COSC 310 - 101

# M4: Testing Update Report

***Team Accord***

*Hoc Nguyen*
*Colin Lefter*
*Toby Nguyen*
*Bao Pham*
*Immanuel Wiessler*

## Table of Contents

# 1. Testing Strategy and Updates

## Current Test Approaches

- Similar to what has been outlined in M3, we are sticking with 3 main testing approaches: ad-hoc, unit, and black box testing.
    - For Ad-hoc testing: We use this approach mainly to check if the application is showing or behaving a certain way that we expect it to be. These tests are discretionary and done quickly.
    - For Unit testing: We plan and develop them for features that we aim to implement within a given sprint before the features are implemented themselves. In contrast to ad hoc tests, they are formally documented as test cases. For example, such tests would cover parsing JSON data from MongoDB queries and checking if the outputs are as expected. In addition, this may also involve checking if a query parameter always returns a unique object.
    - For Black-box testing: These tests are written mainly to target specific components and mock the process of receiving user input from an isolated frontend component followed by handling user input by REST API endpoints.

## Future Additional Test Approaches

- For future weeks closer to our deliverable application, we will implement three more testing approaches: Integration Testing, Acceptance Testing, and Usability Testing.
    - For Integration testing: This will involve writing tests that specifically verify the interaction between the different components in our software. These tests will help us ensure that the individual components that we have built are working when they are integrated into our application.
    - For Acceptance testing: We will perform these tests by having our peers, and TA act as potential users. They will use the application and verify if our developed software meets the business/project requirements of COSC 310.
    - For Usability testing: This type of test will be performed by both our development team and our TA to see if the application is easy to use, intuitive, and meets the targeted users' requirements interim of usability.

## Functionality Updates

- So far in our project, we have implemented many features together with their tests:
    - Registration functionality:

        - The constraint function (ensuring the users input their information in the correct format) works.
        - The register function is also working by posting the correct registered information to the database. Both of these functions pass the ad-hoc, and unit tests that we have implemented so far.
    - Login functionality:
        - The constraint function (ensuring the users input their information in the correct format) works.
        - The login function is also working by sending the input login details by the user and validating them with the data in our database. Both of these functions pass the ad-hoc, and unit tests that we have implemented so far.
    - Linking functionality in our landing page:
        - Ad-hoc and black-box testing have been implemented, and the functionality has passed all the tests.
    - Friend List-display functionality:
        - We have implemented the database with our Friend List. It is now working properly with the data in the database and passed the unit tests written for it. However, the data is of “user1” due to the lack of “State-validation functionality” that is still being worked on.
    - Chat history-display functionality:
        - We have implemented the database with our Chat History. It is now working properly with the data in the database and passed the unit tests written for it. However, the data is of “user1” due to the lack of “State-validation functionality” that is still being worked on.
    - Server List-display functionality:
        - We have implemented the database with our Server List. It is now working properly with the data in the database and passed the unit tests written for it. However, the data is of “user1” due to the lack of “State-validation functionality” that is still being worked on.
    - Live Messaging functionality:
        - So far we have implemented the functionality of live messaging between 2 users. However, it is still a work in progress because we are still developing the “state validation functionality”. Without this function, we have yet to determine the currently logged-in user and their associated information.

# 2. Automation Implementation

- One of the automation processes implemented in our project involves streamlining our Node.js workflow. Previously, executing several command lines and performing multiple downloads was necessary to prepare the project for coding. With the implementation of a new workflow, tasks

such as pulling requests, creating new branches, and downloading resources are now automated.

- As mentioned earlier, the automation scripts are stored in files located at .github/workflows/node.js.yml. This location choice aims to minimize merge conflicts and facilitate automation starting from branch creation using the main branch as a base.

```yaml
name: Node.js CI For Accord

on:
  push: {}
  pull_request: {}

jobs:
  build:

    os: [ubuntu-latest, windows-latest, macos-latest]

    strategy:
      matrix:
        node-version: [10.x, 14.x, 16.x, 18.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
    - uses: actions/checkout@v3
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v3
      with:
        node-version: ${{ matrix.node-version }}
        # Use a specific npm version for Node.js 10.x
        # Adjust the version according to your requirement
        # Check available npm versions: https://npmjs.com/package/npm/v/10.24.0
        npm: '10.24.0'
        cache: 'npm'
    - run: cd Application/Frontend # Navigating to the Frontend folder before anything else
    - run: npm install -g npm-check
    - run: npm --version
    - run: npm test:jest
```

  - **Overview of Automation Process**:
    - The automation process within our project encompasses several key components, primarily focusing on streamlining our Node.js workflow. Below is a breakdown of the various elements involved in this automation process:
  - **on**:
    - This section dictates the actions that trigger the automation process. Specifically, the automation is set to be initiated upon any push or pull request made to any branch within the repository.
  - **jobs**:
    - The automation process consists of multiple jobs, each serving distinct purposes. Key considerations include:

- **os**:
    - Specifies the permissible operating systems for executing the automation scripts. In our case, the automation is compatible with any Ubuntu, Windows, or MacOS operating system of the latest version.
- **strategy/matrix/node-version**:
    - Defines the compatibility of the automation process with different versions of Node.js, encompassing versions 10, 12, 14, or 16.
- **steps**:
    - This section outlines the sequential steps to be executed once all specified criteria are met. These steps include:
        - Preliminary Node.js Version Check and npm Package Verification:
            - Ensures compatibility with Node.js version 10.24.0, the latest version as of March 22nd. This verification is essential for maintaining consistency across packages, applications, and versions, mitigating potential conflicts previously encountered with package.lock files.
        - Directory Navigation to Frontend Application:
            - Navigates to the designated directory containing the frontend portion of the application, where node modules are typically located.
        - npm Package Installation:
            - Installs npm packages if the node_modules directory does not already exist, facilitating a fresh installation of packages when necessary.
        - npm Version Check:
            - Verifies the current version of npm installed within the environment.
        - Execution of Frontend Tests:
            - Runs tests embedded within the frontend using Jest, our primary testing library, to ensure the integrity and functionality of the application.
- By implementing these automation procedures, we aim to enhance efficiency, maintain consistency, and mitigate potential conflicts, thereby optimizing our software development workflow.

# 3. Summary

- For our team Accord, we believe that the project has made significant progress in implementing various functionalities such as registration, login, linking, friend list display, chat history display,

server list display, and live messaging. These functionalities have undergone ad-hoc, unit, and black-box testing to ensure their correctness and reliability. However, some features are still under development, notably the state-validation functionality, which impacts the accuracy of user data display.

- Process:
    - Our team has adopted a structured approach to testing, utilizing ad-hoc, unit, and black-box testing in the current phase, with plans to incorporate integration testing, acceptance testing, and usability testing in the future. The division of testing approaches aligns well with the development cycle, with ad-hoc testing for quick checks, unit testing for feature validation, and black-box testing for component verification. However, the team may need to ensure better synchronization between testing and development to address ongoing issues with state validation functionality.
- Quality Assurance:
    - To ensure the quality of the source code, the team has implemented various measures. Firstly, they have integrated unit testing into their development process, writing test cases for features before implementation. This helps catch bugs early and ensures that features meet specified requirements. Additionally, the team has employed ad-hoc testing for quick validation and black-box testing to simulate user interactions and verify component functionality. However, to further improve quality, the team should prioritize the completion of the state-validation functionality, which is crucial for accurate data display. Additionally, incorporating integration testing, acceptance testing, and usability testing in the future will provide comprehensive validation of the application's functionality and user experience. Overall, while the project has made good progress, there is room for refinement in the testing process and completion of critical functionalities for enhanced quality assurance.