FEBRUARY 8, 2024

VERIFICATION AND VALIDATION

GROUP 17

JIE ZHANG – ZHANJ265@MCMASTER.CA
ZIRUI HE – HEZ66@MCMASTER.CA
KAICHENG XU – XUK46@MCMASTER.CA
SHUREN XU – XUS83@MCMASTER.CA

Revision History

Version	Date	Notes
0	**Feb 8, 2024**	**Initial VnV document**

Project description:

Component Test plan: For every component of the system, provide one or both followings, based on the advice below:
1) Unit tests: A testing methodology that verifies desired behavior and lack of undesired behaviors which you identified in your design. Typically, this will be unit tests. Any software component needs testing (both frontend and backend).
2) Performance tests and metrics: A metric and measurement methodology to verify your claims in terms of performance. This is very critical for projects with a machine leaning component. The metric must be appropriate. For example, for an unbalanced dataset, accuracy is not a good metric. Note that performance is not only for ML systems. If you have a web-based system, measuring the latency is important.

Backend:

Authentication Module

Unit Tests:

- Test valid login with correct credentials.
- Test invalid login with incorrect credentials.
- Test account lockout after a predefined number of failed attempts.
- Test token generation and expiration mechanism.

Performance Tests and Metrics:

- Measure login response time under normal load.
- Measure token generation time.
- Test the maximum number of authentications the system can handle per second (throughput).
- Test system behavior under high concurrent login attempts.

Coverage-Based Testing:

- Ensure all branches in the authentication logic are tested, including success, failure, and token expiration scenarios.
- Achieve as close to 100% code coverage as possible with special attention to security-related code paths.

Fault Injection:

- Simulate database unavailability to test how the system responds to failed authentication requests.
- Inject network latency to simulate delayed responses from authentication databases or services.
- Introduce corrupted tokens to test system's ability to reject them.

Data Module

Unit Tests:

- Test Create, Read, Update, and Delete (CRUD) operations on user and recipe data.
- Test data validation logic to prevent SQL injection or other injection attacks.
- Test data retrieval speed and integrity.

Performance Tests and Metrics:

- Measure the response time for CRUD operations.
- Evaluate the transaction throughput for high-load scenarios.
- Assess database indexing efficiency.

Coverage-Based Testing:

- Ensure each CRUD operation is covered with tests for normal, boundary, and error conditions.
- Test all data validation paths for robustness against malformed input.

Fault Injection:

- Simulate faults in the data layer, such as read/write failures, to test the system's resilience.

- Inject delays or failures in database transactions to assess transactional integrity.

Content Inspection Module

Unit Testing

- Ensure administrator authentication correctness and safety
- Check input and output format
- Test if the module meets desired outcomes

Performance Tests and Metrics:

- Measure response time for CRUD operations
- Test robustness and performance with large load of data
- Evaluate latency for desired usage

Security Testing

- Identify potential vulnerabilities in authentication and transmission

Black-box Testing

- Simulate real scenario to test in the perspective of the users (in this case the admins)

Social module

Unit Testing

- Ensure user privileges for each user group are correctly set
- Check input and output format
- Test if the module meets desired outcomes

Performance Tests and Metrics:

- Measure response time for CRUD operations
- Test robustness and performance with large load of data
- Evaluate latency for desired usage

Compatibility Testing

- Test compatibility for mainstream platforms and common network conditions

Black-box Testing

- Simulate real scenario to test in the perspective of the users (in this case the admins)

Frontend:

Exploratory Testing

Ask potential users to explore the UI and uncover defects in this process.

Usability Testing

Ask potential users to explore the UI and see if they can intuitively understand the functions of buttons and find out links. We can collect opinions about the UI's unexpected behaviors in this process.

Specification-based Testing

We will go testing our front-end features against all the explicit stated specifications and check if they work well as intended.

Fuzz Testing

Feeding in random input with no constraints and see if the program crashes or sending back error message.

Partition Testing

Testing the recipe creation page by giving partitioned inputs, for example, all numbers, all special characters, etc. Testing the Login page by using Facebook accounts, Gmail accounts, and DishyDishes own accounts.

Boundary Testing

Provide no input, text without picture, long text without picture, long text with a lot of pictures to recipe creation page and see if there is any unexpected behavior.
Provide no input to Login Page and see if it will fail.
Provide no input and next time a lot of inputs in user page when editing preferences, name, or pictures and see if it will fail or preferences stay the same.