ReDI School of
Digital Integration

Unit & Integration Testing

Version 1.0

We use tech to connect human potential and opportunity with dignity & humility

Introduction to Testing

Why Testing?

1. Testing is a crucial practice in software development to ensure code quality, identify bugs, and prevent regressions.
2. There are different levels of testing, including unit testing and integration testing.
3. **Some Example Libraries for testing in Javascript (React/Nodejs)**:

Unit Testing

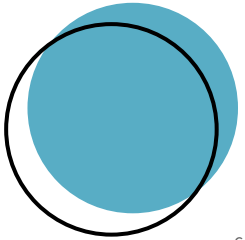
- Unit testing focuses on testing individual units or components of the code in isolation.
- It helps ensure that each unit of code behaves as expected.
- Mocks and stubs are often used to isolate the unit from external dependencies.
- **Benefits**:
    - Early Detection of Bugs: Unit tests catch bugs early in the development process, making them easier and cheaper to fix.
    - Code Quality: Unit testing promotes modular, maintainable, and testable code.
    - Documentation: Tests act as living documentation, explaining how components should function.

Writing a unit test (using Nodejs/React)

```
// Example: A function to add two numbers
function add(a, b) {
    return a + b;
}

// Unit Test using a testing library like Jest
test('add function should return the correct sum', () => {
    // Arrange
    const num1 = 5;
    const num2 = 10;

    // Act
    const result = add(num1, num2);

    // Assert
    expect(result).toBe(15);
});
```

Integration Testing

- Integration testing verifies the interactions between different units or components within the system.
- It ensures that integrated parts of the application work together as expected.
- **Benefits**:
    - **Catching Integration Issues**: Integration tests detect issues that may arise when components interact with each other.
    - **Full System Validation**: Integration tests provide confidence in the entire system's behavior.
    - **Higher Level of Assurance**: Integration tests complement unit tests, offering a higher level of assurance in the application's correctness.



Writing an integration test (using Nodejs/React)

```
// Example: An integration test using Jest and Supertest
const request = require('supertest');
const app = require('./app'); // Your Express app or React component to test

test('GET /api/users should return a list of users', async () => {
    const response = await request(app).get('/api/users');
    expect(response.status).toBe(200);
    expect(response.body).toHaveLength(3); // Assuming there are 3 users in the database
});
```

Conclusion

- Unit and integration testing are essential practices for ensuring code quality, preventing regressions, and building reliable applications.

- Adopting testing early in the development process leads to more robust and maintainable codebases.

References and more Resources

- https://dev.to/franciscomendes10866/testing-express-api-with-jest-and-supertest-3gf
- https://medium.com/@csalazar94/javascript-testing-made-easy-a-step-by-step-guide-with-jest-and-supertest-8e2a35f13506
- https://hackernoon.com/a-guide-on-writing-tests-in-full-stack-mern-web-application
-

Thanks a lot!

Contact

Muhammad Abdul Moeed
IT-Consultant at Lufthansa

LinkedIn