CS673 Software Engineering 

Team 1 - GetActive

Software Test Document

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Jin Hao Li | Design and Implementation Lead, Team Lead | JHL | 5/26/2025 |
| Arshdeep Dhillon | Security Lead  Design and Implementations  Team Lead | *AD* | 5/17/2025 |
| Shaohua Yue | Configuration Lead | *Shaohua Yue* | 5/26/2025 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**R**evision history

| Version | Author | Date | Change |
| --- | --- | --- | --- |
| 0.1 | Jin Hao Li | 5/26/2025 | Filled out Test Summary |
| 0.2 | Shaohua Yue | 5/26/2025 | Filled out part of frontend test summary |
| 1 | Arshdeep Dhillon | 5/26/2025 | Finalized for Iteration 1 |

[Testing Summary](#_heading=h.j9qa868oc4gb)

[Manuel Tests Reports](#_heading=h.b07hj7so1vs7)

[Automated Testing Reports](#_heading=h.udxhdtgov95)

[Testing Metrics](#_heading=h.8xt8kndm5zas)

[References](#_heading=h.381cbpgkvzcb)

[Glossary](#_heading=h.dm4bbqyaypoq)

# 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
    - JUnit tests for the following APIs:
      * GET /v1/activities
      * GET /v1/activity/{name}
      * POST /v1/login
      * POST /v1/register
    - JUnit tests for the following APIs:
      * JWTService
  + Integration testing
    - Under Investigation
  + System Testing
    - Under Investigation
  + Acceptance Testing
    - Under Investigation
  + Regression Testing
    - Under Investigation

# Manual Testing Report

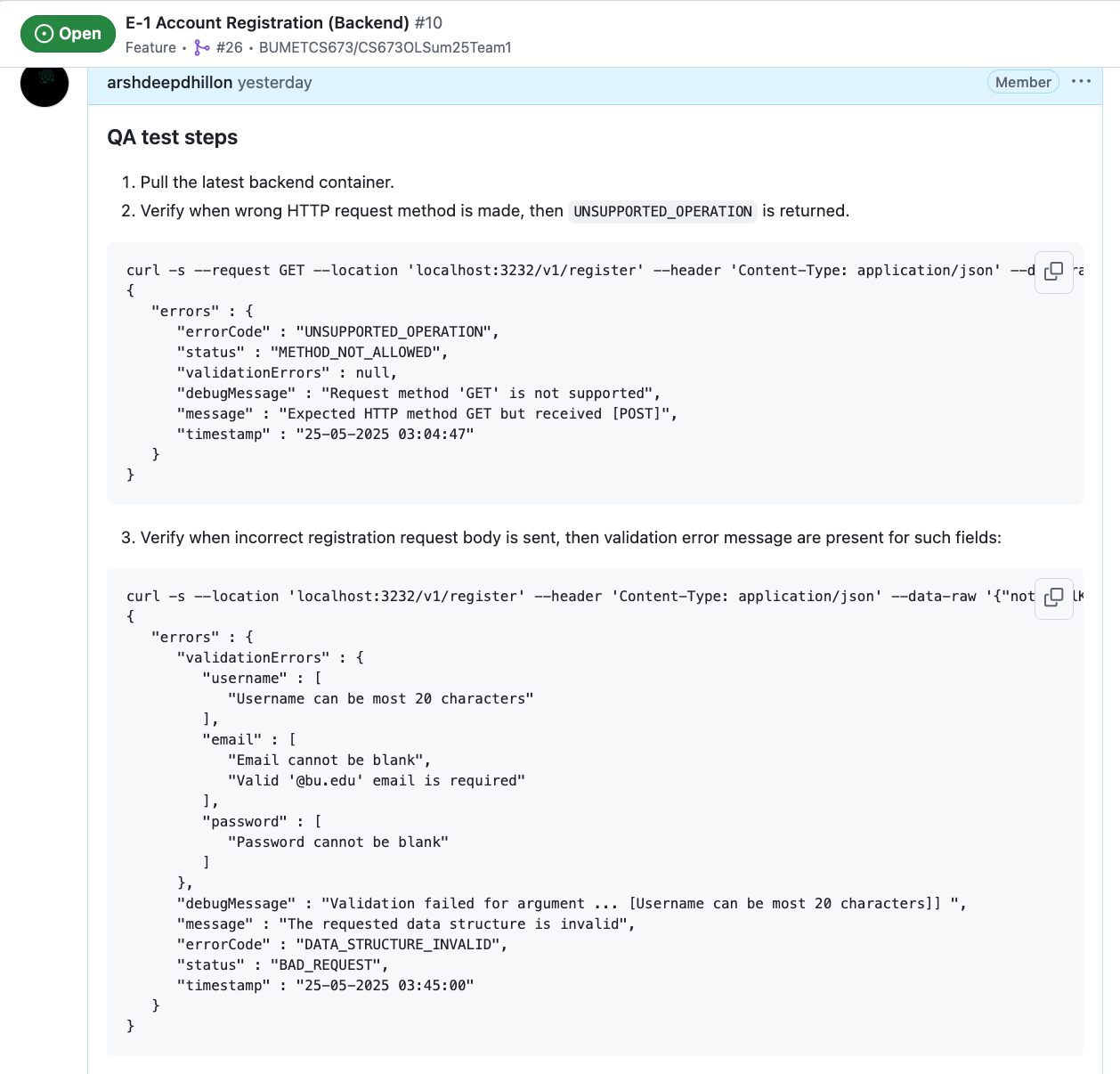
In this section, you will give a detailed description of each manual test case performed and the result. If this is a previous You shall list what are existing tests developed in the previous semester and what are new tests developed currently.

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, name
* New or old:
* Test items: (what do you test )
* Test priority (high/medium/low)
* Dependencies (to other test case/requirement if any):
* Preconditions: (if any)
* input data:
* Test steps:
* Postconditions:
* Expected output:
* Actual output:
* Pass or Fail:
* Bug id/link: (this should link to your github issue id)
* Additional notes:

(You can use an additional spreadsheet for this section as well)

A simple format for manual testing is used where each ticket will contain test steps for verifying the functionality of the changes and that the whole system continues to work as before:



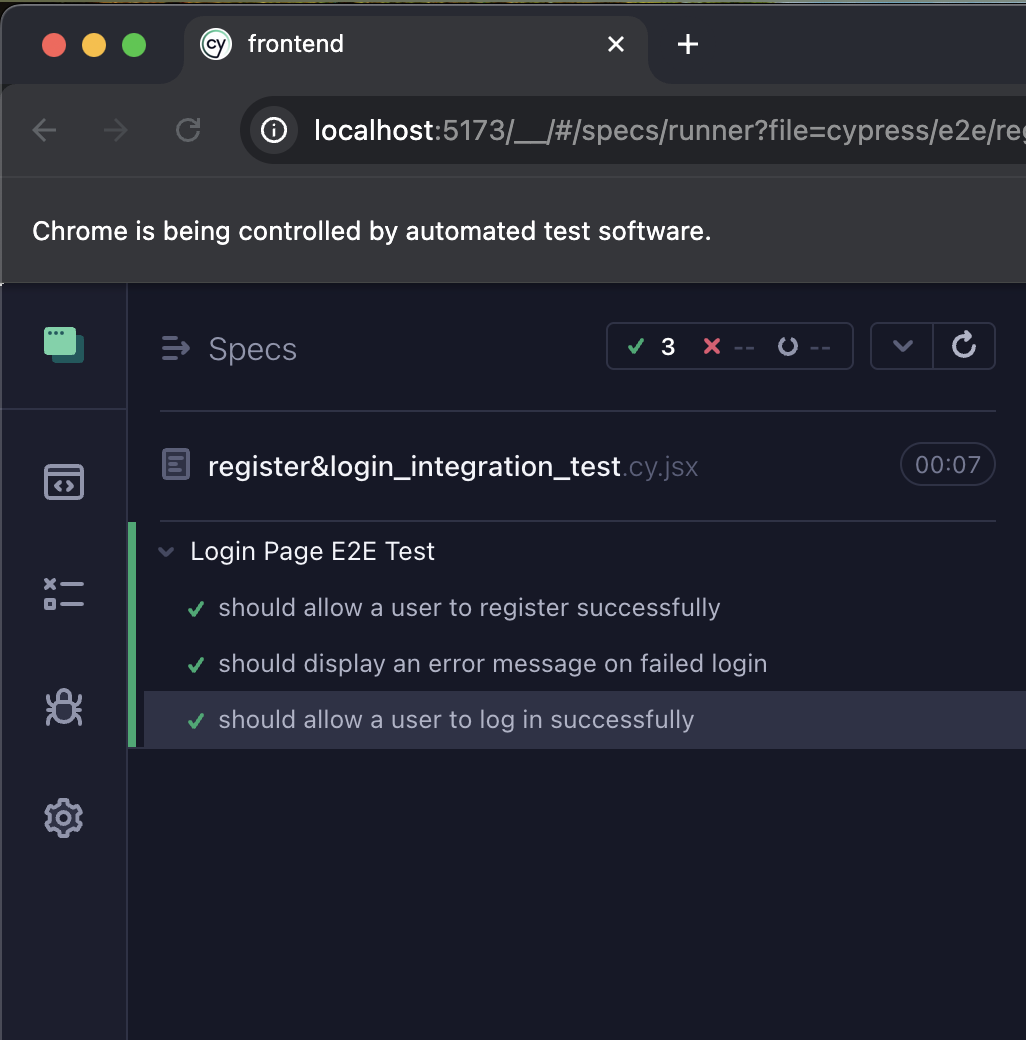
# Automated Testing Report

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.

Project includes automated testing for both the frontend and backend to ensure reliability and maintainability.

On the frontend:

1. we use Vitest with React Testing Library to perform unit and integration testing of React components. Key features such as the login form are thoroughly tested with simulated user input using userEvent, verifying successful and failed login flows, error handling, and navigation upon authentication. We mock dependencies like useAuth and useNavigate to isolate component behavior and confirm expected outcomes without relying on actual routing or authentication services. These tests are stored in the *tests* directory and integrated into the development workflow with coverage reports generated via *vitest --coverage*.
2. We also use Cypress for E2E testing. Now the test cases covered register and login process. It also could be executed automatically in CI workflow.



1. These tests reside under the *frontend/cypress* directory and *frontend/test* directory.

As for the backend:

1. automated testing is implemented using JUnit 5, Spring Boot Test, and MockMvc. The tests focus on validating REST API endpoints, particularly the */v1/* routes. Test cases ensure that invalid inputs, empty request bodies, and incorrect content types are handled gracefully by returning appropriate 4xx responses with detailed error messages. Dependencies like the email service and Gmail configuration are mocked using @MockBean to ensure isolation and avoid side effects during testing. These backend tests reside under the *src/test/java/com/bu/getactivecore* directory and are executed using standard Gradle command. Together, our automated test suites on both ends provide a robust foundation for early detection of issues.

# Testing Metrics

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

Under Investigation

# References

# Glossary