**CS673 Software Engineering** 

**Team 1 - Terriers Mall**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Haoran Dai | Team & Security | *Haoran Dai* | Nov. 4, 2023 |
| Qinchen Gu | Q & A | *Qinchen Gu* | Nov. 4, 2023 |
| Xiaocheng Kang | Requirement & Configuration | *Xiaocheng Kang* | Nov. 4, 2023 |
| Bhargav Sai | Implementation | *Bhargav Sai* | Nov. 4, 2023 |
| Huanzhou Wang | Design | *Huanzhou Wang* | Nov. 4, 2023 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1** | **Qinchen Gu, Haoran Dai** | **Nov.4, 2023** | **Nov.4, 2023** |
| **2** | **Huanzhou Wang** | **Nov. 8, 2023** | **Added 5 Manual Testing Cases** |
| **3** | **Haoran Dai, Qinchen Gu** | **Nov.8, 2023** | **Add the Testing Summary Part, Add 5 Manual Testing Cases** |
| **4** | **Haoran Dai** | **Nov.8,2023** | **Add 1 Manual Testing Case** |

[Testing Summary](#_heading=h.gjdgxs)

[Manuel Tests Reports](#_heading=h.30j0zll)

[Automated Testing Reports](#_heading=h.1fob9te)

[Testing Metrics](#_heading=h.3znysh7)

[References](#_heading=h.2et92p0)

[Glossary](#_heading=h.tyjcwt)

# Testing Summary

The TerrierMall Project is currently in the midst of the testing phase, with a focus on ensuring the application meets its functional, performance, and security requirements. This interim report summarizes the testing activities that have been planned and those that have been executed to date.

* + Test Planning

**Objective**: Establish a structured approach to the testing activities.

**Participants**: All Members.

**Techniques**: Analysis of requirements, risk assessment, and resource allocation.

Outcome: A comprehensive test plan has been developed, outlining the scope, strategy, and schedule for all testing activities.

* + Unit Testing

**Objective**: Validate the functionality of individual units of code.

**Participants**: Haoran Dai for now.

**Techniques** Used: Test-driven development (TDD) using JUnit for Spring Boot.

**Progress**: Unit testing for the 'Post' component of the backend has been completed successfully.

* + Integration testing

**Objective:** Ensure that different modules of the application work together as expected.

**Participants:** Integration team comprising of backend and frontend developers.

**Techniques:** Test RESTful API endpoints using Spring MVC Test and verify integration with Vue.js components**.**

**Results:** Still to be completed.

* + System Testing

**Objective:** Evaluate the system's compliance with the specified requirements.

**Participants:** System testing team(All team) with QA leader.

**Techniques:** Black-box testing, performance testing using JMeter, and security testing with OWASP ZAP.

**Results:** Still to be completed.

As the

* + Acceptance Testing

**Objective**: Confirm that the system is ready for use.

**Participants**: All group members.

**Techniques Used**: User scenario testing, beta testing.

**Results**: Still to be completed.

* + Regression Testing

**Objective**: Ensure that new code changes have not adversely affected existing functionalities.

**Participants**: QA leader and the team.

**Techniques Used**: Automated regression suite using Selenium for end-to-end testing.

**Results**: Still to be completed.

* + Tools Employed

A variety of tools were employed throughout the testing process:

* Unit Testing: JUnit, Jest.
* Integration/System Testing(Expected): Spring MVC Test, Postman.
* Performance/Security Testing(Expected): JMeter, OWASP ZAP.
* Regression Testing(Expected): Selenium WebDriver.
* Test Management(Expected): TestRail for test case management and tracking.
  + Conclusion

The testing phase will be rigorous and provide significant insights into the system's quality and readiness for launch. With the identified issues to be addressed, the TerrierMall Project is well-positioned to provide a stable, secure, and user-friendly experience for its intended audience.

# Manual Testing Report

In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.

Here is a sample template that can be used for each test case. For system tests or acceptance tests, you may also include some screenshots.

* Test case ID 1, name: Home page test (Qinchen Gu)
* New or old: New
* Test items: (what do you test ) Home page is correctly exist and can be access correctly
* Test priority (high/medium/low): High
* Dependencies (to other test case/requirement if any): Database, Frontend, Backend
* Preconditions: (if any): Every part of project codes related to the home page is working well and cooperating well.
* input data: Address of home page
* Test steps: Input the address of home page in browser and press enter button on computer
* Postconditions: Every part cooperating well and work together without bugs or errors
* Expected output: Home page is correctly displayed
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:
* Test case ID: 2, name: Login function test (Qinchen Gu)
* New or old: New
* Test items: (what do you test ): Login button and login functionality
* Test priority (high/medium/low): High
* Dependencies (to other test case/requirement if any): Database
* Preconditions: (if any) Database is correctly connected, data can be stored and extracted from db
* input data: Username and password
* Test steps: Press login button on the top right corner of page, input data and press login button
* Postconditions: User’s information is already stored in database
* Expected output: Directly jump to home page and the login button on the top right corner changed to your profile photo for entrance of user profile and a bell to notify notifications
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:
* Test case ID: 3, name: Register function test (Qinchen Gu)
* New or old: New
* Test items: (what do you test ): Register button and register functionality
* Test priority (high/medium/low): High
* Dependencies (to other test case/requirement if any): Database
* Preconditions: (if any) Database is correctly connected, data can be stored and extracted from db
* input data:Username, password and confirm password
* Test steps: Press login button on the top right corner of page, press register button, input data and press register button
* Postconditions: User’s information should be stored in database after register button is pressed
* Expected output: Directly jump to home page and the login button on the top right corner changed to your profile photo for entrance of user profile and a bell to notify notifications, also user’s information correctly displayed in the database
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:
* Test case ID 4, name: Forget password function (Phone) test (Qinchen Gu)
* New or old: New
* Test items: (what do you test ) Forget password button and forget password (phone) function
* Test priority (high/medium/low): Medium
* Dependencies (to other test case/requirement if any): Database
* Preconditions: (if any): Database is correctly connected, data can be stored and extracted from db
* input data: Phone number, verification code and new password
* Test steps: Press login button on the top right corner of page, press forget password button, input data and press save changes button
* Postconditions: User’s new information should be stored in database after save changes button is pressed, and user’s password should be changed to the new one
* Expected output: Directly jump to home page and the login button on the top right corner changed to your profile photo for entrance of user profile and a bell to notify notifications, also user’s information correctly displayed in the database
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:
* Test case ID 5, name: Forget password function (Email) test (Qinchen Gu)
* New or old: New
* Test items: (what do you test ) Forget password button and forget password (email) function
* Test priority (high/medium/low): Medium
* Dependencies (to other test case/requirement if any): Database
* Preconditions: (if any): Database is correctly connected, data can be stored and extracted from db
* input data: Email address, verification code and new password
* Test steps: Press login button on the top right corner of page, press forget password button, then press Email Reset Password button, input data and press save changes button
* Postconditions: User’s new information should be stored in database after save changes button is pressed, and user’s password should be changed to the new one
* Expected output: Directly jump to home page and the login button on the top right corner changed to your profile photo for entrance of user profile and a bell to notify notifications, also user’s information correctly displayed in the database
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:
* Test case ID 6, name: Search Function Test
* New or old: New
* Test items: Search bar functionality
* Test priority: High
* Dependencies: Database, Frontend
* Preconditions: Search functionality is integrated, and the database is populated with product data.
* Input data: A specific product name or keyword
* Test steps: Enter a product name or keyword in the search bar and press the search button.
* Postconditions: The search query is processed, and relevant products are displayed.
* Expected output: A list of products relevant to the search query is displayed.
* Actual output:
* Pass or Fail:
* Bug id/link:
* Additional notes: Testing includes verifying the accuracy of search results and the response time.
* Test case ID 7, name: Product Details Page Test
* New or old: New
* Test items: Product details display and functionality
* Test priority: High
* Dependencies: Database, Frontend
* Preconditions: Product details are correctly stored in the database.
* Input data: Click on a product
* Test steps: From the list of products, click on any product to view its details.
* Postconditions: Detailed information about the product is displayed.
* Expected output: Product details page with information like price, specifications, reviews, etc.
* Actual output:
* Pass or Fail:
* Bug id/link:
* Additional notes: Focus on the load time and the accuracy of the information displayed.
* Test Case ID 8: Featured Items on Homepage
* New or old: New
* Test items: Featuring items on the homepage
* Test priority: High
* Dependencies: Database, Frontend, Backend
* Preconditions: Feature to highlight items on the homepage is integrated.
* Input data: Selection of items to be featured
* Test steps:
* Choose an item to feature.
* Apply the feature to highlight it on the homepage.
* Check the homepage for the item's presence and prominence.
* Postconditions: The featured item should be prominently displayed on the homepage.
* Expected output: The homepage displays the featured item distinctly.
* Actual output:
* Pass or Fail:
* Bug id/link:
* Additional notes:
  + Test Case ID 9: User Profile Editing
  + New or old: New
  + Test items: Editing and saving changes to a user profile
  + Test priority: High
  + Dependencies: Database, Frontend, Backend, User Authentication System
  + Preconditions: User is registered and can log into their account. Profile editing functionality is integrated.
  + Input data: Updated user information (e.g., name, profile picture, contact details)
  + Test steps:
  + Log into the user account.
  + Navigate to the user profile section.
  + Enter new details or changes in the profile edit form (such as changing the profile picture, updating contact information, etc.).
  + Save the changes.
  + Reload or revisit the profile to ensure changes are retained.
  + Postconditions: The user’s profile should be updated with the new information.
  + Expected output: The user profile displays the updated information, reflecting the changes made.
  + Actual output:
  + Pass or Fail:
  + Bug id/link:
  + Additional notes:
* Test case ID 10 name: Category Page Functionality Test
* New or old: New
* Test items: Functionality of the product category pages
* Test priority: High
* Dependencies: Database, Frontend, Backend
* Preconditions: Category pages are set up with products correctly categorized in the database.
* Input data: Selection of a specific product category
* Test steps:
  + Navigate to the category selection on the home page or menu.
  + Select a specific category (e.g., Electronics, Clothing, etc.).
  + Browse through the products listed in that category.
  + Optionally apply filters (like price range, brand, etc.) and sort options (like popularity, new arrivals, etc.).
* Postconditions: The selected category page displays with products, filters, and sorting options functioning correctly.
* Expected output: Products relevant to the chosen category are displayed with accurate filter and sorting functions.
* Actual output:
* Pass or Fail:
* Bug id/link:

Test case ID 11 name: Label Functionality Test (Haoran Dai)

* New or old: New
* Test items: Functionality of the post labels, Create Labels
* Test priority: High
* Dependencies: Database, Frontend, Backend
* Preconditions: Only administrators can create labels, ordinary users cannot operate them.
* Input data: Click the “Create Label” button
* Test steps:
  + The create() method of LabelController submits a POST request and passes label Name and logo (label image) as parameters to the method, which are encapsulated in LabelDTO.
  + After receiving the request, the Create() method transmits the parameters to the Create () method of the Label Service. The method then calls the insert Selective() of the LabelMapper to insert a piece of data into the label table LabelDTO and returns the result.
* Postconditions: The front page makes corresponding jumps and prompts according to the returned results.
* Expected output: The label was successfully created.
* Actual output:
* Pass or Fail:
* Bug id/link:

# 

# Automated Testing Report

Describe briefly the automated testing you have done, including where the test code resides in your code repository, what test frameworks are used, and the screenshots or generated testing report.

1. Everyone should test their code before gathering, testing methods including but not limited to JUnit, and Jest. Also, everyone’s mind and hand will be another important part of the test, for example for the front end, members can test by opening pages in the browser and checking error codes in the browser console, or checking the error code provided by IDE.

2. Test codes will be located under the main branch under the program in the repository named “test” to make the developer and Q&A leader clear where test cases are and ready to do a double check at any time.

3. Tests should include as many test cases as possible, including but not limited to all buttons connected to the right function, all functions work well, and all databases work properly.

# 

# Testing Metrics

In this section, you shall report any metrics used for the evaluation, e.g. # of test cases, test coverage, defects rate, etc.

| Metric Name | Description |
| --- | --- |
| cost | Cost of person-hours used, personal |
| total cost | Cost of person-hours used, for whole group |
| Number of files | in the whole project, can show product complexity |
| Number of test cases | prepared for the product |
| Test case pass rate | Rate of test cases passed, can show the error rate of product |

# References

<https://zhuanlan.zhihu.com/p/79413784>

[Coding Standards and Guidelines - GeeksforGeeks](https://www.geeksforgeeks.org/coding-standards-and-guidelines/)

<https://spring.io/guides/gs/testing-web/>

<https://junit.org/junit5/>

<https://junit.org/junit5/docs/current/user-guide/#writing-tests>

# Glossary