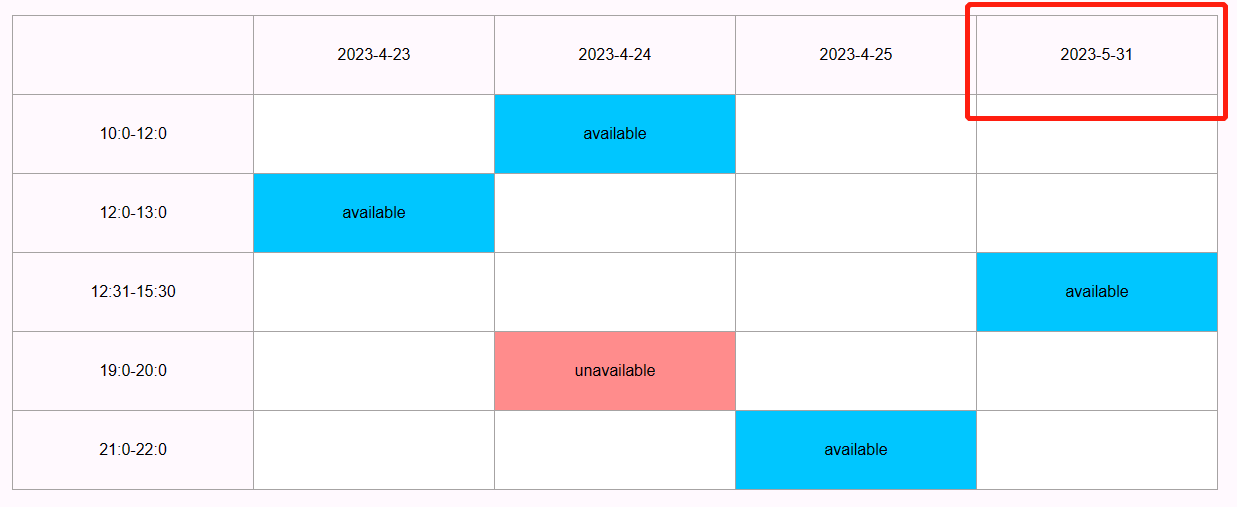




Ex. Allow for the multi-line input, ‘description’ field be updated without any mistakes.

* **Numeric Input box:** boundary values, digits, exceptional values and special characters.
* **Date type input box:** boundary values for day and month, leap years, exceptional values, special characters.





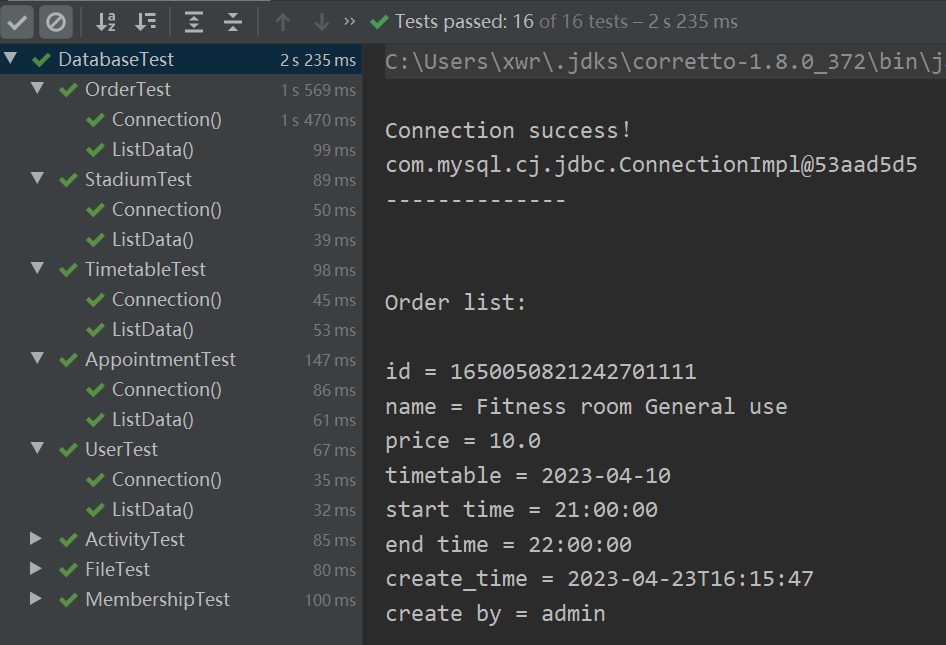
ex. allow for the boundary day, timetable be updated without mistake.

4.2 MySQL connection tests

We tested the database connection test for every table in our database. These includes:

* Tests for the order table
* Tests for the stadium table
* Tests for the Timetable table
* Tests for the Appointment table
* Tests for the User table
* Tests for the Activity table
* Tests for file storing and uploading
* Tests for membership data

We have made 16 unit tests for the database connection in total. All the tests were passed and the information of the specific table is printed at the terminal.

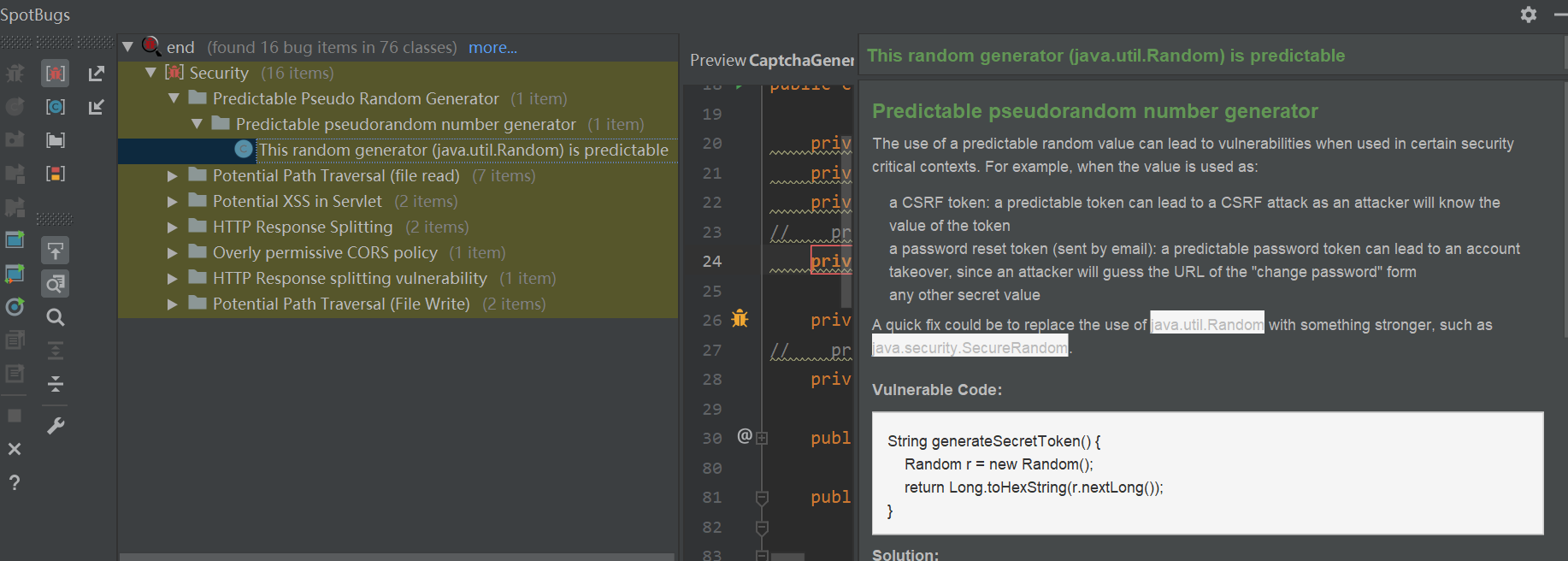


Once the test is passed, a successful prompt will appear at the terminal, followed the by the value of each attribute of each entity object in a unit of row of the table.

4.3 Security Test

We use SpotBugs plug-in for our Java Web application security audit that rely on static code analysis.

At the beginning, we got less ideal results. Through testing, we found 16 potential security vulnerabilities in static code, they could be categorized into 8 types:

* Predictable Pseudo Random Generator
* Potential Path Traversal
* Potential XSS in Servlet
* HTTP Response Splitting
* Overly permissive CORS policy
* HTTP Response splitting vulnerability
* Potential Path Traversal

This means the security of our application is relatively weak. Then we stimulate intruders to carry out security detection on our web application.

**a. XSS cross-site scripting attack**

Similar to SQL injection, XSS inserts malicious scripts through web pages, mainly using front-end HTML and JavaScript scripts. When users browse web pages, the implementation of the attack mode to control the user browser behavior. A successful XSS can obtain the user's cookie and use the cookie to steal the user's operation authority on the website. They can also obtain user contact lists, use the victim's identity to send large amounts of spam to specific targets, and so on.

XSS falls into three categories: storage (persistent XSS), reflection (non-persistent XSS), and DOM.

Test method:

* On the data input screen, enter: <script>alert(/123/)</script>. If a dialog box is displayed after the data is saved, it indicates that an XSS vulnerability exists.
* Change the URL request parameter to <script>alert(/123/)</script>. If a dialog box pops up, there is an XSS vulnerability.

**b. SQL injection**

SQL injection is used to trick the server into executing malicious SQL commands by inserting SQL commands into a Web form submission or entering a query string for a domain name or page request. SQL injection may cause the following hazards: Web pages and data are tampered, core data is stolen, and the database server is attacked to become a puppet host.

For example, some websites do not use pre-compiled SQL. Some fields entered by the user on the interface are added to the SQL. It is possible that these fields contain some malicious SQL commands.

Test methods:

On the page to be queried, enter a simple SQL statement, such as the correct query conditions and 1=1, and view the response results. If the results are consistent with the correct query conditions, it indicates that the application program has not filtered the user input, and the SQL injection vulnerability can be preliminarily determined.

**c. URL jump vulnerability**

URL jump vulnerability, also known as unverified redirection vulnerability, refers to that Web programs directly jump to URLs in parameters or introduce URLs of any developer into the page, leading to security problems.

Test method:

* We use the packet capture tool to capture requests.
* Capture the URL of 302, modify the target address, and check whether the URL can be redirected.

**d. File upload vulnerability**

File upload attack refers to the attacker uploads an executable file to the server and executes it. This type of attack is the most direct and effective. The uploaded files can be viruses, trojans, malicious scripts, webshell, and so on.

Test method:

Strictly verify the type and size of uploaded files, and do not upload files with malicious codes.

After verifying the execution permission of related directories, you can access all directories on the Web server using a browser and check whether the directory structure is displayed. If the directory structure is displayed, security problems may occur.

**e. Cross site forgery request attack (CSRF)**

CSRF uses the user identity to send malicious requests in the name of the user to complete illegal operations. For example, if A user browses and trusts the website A with CSRF vulnerability, the browser generates corresponding cookies, and the user visits the dangerous website B without quitting the website. Dangerous site B is asking to go to site A and make a request. The browser visits website A with the user's cookie information. Because website A does not know whether it is the request sent by the user or the request sent by the dangerous website B, it will process the request from the dangerous website B, thus completing the purpose of simulating the user's operation. This is the basic idea of the CSRF attack.

Test method:

* Open two pages on the same browser. If the permission on one page is invalid, check whether the operation on the other page is successful.
* Use the tool to send the request, do not add the referer field in the http request header, check the response of the returned message, should be repositioned to the error screen or login screen.

**4.4 Compatibility Test**

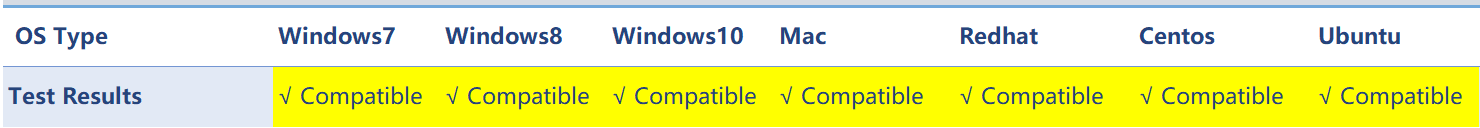
The compatibility test refers to the test that the system can run normally on a specific hardware platform, between different application software, on different operating system platforms, and in different network environments. The types of Web compatibility testing this time mainly include: compatibility testing of operating system, browser, resolution, antivirus software and firewall.

1. **Operating System Compatibility Test (Platform Compatibility)**

The simulated operating systems that need to be used in this compatibility test include win7, win8, and win10 under the Windows system; Redhat, Centos, and Ubuntu under the Mac system and the Linux system.

* Test Purpose: The purpose of the Operating System Compatibility Test is to ensure that the system can run on the specified operating systems without failure. This testing ensures compatibility with multiple operating systems, reduces the risk of any operating system-specific issues, and guarantees that the system performs consistently under different environments.
* Test Focus: he focus of this test is to check whether the system can support the current popular operating systems and whether it can run normally on these different operating systems. Additionally, it checks whether the functions can be used normally, whether the display is correct and whether various components and dependencies work correctly..
* Specific Operation: To perform the operating system compatibility test, first, we need to download and install the ISO image file of the corresponding operating system on Vmware, which can simulate the use environment of different operating systems. Once the environment is set up, we can test the system on each operating system and verify if it behaves correctly and consistently across all operating systems..

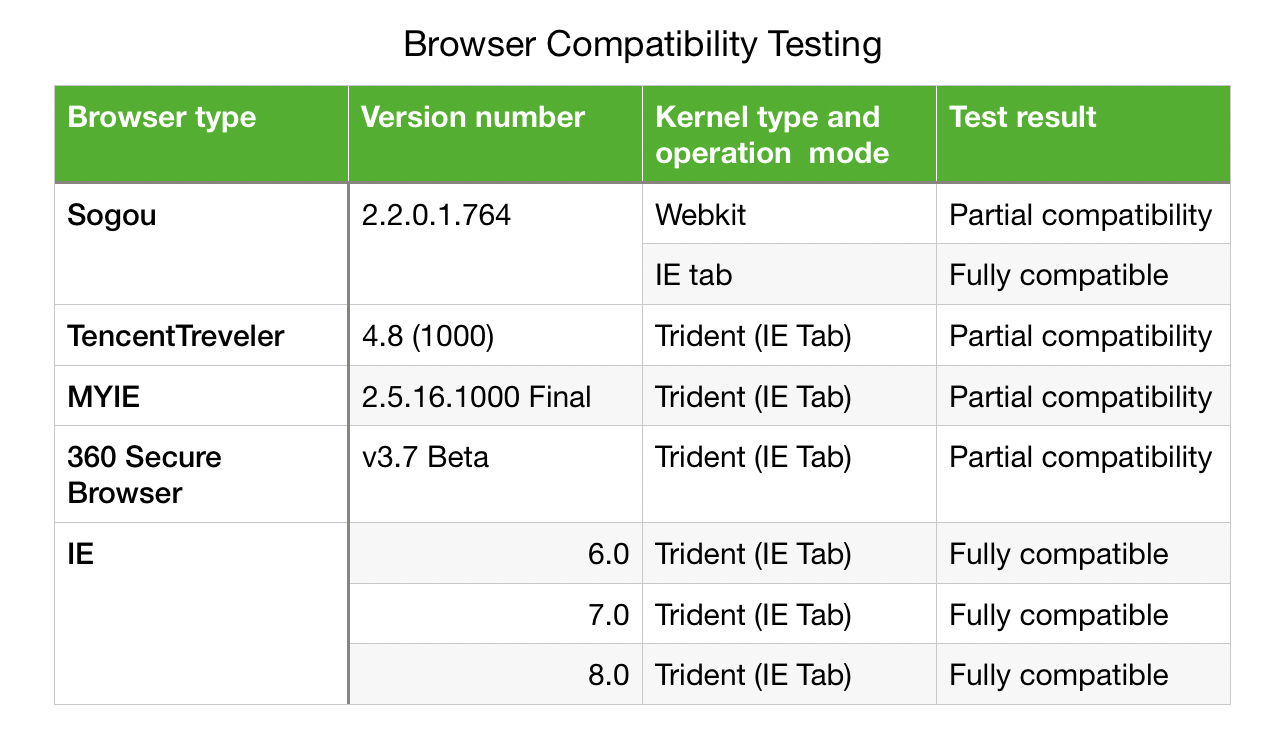
During the testing process, we will closely analyze the log for any operating system-specific issues and errors. The testing will include performing standard operations on the system, such as data input, database access, formatting, uploading, and downloading.

The test results are shown in the figure below:

1. **Browser Compatibility Test**

The compatibility issue of browsers means that different browsers use different kernels and support web page language standards such as HTML, and different user client environments result in different display effects of the final web client.

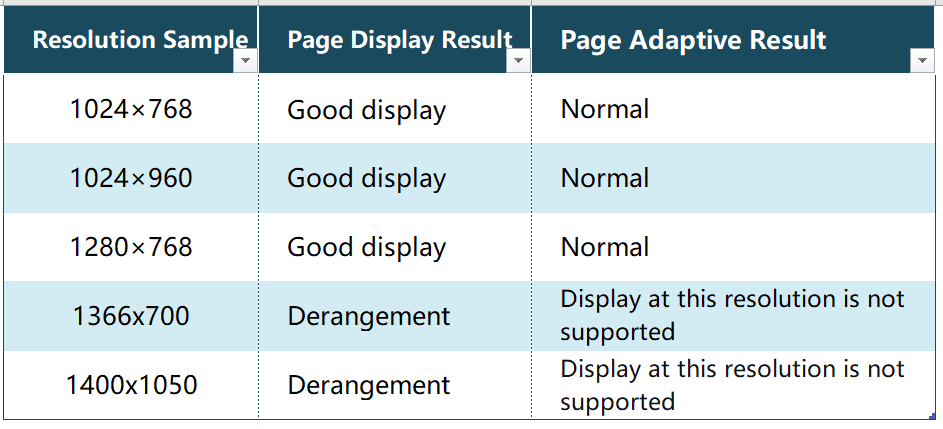
* Test Purpose: The purpose of the browser compatibility test is to ensure that the system can run on different web browsers without failure. The system should be able to run on the specified browser, and all functions should run normally.
* Test Focus: The focus of the browser compatibility test is to check whether the system is compatible with the major web browsers and whether it can run normally on these different web browsers. Additionally, it checks whether all functions and features provided by the system can be used normally, whether the display is correct, and whether various components and dependencies work as expected.
* Specific Operation: To perform the browser compatibility test, we need to test the system on each of the major web browsers such as Google Chrome, Microsoft Edge, Mozilla Firefox, and Safari. We will verify whether the system functions correctly and consistently across all web browsers. We will check for any discrepancies in the display, as well as whether the functions and features of the system work without issues on each browser. This will ensure that the system can provide a seamless experience to all users, regardless of their browser preference.

The test results are shown in the figure below:

1. **Resolution Compatibility Test**

The resolution compatibility test refers to verifying the display of page layout, the interface display, and related characters in different screen resolutions.

* Test Purpose: The purpose of the Resolution Compatibility Test is to ensure that the system can support various resolutions. It aims to guarantee that the system's display functionalities will work correctly and consistently under different resolutions.
* Test Focus: The focus of the resolution compatibility test is to check whether the system can adjust to different screen resolutions and whether it can display correctly under these different resolutions. Additionally, it verifies whether crucial functions and features provided by the system can be used normally, whether the display is correct, and whether various components and dependencies work correctly. This testing helps identify any discrepancies in the display of the system, such as pages, images, and texts, and ensures that the system provides an optimal viewing experience for the user.
* Specific Operation: To establish the resolution compatibility test, we need to adjust the display resolution of the monitor and test the system under different mainstream resolutions on the market like 1024×768, 1024×960 and 1366x700 and so on. We should verify that the system adjusts well to different resolutions and produces a clear and appropriate display without stretching or distortion of pages, images or texts. We need to ensure that the interface display and related characters are visible and legible on different resolutions. Additionally, we need to exercise the system's different functions under different resolutions, such as data input, image, text, and video display, etc., to ensure that they work as expected and function correctly across all supported resolutions.

The test results are shown in the figure below:

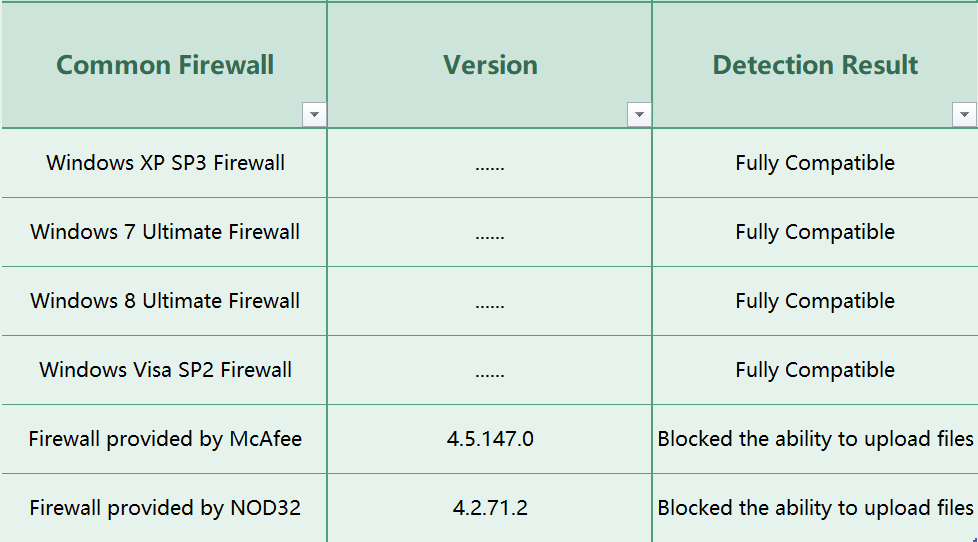
1. **Firewall Compatibility Test**

The Firewall Compatibility Test is a type of compatibility testing conducted to verify whether the system functions correctly under the default settings of the firewall. This testing evaluates the system's response to various firewall configurations that control application traffic to and from the system.

* Test Purpose: The purpose of the Firewall Compatibility Test is to ensure that all system functions related to data interaction, database access, uploading and downloading, and other activities, can be performed without interruption under the default settings of the firewall.
* Test Focus: The focus of the Firewall Compatibility Test is to verify whether the communication port of the system can function normally under the default setting of the firewall. It checks whether the system can communicate correctly with other system components as well as with other external dependencies such as APIs, servers, and web services. This testing aims to ensure that the system behaves consistently and correctly, regardless of the firewall configuration that a user may have in place.
* Specific Operation: To perform the Firewall Compatibility Test, we need to test the system on different operating systems, such as win7, win8, win10, Redhat, Centos, and Ubuntu. We will test the system on the basis of the operating system compatibility test and ensure that the website test is carried out under the firewall of different operating systems.

Our specific testing activities involve observing system behavior while performing tasks such as data interaction, database access, uploading and downloading. We will test the system's response to different firewall configurations by changing the system's default firewall settings to determine whether the system responds as expected and provides full functionality in different configurations. We will ensure that the tests conducted cover different scenarios, including basic firewalls to more complex firewall settings, to guarantee that the system operates optimally under different firewall configurations.

The test results are shown in the figure below:



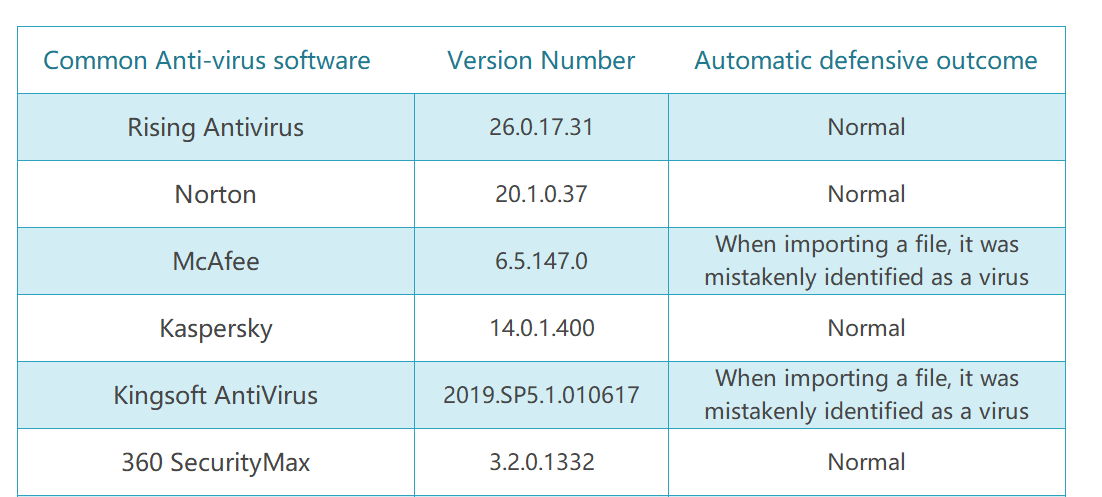
1. **Compatibility Test of Antivirus Software**

The Compatibility Test of Antivirus Software is a type of compatibility testing that focuses on verifying whether the system is compatible with existing and popular antivirus software and can coexist well with them on the user's computer.

* Test Purpose: The purpose of the Compatibility Test of Antivirus Software is to ensure that the system files and all system functions are not identified as viruses by anti-virus software. This testing aims to confirm that the system functions correctly with the existing antivirus software installed in the users' computer, which identifies and eliminates viruses protecting the computer from potential threats.
* Test Focus: The focus of the Antivirus Software Compatibility Test is to determine whether the system is compatible with existing and widely used antivirus software. This testing puts an end to the possibility of a conflict between the system and the antivirus software by verifying that they can coexist on the user's computer. Additionally, it aims to confirm that the antivirus software does not flag any system files as viruses, which may conflict with the system's functions and operations.
* Specific Operation: To carry out the Compatibility Test of Antivirus Software, we will manually download various popular antivirus software on the market, such as Avast, Norton, McAfee, and Kaspersky, among others. Then, we will verify that only one antivirus software is running in each antivirus software compatibility test, and the others remain closed. We will then install the system under test on the device with the antivirus software running and verify that it operates without any issue or conflict. The testing will include performing standard operations on the system, such as data input, database access, uploading, and downloading.

During the testing, we will closely monitor the antivirus software log, which contains notifications of potential system threats, to ensure that no system files are flagged as a virus. We will also check for any interference or errors that may occur when running the antivirus software and the system simultaneously.

In conclusion, the Compatibility Testing of Antivirus Software is crucial to ensuring the successful coexistence of the system and the antivirus software. Through this testing, we can verify that the system and the antivirus software can function correctly and without interference on the user's computer. By performing this testing, we can ensure that users continue to use both the system and antivirus software without fear of any adverse effects or conflicts. Ultimately, this testing ensures that the system is of the highest quality possible and performs and operates correctly and optimally under different scenarios.

The test results are shown in the figure below:

**4.5** **Usability Test**

This test is a usability test of the software, mainly focuses on the ease of understanding, ease of learning, ease of use, and attractiveness of the software.

* Ease of understanding refers to the degree of difficulty for users to understand the structure, function, guide, logic, etc. of the software and should focus on whether users can understand the software's functions and operation logic.
* Ease of learning refers to the ease of using the software or product (operation control, input, output), should focus on whether the user can quickly and easily learn how to use the software.
* Ease of use refers to the difficulty of users operating and controlling the products, should focus on whether the software interface design is scientifically reasonable and whether users can operate it easily.

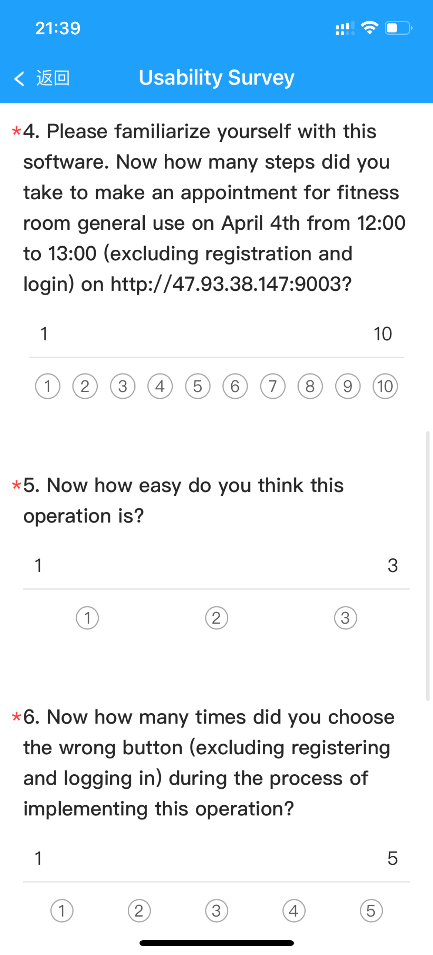
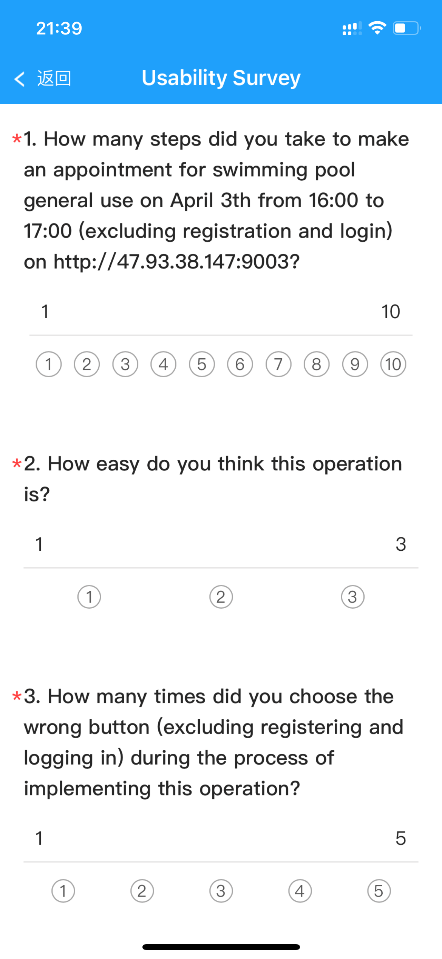
**4.5.1 Test Overview**

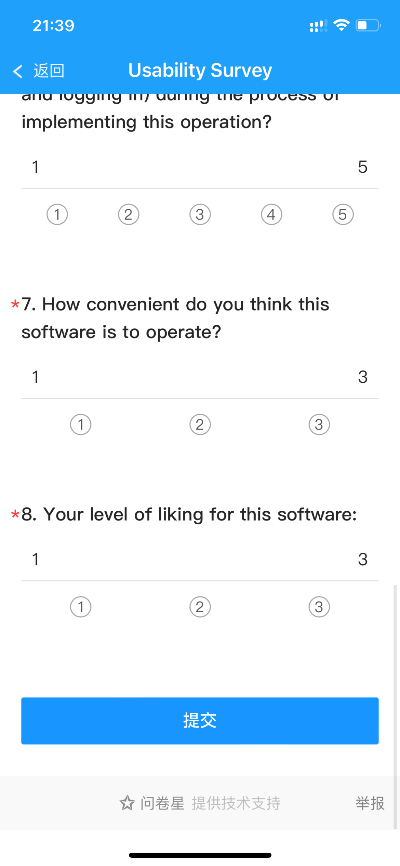
The test was conducted in the form of a questionnaire, consists of the following contents:

1. The number of steps and error rate for new users to complete tasks.
2. The number of steps and error rate for users familiar with the software to complete tasks.
3. User satisfaction.
4. The number of steps and error rate for users to complete tasks again after a period of time.

There are 30 volunteers from different groups (including age, gender, and experience level) participating. We have set up two specific scenarios for users to complete, and filled in the number of steps taken and errors made, as well as scoring the software's usability.

The questionnaire content is as follows:



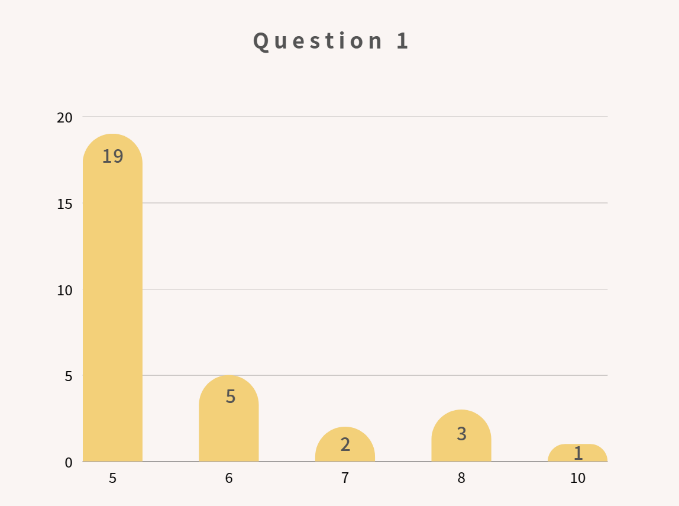
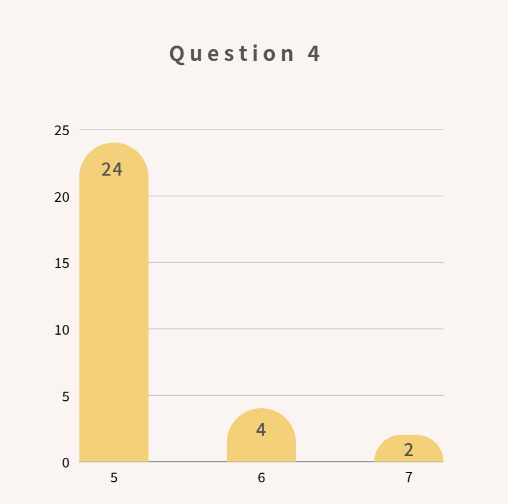


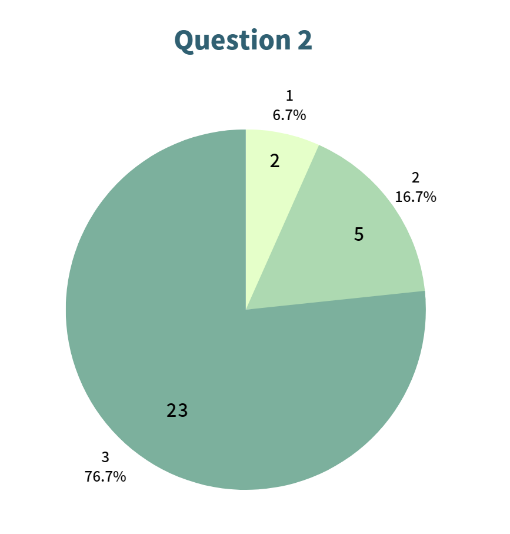
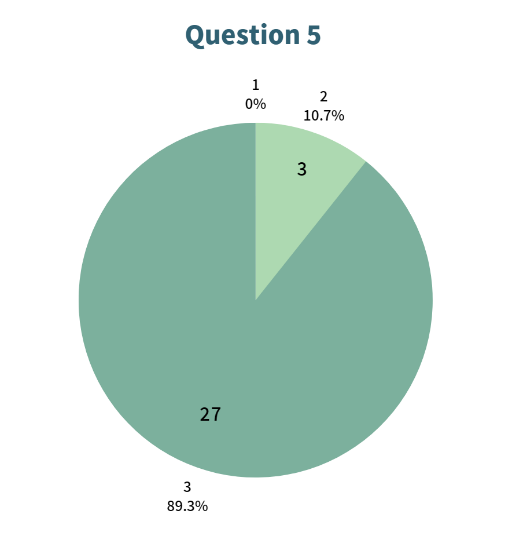
The test was divided into two sessions. The first session was held on April 3rd, and the questionnaire was used to test the ease of understanding, ease of learning, ease of use by analyzing the user's efficiency and error rate, and test the attractiveness of the software based on user ratings. The second session was held on April 20th, users operated the software again to test its memorability.

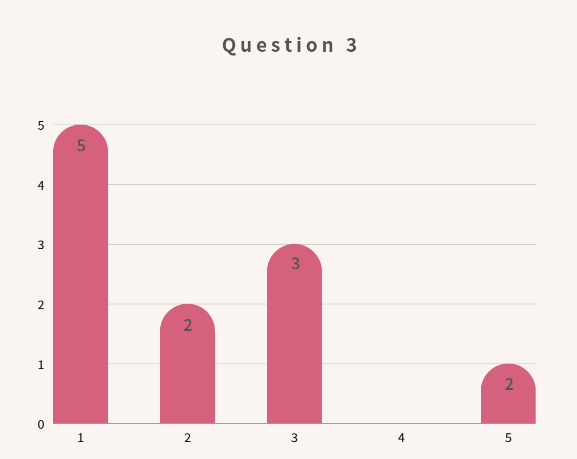
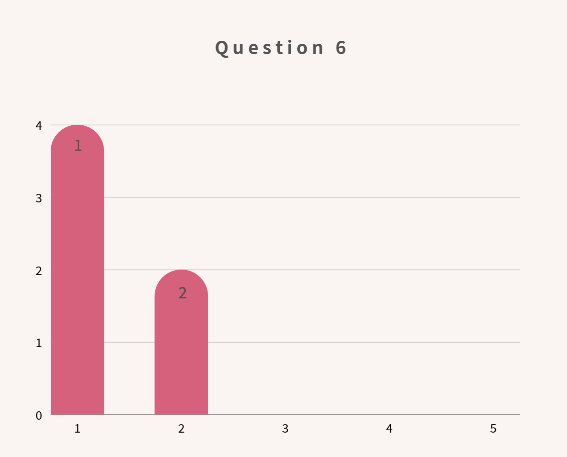
* + 1. **Test Result**

1. Task completion time: Most users can complete tasks within 5 minutes;
2. Error rate: Most users have not made any mistakes;
3. User satisfaction: Most users are very satisfied with the software interface and functionality.

The result is as follows:

** **

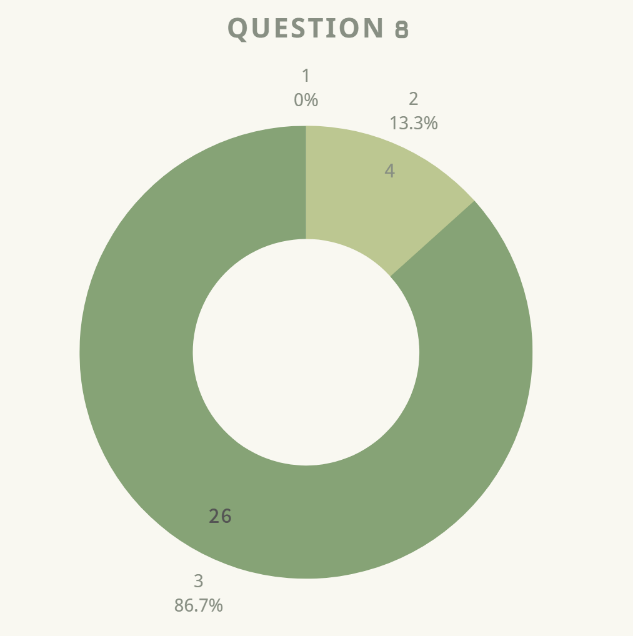
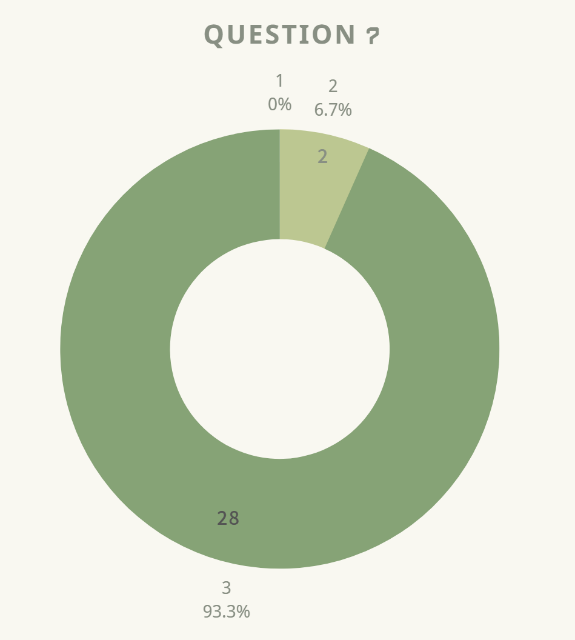
** **

** **

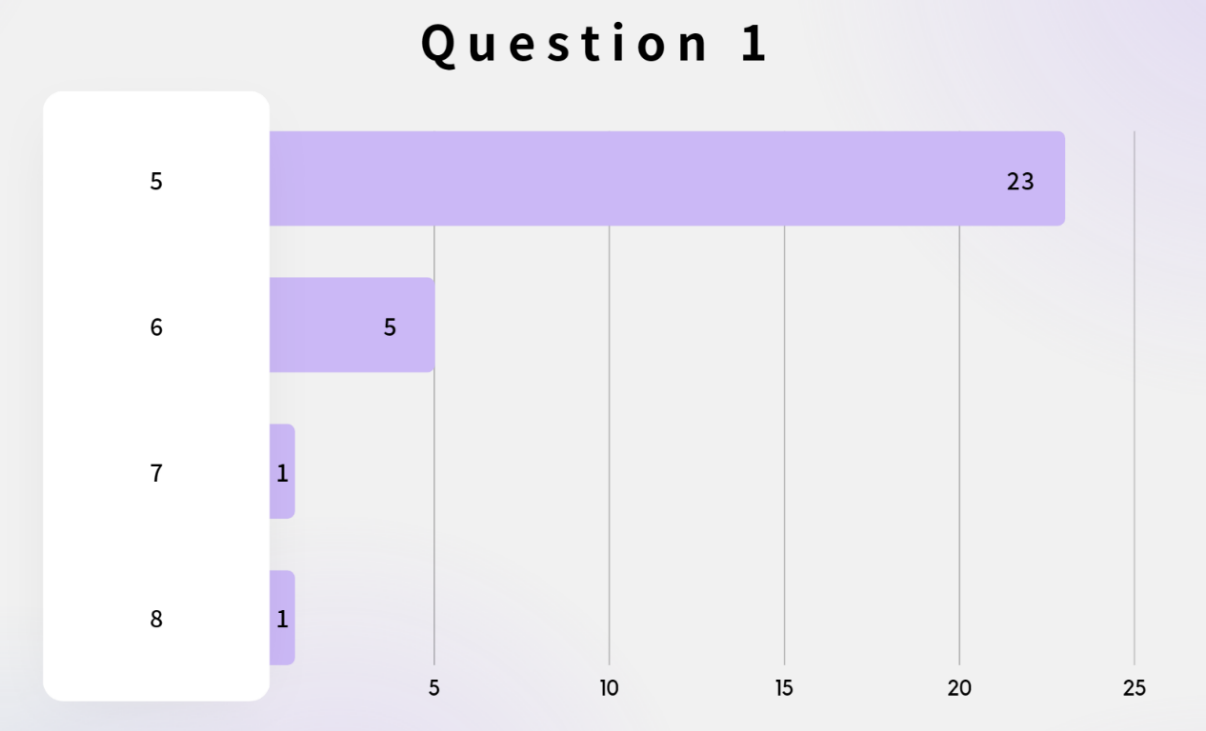
The left side shows the completion of questionnaire questions by new users, and the right side shows the completion of questionnaires by users who are familiar with the software.

From the chart, we can easily know:

* Most users can find the most direct way to make appointments without making mistakes.
* After becoming familiar with the software, users can basically operate the software without making mistakes.
* Most users think that the software is simple and easy to operate, and after becoming familiar with the software, more users think that the software is easy to operate.

****

The left image shows the user's convenience rating of the whole software, the right image shows the user's liking rating. Most users think that the software is convenient and express their love for it.



The above figure shows the number of steps required for users to make an appointment by operating the software again on April 20th. It can be seen that most users can still complete the appointment operation without any errors, and only a few users have made errors less than three steps.

* + 1. **Test Conclusion**

In summary, we think that the software has high usability and can meet the main needs of users. However, in order to further improve the user experience of the software, the software should consider the following points:

1. Simplify labels and instructions: By ensuring that the language of labels and instructions is simple and clear, it can help users better understand software functions. Additionally, adding more instructions and error message prompts will also be helpful.
2. Provide more information: Some users hope to obtain more venue information and event notifications in the software. Therefore, we suggest adding this information to the software to meet the needs of different users.
3. Consider accessibility design: In order to enable more users to use the software, we recommend developers to consider accessibility design, such as adjusting font size, color contrast, and keyboard operation.

In short, the software already has high usability and user satisfaction. But we can still continue to improve to improve the user experience of the software and attract more users to use it.

* 1. **Login Test**

The purpose of login testing in software development is to validate the functionality and security of the login process for a software application. The login process is often the first point of entry for users into an application, and it's important that it works correctly and securely.

During login testing, a tester or QA engineer will verify that the login process is working as intended, including verifying that the correct user credentials (such as username and password) are accepted and that the user is granted access to the appropriate parts of the application. The tester may also check for error messages and ensure that the application behaves appropriately in the event of incorrect login attempts.

In addition to functional testing, security testing is also an important aspect of login testing. This may involve testing for vulnerabilities such as SQL injection, cross-site scripting (XSS), and other types of attacks that could compromise the security of the application and put user data at risk.

Overall, login testing is an essential part of software development to ensure the security and usability of the application.

**4.6.1 Test Design**

The login test cases are design as:

* Functionality

1. Enter the correct user name and password to check whether the login is successful.

2. Enter the correct user name and incorrect password to check whether the login fails and prompt the incorrect user name or password.

3. Enter an incorrect user name and password to check whether the login fails and display a correct message.

4. Enter an incorrect user name or password to check whether the login fails and a message is displayed indicating that the login is correct.

5. If either the user name or password is empty, check whether the login fails and the information is correct.

6. If the user name and password are empty, verify that the login fails and the information is correct.

* Security

1. Whether the user password is encrypted during network transmission.

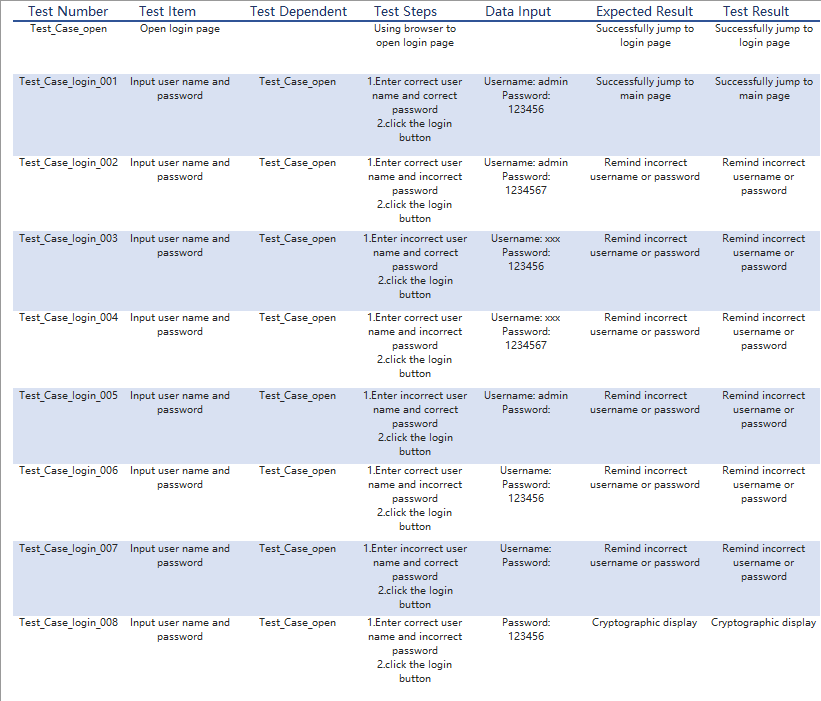
2. If you do not log in, enter the URL in the browser to check whether the login is repeated.

3. Whether the password input box does not support copy and paste.

* Compatibility

Verify the display and function of the login page in different browsers.

* + 1. **Test Results**

The security test and compatibility tests both run smoothly and pass. The results of the functionality test are shown below:

**4.6.3 Test Improvement (Change password)**

The password change tests are designed as:

1. Enter the new password, the old password and the correct new password to verify whether the password is changed successfully.

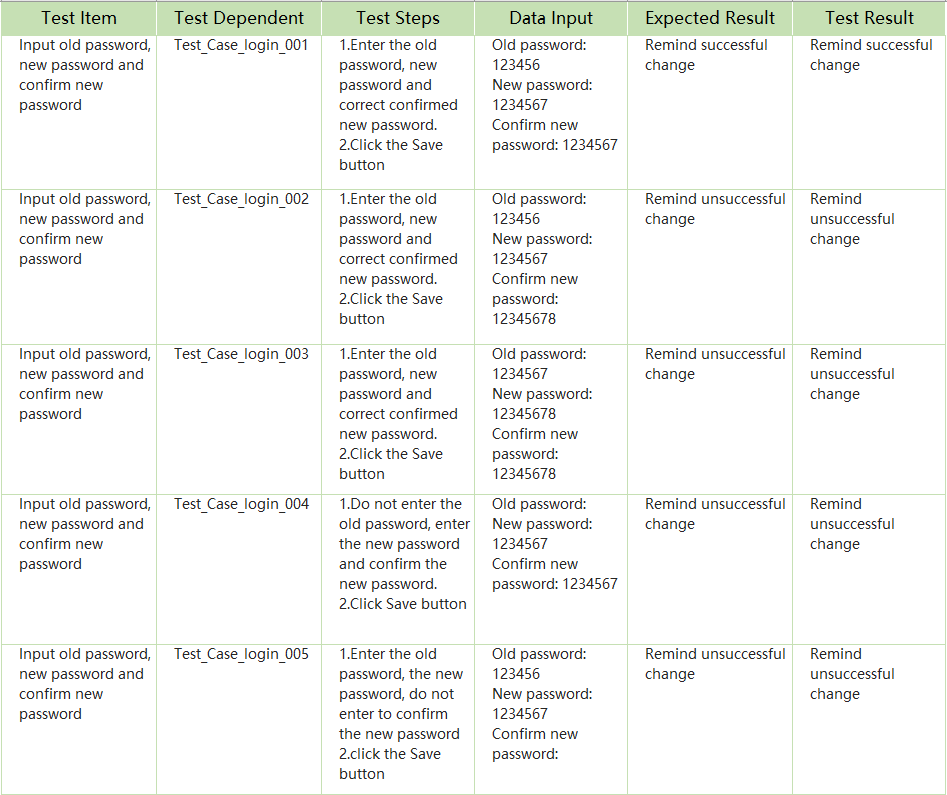
2. Enter the new password, the old password, and the incorrect new password. Verify that the new password fails to be changed and a message is displayed.

3. Enter an incorrect old password, a new password, and a correct new password to verify whether the modification fails.

4. Old password, new password, confirm that one of the three new password is empty, verify whether the modification fails.

5, old password, new password, confirm that the new password three are empty, verify whether the change fails.

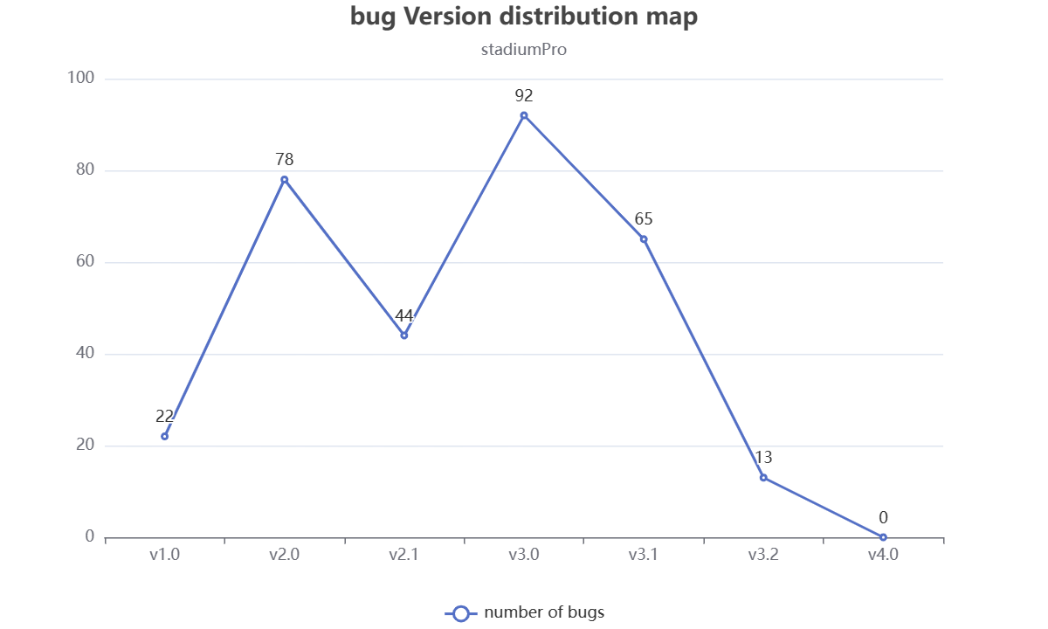
**4.6.4 Test Results (modified)**

The results of these tests are shown below:

**5.Tests Results (all-phase)**

**5.1 Bug Version Distribution map**

We have released 7 versions of tests in total, a total of 314 bugs were found for all versions.

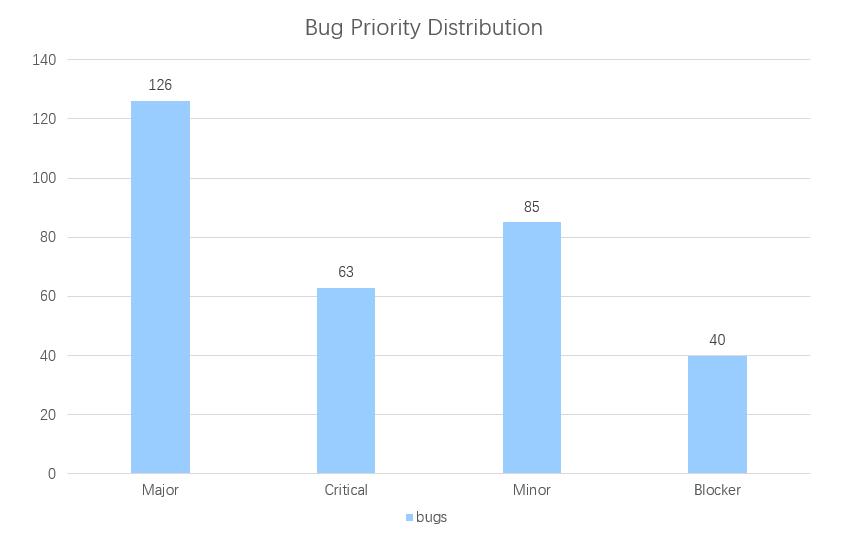


It can be seen from the bug version distribution diagram that, the quality of v1.0-v3.0 is very unstable, with the maximum number of bugs reaching 92, while V1.0, as the first version, has 22 bugs. On the basis of V2.1 verifying all the bugs previously discovered, the quality performance of the remaining bugs was not stable enough at 44. In V3.0, the number of bugs suddenly increased to 92 due to the addition of file upload, QR code generating, receipt handling and other functions. The number of bugs in version V4.0 will gradually reach zero as the version iterates.

5.2 Bugs priority distribution

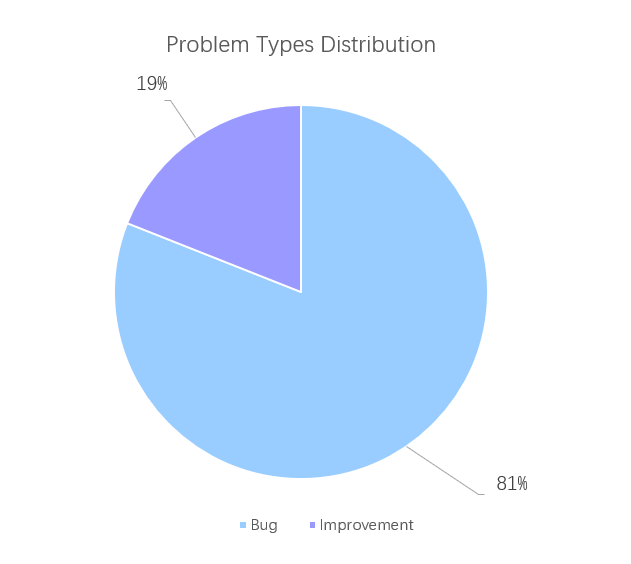
Bugs found in the test mainly focus on the incomplete function level major, which is a general function defect. However, during the test, there were 41 critical bugs involving program crash, program startup failure etc. Serious bugs are mainly manifested in the following aspects:

* The main functions of the system has not been implemented.
* Local database data volume is relatively large when the program crashes.
* The logic of major functions of the system is confused, resulting in unexpected bugs.
* The program cannot start because the background process does not stop after the program is shut down.

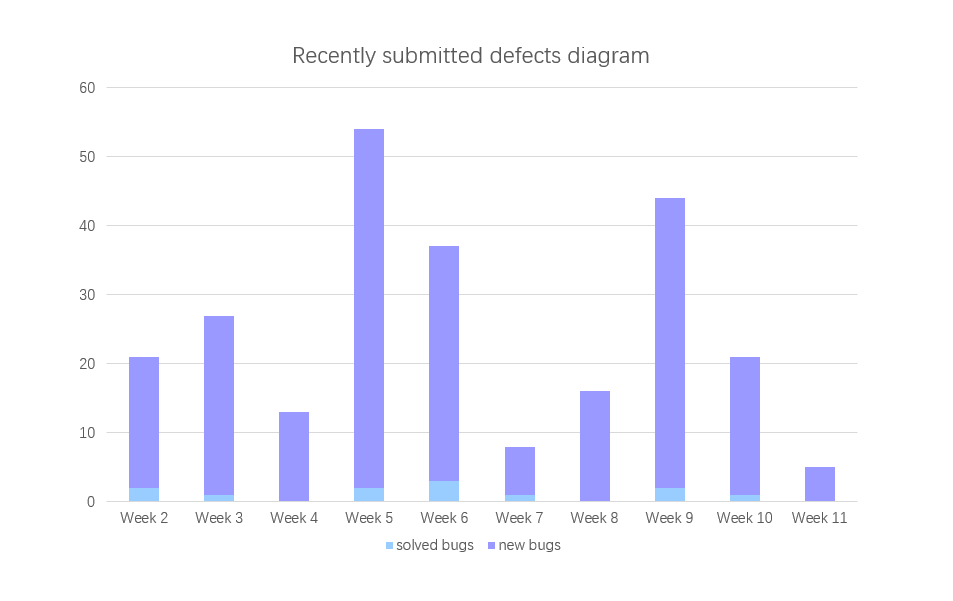


**5.3 Problem Types Distribution**

The problem types of the system are mainly distributed in the testing process and maintenance process to find defects and bugs affecting the system operation and improve the existing system functions. Bugs as a percentage of all problem types: 81%, improvement as a percentage of all problem types: 19%. The results on the figure show that the system takes comprehensive consideration in the process of requirement collection and program design, and there are few function design omissions.



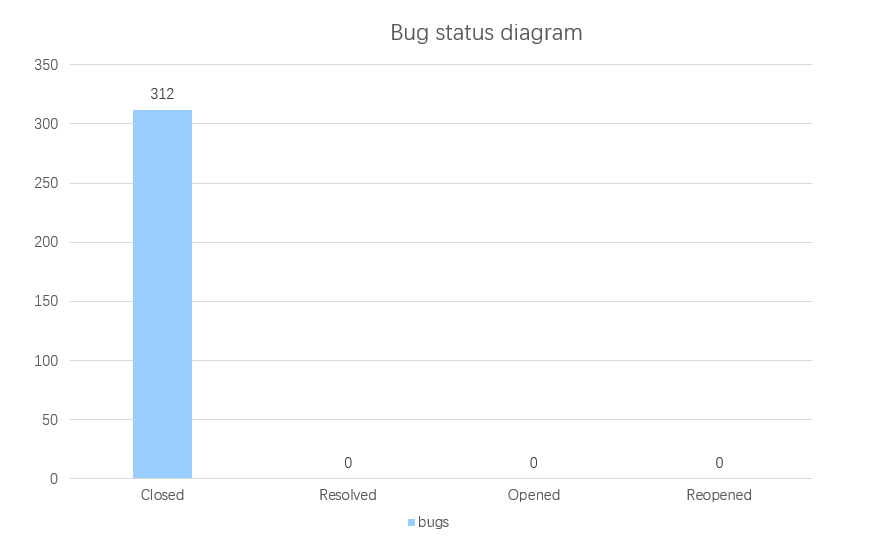
**5.4 Recently submitted defects Diagram**



As it can be seen from the figure above, bugs submitted and solved in the statistical 10 weeks are in an ideal state. The bugs submitted at present can be fixed in a very short time, and the number of bugs solved gradually decreases to 0 as the version becomes stable and the number of bugs solved gradually decreases to 0. The whole process is a normal software version iteration process.

5.5 Bug Status Diagram

As it can be seen from the bug status diagram below, there are 0 open bugs and 0 reopened bugs. There are 2 solved bugs, mainly the interface usability bug submitted in V2.0, while the other 312 bugs are verified and fixed and closed. The number of remaining bugs in the whole system reaches the test end standard.



**6. Test Summary**

In today's rapidly changing technology landscape, software systems play an important role in various industries. For sports venues, implementing a reliable and user-friendly booking management system can significantly enhance the customer experience and improve operational efficiency. To ensure that the StadiumPro sports venue booking management system meets these goals, we conducted comprehensive testing, including usability, reliability, compatibility, security, and functionality tests. This report summarizes our findings, including strengths and weaknesses of the system, and identifies areas for improvement.

**Usability:**

In terms of usability testing, the system performed well. It was designed with good user interaction, page layout, and navigation, making it easy to use for most users without requiring too much technical background knowledge.

However, during testing, some usability issues were discovered, such as:

* Unattractive interface design: The system interface has some aesthetic issues that could impact user experience.
* Poor comprehensibility of input/output fields: Some input/output field meanings are not clear, which may lead to user misunderstandings.
* Mixing Chinese and English: Using mixed Chinese and English on the same interface may confuse users.

**Reliability:**

The system demonstrated stable and reliable performance during our tests. No crashes or errors were found during database connection testing or functional testing, and some common error risks were avoided. However, there were still some reliability issues that need to be addressed, such as:

* Unstable database connections: In rare cases, the database connection may encounter problems that require more detailed troubleshooting and resolution.
* Operation delay: In some high-load situations, the system's response speed may slow down, so further optimization is needed.

**Compatibility:**

The system performed relatively well in compatibility testing. It can run normally on mainstream browsers and operating systems, and there were no significant compatibility issues. However, there were still some compatibility issues that need improvement, such as:

* Rarely, the system may encounter compatibility issues on some low-version browsers.
* The system's resolution adaptation for different devices is not perfect, and display effects may be lacking on some devices.

**Security:**

In terms of security testing, the system has implemented some basic security measures to effectively prevent some common security risks. However, there are still some security issues that need to be addressed, such as:

* Insufficient password complexity requirements and encryption mechanisms make the system vulnerable to brute-force cracking and information leakage attacks.
* Lack of mandatory user identity authentication may lead to unauthorized access and other security risks.

**Functionality:**

During functional testing, the system's performance was good, and it possesses all required functions with simplistic yet clear operations. However, there are still some functional issues that need improvement, such as:

* Some features require further improvement and optimization, such as choosing time slots for venue reservations.
* In extreme cases, the system may encounter data inconsistency problems that require more detailed troubleshooting and resolution.

In conclusion, the StadiumPro sports venue booking management system performed well in usability, reliability, compatibility, security, and functionality, but there were still some issues that need to be addressed continuously and optimized accordingly. Enterprises should maintain a rigorous attitude and perform comprehensive testing and verification on various aspects to provide users with a more stable, efficient, convenient, and secure sports venue booking management system.

**7 Measurement**

**7.1Resource consumed**

|  |  |
| --- | --- |
| Testing time | 2/3/2023-6/5/2023 66 days in total |
| Testing manpower | 1person x 14days + 1person x 57days = 72 person\*day |
| Hardware resources | Server: 1 PC  Client: 2 PC |