**CS673 Software Engineering** 

**Team 3 - Study Buddy**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Bohan Lin | Design & Implementation Leader / QA Leader | *Bohan Lin* | 10/16/2025 |
| Ittoop Shinu Shibu | Configurations Management Leader | *Ittoop Shinu Shibu* | 10/16/2025 |
| Melissa Cron | Project Leader and Requirements Leader | *Melissa Cron* | 10/16/2025 |
| Junzhe Chen | Security Leader | *Junzhe Chen* | 10/16/2025 |
| Qiuting Zhao | Design & Implementation Leader / QA Leader | *Qiuting Zhao* | 10/16/2025 |
| Dexiao Zhang | Design & Implementation Leader / QA Leader | *Dexiao Zhang* | 10/16/2025 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| **1.1** | **Bohan Lin** | **10/6/2025** | **Backend test** |
| **1.2** | **Bohan Lin** | **10/16/2025** | **Metrics of backend test** |
| **1.3** | **Melissa Cron** | **10/16/2025** | **Manual Test Reports** |

[Testing Summary](#_sm5odwyvuk3j)

[Manuel Tests Reports](#_pqso2mbjyzx4)

[Automated Testing Reports](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Testing Summary

In this section, you will summarize what was tested, who is involved in testing, when to test, testing techniques used, and testing result. You may have the following tests

* + Unit Testing
  + Integration testing
  + System Testing
  + Acceptance Testing
  + Regression Testing

Testing is separated in backend and frontend. Backend tests are implemented by Bohan Lin.

# 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.

* + Screen Traversal
  + New or old: new
  + Test items: Can all screens be accessed through only links
  + Test priority: medium
  + Dependencies: no
  + Preconditions: no
  + input data: Click on the links one at a time
  + Test steps: Starting on the home screen, click through every link
  + Postconditions: Either every link has been tested or every screen has been reached
  + Expected output: Every screen has been reached
  + Actual output: Every screen has been reached
  + Pass or Fail: Pass
  + Bug id/link: N/A
  + Additional notes: This might be able to be automated
  + Clock Changes Screen
  + New or old: new
  + Test items: When the clock hits 0, it changes the state of the screen
  + Test priority: high
  + Dependencies: no
  + Preconditions: N/A
  + Input data: Click on the 3 second test button
  + Test steps: Go to the “select study time” page. Click the 3 second test button. Wait for the clock to run out
  + Postconditions:
  + Expected output: The screen should switch to read “You finished!”
  + Actual output: See expected output
  + Pass or Fail: Pass
  + Bug id/link: N/A
  + Additional notes: On the public version, 3 second test button needs to be removed

# 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 screen shots or generated testing report.

Jest and supertest are tools used in backend testing. A mix of unit tests, integration tests, acceptance tests, and system tests are applied. The unit tests focus on isolated middleware logic, such as JWT authentication and logging, without relying on the database or HTTP layer. The integration tests cover the interaction between models and the database. The acceptance tests test the full API workflow via HTTP request. The system tests simulate a complete user journey across multiple components, which would be helpful to validate the end-to-end functionality, state changes, and security under realistic scenarios.

All backend test files are in the *backend/\_\_tests\_\_****/*** *folder* in the develop branch. Manual tests will occur on the terminal using npm test command after the backend code is complete. When a pull request happens on Github, all the tests in the folder will be tested automatically.

| **Unit Tests** | | | |
| --- | --- | --- | --- |
| **Test File** | **Covered Functionality** | **Test Points** | **Improvement** |
| unit/auth.test.js | JWT verification logic | - No token -> 401  - Invalid token -> 401  - Valid token -> calls next | - Test expired tokens |
| unit/logger.test.js | Logging middleware | - Logs method/url  - Calls next |  |

| **Integration Tests** | | | |
| --- | --- | --- | --- |
| **Test File** | **Covered Functionality** | **Test Points** | **Improvement** |
| energyDecay.test.js | Buddy energy decay logic | - Create user & buddy  - Energy decay calculation  - Status update | - Test expired tokens |
| studySession.test.js | Study session model & database | - Add Session  - Get all sessions  - Get latest session |  |
| profile.test.js | Profile model + API | - Get profile  - Update profile  - Edge cases (no token / invalid data) |  |
| settings.test.js | Settings model + API | - Get settings  - Update settings  - Edge cases (no token / invalid data) |  |

| **Acceptance Tests (API Tests)** | | | |
| --- | --- | --- | --- |
| **Test File** | **Covered Functionality** | **Test Points** | **Improvement** |
| buddyAPI.test.js | Buddy API workflow | - Create buddy  - Get buddy  - Edge cases (duplicate creation / no token) | - Add integration with energy decay  - Verify token works when calling other protected endpoints |
| usersAPI.test.js | Login workflow | - Successful login  - Wrong password  - Non-existent user | - Test empty username/password  - Test expired or forged token calling protected APIs |
| profileAPI.test.js | Profile API | - Get profile  - Update profile  - No token / invalid data |  |
| settingsAPI.test.js | Settings API | - Get settings  - Update settings  - No token / invalid data |  |
| studyAPI.test.js | Study session API | - Add/Get/Validate sessions  - Invalid cases (illegal duration / no token) | - Test timestamp boundaries  - Test consecutive session additions |

# Testing Metrics

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

For backend tests in automated testing, 18 of total 18 tests passed including unit tests, integration tests, and acceptance tests.

# References

# Glossary