Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

IT322 – ADVANCED SYSTEMS INTEGRATION AND ARCHITECTURE

Learning Module No.5

Testing and Evaluation

Topic Outcomes

This module provides a comprehensive understanding of software testing, evaluation, and quality assurance (QA) techniques. Students will learn fundamental testing principles, QA methodologies, ISO standards for software evaluation, and the process of analyzing test results to ensure high-quality software products.

Learning Objectives

By the end of this module, students should be able to:

Understand the fundamentals of software testing and its importance in software development.
Apply various quality assurance techniques to improve software reliability.
Differentiate between different types of testing (unit, integration, system, acceptance).
Comprehend ISO standards and metrics used in software evaluation.
Conduct a structured evaluation process and analyze test results effectively.

Learning Content

Quality Assurance (QA) Techniques

What is Quality Assurance?

Quality − Quality is difficult to describe, however it may be summarized as "suitable for use or purpose." It is all about satisfying the demands and requirements of consumers in terms of product performance, design, dependability, longevity, and pricing.

Assurance − Assurance is simply a positive remark about a product or service that instills trust. It is the assurance that a product or service will operate properly. It ensures that the product will function properly and in accordance with the expectations or needs.

Quality Assurance in Software Testing

Quality Assurance in Software Testing is described as a method used to assure the quality of software products or services offered by a company to its consumers. Quality assurance is concerned with enhancing the software development process and making it more efficient and productive in accordance with the quality standards established for software products. Quality assurance testing is also known as QA testing.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

How to do Quality Assurance?

The PDCA cycle, often known as the Deming cycle, is a specified cycle in quality assurance technique. This cycle's stages are as follows

Plan − The organization should plan and define process-related targets, as well as identify the procedures necessary to achieve a high-quality final result.
Do − Process development and testing, as well as "do" adjustments to the procedures
Check − Process monitoring, process modification, and assessment of whether the processes achieve the set objectives.
Act − A Quality Assurance tester should carry out the activities required to enhance the procedures.

Quality Assurance vs. Quality Control

While QA and QC are closely linked, they are distinct concepts when it comes to managing both product or service quality. The main difference between QA and QC is that quality assurance focuses on planning to prevent risks that could affect quality during the production of services or goods, while quality control focuses on testing the quality of the output once the product or service has been delivered.

In project management, a quality assurance plan is a proactive approach to process improvement. It helps ensure the right processes, standards and guidelines are in place to meet quality objectives. Quality control procedures, on the other hand, are a more reactive process that consists of looking for defects once the product has already been made.

Quality Control Activities	Quality Assurance Activities
Walkthrough	Audit of Quality
Testing	Process Definition
Inspection	Recognition and selection of tools
Checkpoint review	Quality Standard and Process Training

The actions listed above are related to Quality Assurance and Control procedures for any product, not only software. In terms of software –

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

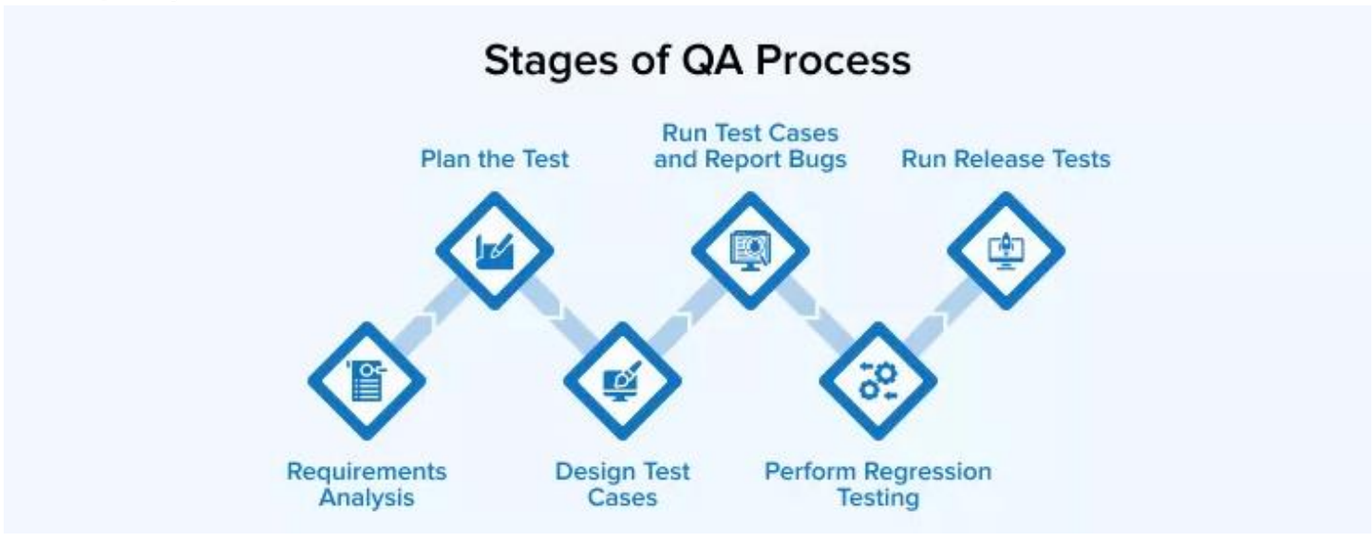
College of Informatics and Computing Sciences

Differences between SQA and Software Testing

The chart below compares and contrasts SQA with Software Testing

SQA	Software Testing
Software Quality Assurance refers to the engineering procedure that assures the quality of software.	Before a product goes online, it is subjected to software testing to ensure that it is free of flaws.
Activities associated with the implementation of processes, methods, and regulations are included. An Audit Training is one example.	Involves activities pertaining to product verification, such as Review Testing.
Process-oriented	Product-centered
Preventive measure	Corrective action
Prevention strategies	Reactionary measure
The scope of SQA was applied to all products that the organization would develop.	The scope of Software Testing is specific to the product being tested.

Quality Assurance Process



1. Analyze the Requirements

The QAs must understand and analyze the project requirements, including functional and non-functional aspects. Fixing a bug at the early stages of development will cost less when compared to fixing a bug at the testing/production stage. That is why QAs must be involved in the initial stages of requirement analysis so they understand the requirements clearly and design the tests accordingly.

2. Plan the Tests

Now, the information gathered in the requirements analysis phase will be used for test planning. The test plan includes the testing strategy, scope, project budget, and deadlines. Also, it should include information about the required types and levels of testing and the testing tools. And when the test plan is ready, the test manager will allocate responsibilities to individual testers.

3. Design the Tests

Now, the QA teams create test cases and checklists based on the software requirements. Each test case includes conditions, data, and steps to validate functionality. Testers compare actual results with expected results to ensure accuracy.

If automation is part of the test scope, this is the stage where automation testing scenarios are created. Also, the testing team should prepare the staging environment for test execution in this stage. The staging environment should closely resemble the production environment, including hardware, software, network configurations, databases, and system settings. It helps ensure accurate testing conditions.

4. Execute Tests and Report Defects

Tests begin with developers conducting unit tests at the individual code component level. Following that, the QA testing team carries out tests at the API and UI levels. Manual tests are conducted based on pre-designed test cases. All identified bugs will be recorded in a defect-tracking system for efficient management. Moreover, test automation engineers can utilize automated test tools like Test sigma to execute test scripts and generate detailed test reports.

5. Run Re-Tests and Regression Tests

After bugs have been identified, reported, and resolved, QA will again re-test the functions to ensure thorough validation and ensure they didn't miss any user scenario. Additionally, they perform regression tests to confirm that the fixes haven't caused any negative impact on the existing functionalities.

6. Run Release Tests

After developers notify about a release, the QA team identifies the affected functionalities. They create new test suites to cover the changes. The team also conducts smoke tests to check stability. If the tests pass, they run the modified test suites and generate a report.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

QA Processing Approaches

Manual Testing

Manual testing involves human testers executing test cases and generating test reports. It is suitable for small test projects and can identify issues that may be missed in automated tests. Manual testing does not require advanced scripting knowledge and is relatively easy to perform. However, it can be time-consuming for large projects and is prone to human errors.

Automated Testing

Automation testing, on the other hand, is performed using automation testing tools. Here, test engineers create and write test scripts to test the application. It simulates user experiences, and test scripts can be reused multiple times. Automation testing is beneficial for large projects but lacks the human touch and may overlook errors beyond its programmed scope. That is why exploratory testing is preferred, in which testers will think outside of the box to think of scenarios that might have been missed during automation testing. Combining manual and automated testing yields good results. QA professionals need to evaluate and decide which approach to use for each project based on its specific requirements and considerations. But automation offers a ton of benefits when compared to manual testing, like reusable test scripts, easy parallel testing, less test execution time, fast testing, and more.

Benefits of Implementing a QA Process

On the other hand, implementing a well-organized QA process can significantly benefit your business. It enhances the quality of your product, leading to increased customer satisfaction and loyalty. Additionally:

1. Your team can dedicate more time to enhancing the product and introducing new features.
2. Satisfied users are less likely to raise complaints, reducing the workload on the customer support team.
3. High product quality reduces concerns about potential failures.
4. A proper QA process helps minimize project costs since preventing defects is more cost-effective than fixing them later on.

College of Informatics and Computing Sciences

Common Problems or Challenges in the QA Process

Some of the common issues are:

1. Insufficient Time: If there is not enough time allocated for testing, then this can result in defects going undetected.
2. Lacking Resources: When there are limited skilled QA professionals or tools for testing, this can impede the process.
3. Scope Creep: A change in requirements or functionalities during their development can affect the effort and timelines of testing.
4. Complex Systems: It is a challenge to test complex systems that have multiple components and dependencies.
5. Manual Testing Limitations: It is time-consuming and error-prone to rely only on manual testing.

Quality Assurance Certifications

There are several certifications available in the industry to ensure that Organizations follow Standards Quality Processes. Customers make this as qualifying criteria while selecting a software vendor.

ISO 9000

This standard was first established in 1987, and it is related to Quality Management Systems. This helps the organization ensure quality to their customers and other stakeholders. An organization who wishes to be certified as ISO 9000 is audited based on their functions, products, services and their processes. The main objective is to review and verify whether the organization is following the process as expected and check whether existing processes need improvement.

This certification helps:

- Increase the profit of the organization
- Improves Domestic and International trade
- Reduces waste and increase the productivity of the employees
- Provide Excellent customer satisfaction

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

CMMI level

The **Capability Maturity Model Integrated (CMMI)** is a process improvement approach developed specially for software process improvement. It is based on the process maturity framework and used as a general aid in business processes in the Software Industry. This model is highly regarded and widely used in Software Development Organizations.

CMMI has 5 levels. An organization is certified at CMMI level 1 to 5 based on the maturity of their Quality Assurance Mechanisms.

Level 1 – **Initial:** In this stage the quality environment is unstable. Simply, no processes have been followed or documented

Level 2 – **Repeatable:** Some processes are followed which are repeatable. This level ensures processes are followed at the project level.

Level 3 – **Defined:** Set of processes are defined and documented at the organizational level. Those defined processes are subject to some degree of improvement.

Level 4 – **Managed:** This level uses process metrics and effectively controls the processes that are followed.

Level 5 – **Optimizing:** This level focuses on the continuous improvements of the processes through learning & innovation.

Test Maturity Model (TMM)

This model assesses the maturity of processes in a Testing Environment. Even this model has 5 levels, defined below-

Level 1 – **Initial**: There is no quality standard followed for testing processes and only ad-hoc methods are used at this level

Level 2 – **Definition:** Defined process. Preparation of test strategy, plans, test cases are done.

Level 3 – **Integration:** Testing is carried out throughout the software development lifecycle (SDLC) – which is nothing but integration with the development activities, E.g., V- Model.

Level 4 – **Management and Measurement:** Review of requirements and designs takes place at this level and criteria has been set for each level of testing

Level 5 – **Optimization:** Many preventive techniques are used for testing processes, and tool support (Automation) is used to improve the testing standards and processes.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Software Testing Fundamentals

Definition and Importance of Software Testing

Software testing is essential to ensure that applications work correctly, perform well, and are free from critical bugs before reaching users. It helps identify issues early, improves software quality, and enhances user experience.

Types of Software Testing

Software testing techniques are methods used to design and execute tests to evaluate software applications. The following are common testing techniques:

1. **Manual testing** – Involves manual inspection and testing of the software by a human tester.
2. **Automated testing** – Involves using software tools to automate the testing process.
3. **Functional testing** – Tests the functional requirements of the software to ensure they are met.
4. **Non-functional testing** – Tests non-functional requirements such as performance, security, and usability.
5. **Unit testing** – Tests individual units or components of the software to ensure they are functioning as intended.
6. **Integration testing** – Tests the integration of different components of the software to ensure they work together as a system.
7. **System testing** – Tests the complete software system to ensure it meets the specified requirements.
8. **Acceptance testing** – Tests the software to ensure it meets the customer's or end-user's expectations.
9. **Regression testing** – Tests the software after changes or modifications have been made to ensure the changes have not introduced new defects.
10. **Performance testing** – Tests the software to determine its performance characteristics such as speed, scalability, and stability.
11. **Security testing** – Tests the software to identify vulnerabilities and ensure it meets security requirements.
12. **Exploratory testing** – A type of testing where the tester actively explores the software to find defects, without following a specific test plan.
13. **Boundary value testing** – Tests the software at the boundaries of input values to identify any defects.
14. **Usability testing** – Tests the software to evaluate its user-friendliness and ease of use.
15. **User acceptance testing (UAT)** – Tests the software to determine if it meets the end-user's needs and expectations.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Testing Techniques

What is Black-Box Testing?

Black-box testing is a software testing method where the tester evaluates the functionality of an application without any knowledge of its internal code structure, implementation details, or internal paths1. This approach focuses on the inputs and outputs of the software application, ensuring that it meets the specified requirements and behaves as expected.

What is White Box Testing?

White box testing is a Software Testing Technique that involves testing the internal structure and workings of a Software Application. The tester has access to source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

White box testing is also known as Structural Testing or Code-based Testing, and it is used to test the software's internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.
Gray Box Testing

What is Gray Box Testing?

Gray Box Testing is a software testing technique that is a combination of the Black Box Testing technique and the White Box Testing technique.

1. In the Black Box Testing technique, the tester is unaware of the internal structure of the item being tested and in White Box Testing the internal structure is known to the tester.
2. The internal structure is partially known in Gray Box Testing.
3. This includes access to internal data structures and algorithms to design the test cases.
4. Gray Box Testing is named so because the software program is like a semitransparent or gray box inside which the tester can partially see.
5. It commonly focuses on context-specific errors related to web systems.
6. It is based on requirement test case generation because it has all the conditions presented before the program is tested.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

ISO Evaluation Standards and Metrics

What is ISO?

- International Organization for Standardization
- Promotes global standardization across various industries.

Key ISO Standards Relevant to Evaluation:

ISO 9001 – Quality Management Systems
ISO 9001 sets out the criteria for a quality management system and is based on a number of quality management principles including a strong customer focus, the involvement of top management, a process approach, and continual improvement.

Key Aspects:

- Focuses on meeting customer expectations and delivering customer satisfaction.
- Require organizations to document procedures and processes that affect quality.
- Promotes a process-based approach to managing business operations.
- Emphasizes continual improvement through internal audits and performance evaluations.

ISO/IEC 25010 – System and Software Quality Models
Purpose:
This standard defines a quality model for evaluating the quality of software products. It provides a framework for assessing both product quality and quality in use.

Key Aspects:

Divides software quality into two main categories:

1. Product Quality (e.g., functionality, reliability, usability, efficiency, maintainability, portability, security, compatibility)
2. Quality in Use (e.g., effectiveness, efficiency, satisfaction, freedom from risk, and context coverage

ISO 14001 – Environmental Management Systems (EMS)

Purpose:
ISO 14001 provides a framework for an effective environmental management system. It is used by organizations to enhance environmental performance.

Key Aspects:

- Focuses on managing environmental responsibilities in a systematic manner.
- Help organizations minimize negative impacts on the environment.
- Encourages compliance with applicable laws, regulations, and other environmental requirements.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

ISO/IEC 27001 – Information Security Management Systems (ISMS)

Purpose:
ISO/IEC 27001 specifies the requirements for establishing, implementing, maintaining, and continually improving an information security management system.

Key Aspects:

- Helps protect information assets from threats such as cyberattacks, data breaches, and insider threats.
- Covers security policies, access control, cryptography, physical and environmental security, and incident management.
- Supports risk assessment and treatment.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Evaluation Process and Analysis

Evaluation is a systematic activity that verifies whether a system, software, or process meets the expected requirements and performs effectively. It involves multiple steps from defining what to test, executing the tests, and analyzing results to improve quality.

Steps in the Evaluation Process

1. **Requirement Analysis** – Define test criteria based on requirements.

   This is the foundation of the evaluation process. It involves:

   - Gathering and analyzing system or software requirements.
   - Identifying **key quality attributes** (e.g., functionality, usability, reliability).
   - Defining **measurable evaluation criteria** (testable objectives) based on user needs, business goals, and regulatory requirements.

   Example: For a hospital management system, requirements may include 24/7 uptime, secure patient data access, and fast report generation. These become part of the testing benchmarks.

2. **Test Planning** – Develop test strategy, scope, and resources.

   Test planning defines the **scope, objectives, resources, and schedule** of testing activities.

   - Identify **types of testing**: functional, non-functional, security, performance.
   - Determine **entry and exit criteria**.
   - Allocate resources: test engineers, tools, environments.

   Example: Deciding to use JMeter for performance testing and allocating two QA engineers for a two-week testing window.

3. **Test Case Design** – Create test scenarios and data.

   Based on the requirements, detailed test cases are created:

   - Define **test inputs, execution steps, and expected results**.
   - Include **positive and negative test scenarios**.
   - Prepare **test data** that reflects real-world usage.

   Example: A login test case includes valid credentials (expected: login success) and invalid credentials (expected: error message).

4. **Test Execution** – Run tests and log defects.

   This is where the planned tests are run:

   - Execute test cases in a real or simulated environment.
   - Log **actual outcomes** and any discrepancies.
   - Report and track **defects or anomalies** in a bug tracking tool (e.g., JIRA, Bugzilla).

Example: A payment system crashes when invalid card details are entered — the defect is logged as "High priority."

5. **Result Analysis** – Compare actual vs. expected outcomes.

   Here, we compare actual outcomes with expected results:

   - Identify **deviations** or unexpected behaviors.
   - Assess test coverage and quality risk.
   - Use **analytics tools** or dashboards to spot trends and issues.

Example: 85% of test cases passed, but 4 out of 5 security tests failed — this indicates a serious risk.

6. **Reporting & Improvement** – Document findings and refine processes.

   Final stage where results are documented and shared with stakeholders:

   - Create a **Test Summary Report** with metrics like pass/fail rate, defect density, and execution status.
   - Conduct **post-mortem reviews** to identify process gaps.
   - Provide **recommendations** for future releases or development practices.

   Example: Recommend using better requirements documentation and automated regression testing to reduce defects.

4. Analyzing Test Results

After test execution, test results must be thoroughly analyzed to ensure reliability and guide future improvements.

Defect Classification:

Defects are prioritized based on **impact and severity**:

- **Critical**: System crash, data loss, major functionality failure.
- **Major**: Significant deviation but system still functions.
- **Minor**: Cosmetic or UI issues, spelling errors.

Example: A login page crash = Critical, Misaligned button = Minor.

Root Cause Analysis (RCA):

A structured investigation to find **why** a defect occurred.

- RCA techniques: *5 Whys, Fishbone Diagram (Ishikawa), Fault Tree Analysis.*
- Root causes may include:
    - Poor or ambiguous requirements.
    - Incomplete unit testing.
    - Miscommunication between teams.
    - Coding or configuration errors.

Example: RCA of a failed checkout process reveals missing API token handling logic.

Test Summary Report:

A comprehensive document summarizing:

- **Test case execution statistics**: Number of cases passed/failed.
- **Defect trends**: High-priority vs. low-priority defects, module-wise defect distribution.
- **Test coverage**: Which parts of the system were tested vs. not tested.
- **Recommendations**: Areas needing improvement, training needs, process enhancements.

Example: Summary report suggests adding boundary testing in future cycles due to frequent edge-case failures.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

REST API Testing

1.What is an API?

An **API (Application Programming Interface**
You (the client) give an **order** (request).

- The waiter (API) takes it to the kitchen (server).
- The kitchen prepares the food (processes data).
- The waiter brings back your meal (response).

Real-World Example

When you check the weather on your phone:

- The app **requests** data from a weather API.
- The API **responds** with temperature, humidity, etc.

2. What is REST?

- REST (**REpresentational State Transfer**) is a set of rules for building APIs.

Key Features of REST APIs

- Uses **HTTP methods** (GET, POST, PUT, DELETE).
- Returns data in **JSON** or **XML** format.
- **Stateless** (each request is independent).

Common HTTP Methods

Method	What It Does	Example
GET	Retrieve data	GET /users → Returns all users
POST	Create data	POST /users → Adds a new user
PUT	Update data	PUT /users/1 → Edits user #1
DELETE	Remove data	DELETE /users/1 → Deletes user #1

3. What is API Testing?

API testing checks if:

✅ The API **works correctly** (returns the right data).
✅ It **handles errors** (e.g., wrong input).
✅ It's **fast and reliable** under load.
✅ It's **secure** (no hackers can break in).

Analogy: Testing a Vending Machine

- Press "A1" → You get chips (**correct response**).
- Press "Z9" (invalid) → Shows "Invalid selection" (**error handling**).
- 100 people use it at once → Doesn't crash (**performance**).
- Nobody can hack it to get free snacks (**security**).

4. What You Need to Start Testing APIs

4.1 Basic Knowledge

- **How the web works** (clients, servers, HTTP).
- **JSON/XML** (data formats APIs use).
- **HTTP status codes** (200=OK, 404=Not Found, etc.).

4.2 Tools for Beginners

Tool	Purpose
Postman	Send API requests manually
cURL	Send requests via command line
Swagger/OpenAPI	API documentation + testing

4.3 Sample API for Practice

Try these free fake APIs:

- JSONPlaceholder (Todo lists, users)
- Reqres (User registration)

5. Your First API Test (Step-by-Step)

Test Case: Get a List of Users

1. **Open Postman** (download from postman.com).
2. **Enter the API URL**:

   GET https://jsonplaceholder.typicode.com/users

3. **Click "Send"**.

4. **Check the Response**:
    1. **Status**: 200 OK (success!).
    2. **Body**: List of users in JSON.

Test Case: Error Handling (Invalid Request)

1. **Change the URL to an invalid one**:

   GET https://jsonplaceholder.typicode.com/invalid

2. **Click "Send"**.
3. **Check the Response**:
    - **Status**: 404 Not Found (correct error!).

6. Key Takeaways

- APIs let apps talk to each other (**like waiters in a restaurant**).
- REST APIs use **HTTP methods (GET, POST, PUT, DELETE)**.
- API testing checks **correctness, speed, errors, and security**.
- Start with **Postman** and free APIs like JSONPlaceholder.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Building a REST API with Node.js and MySQL

Lesson Overview:
In this lesson, students will:

- Understand what a REST API is.
- Learn how to set up a Node.js project.
- Connect Node.js to a MySQL database.
- Create CRUD endpoints (Create, Read, Update, Delete).
- Test the API using Postman.

Tools & Technologies:

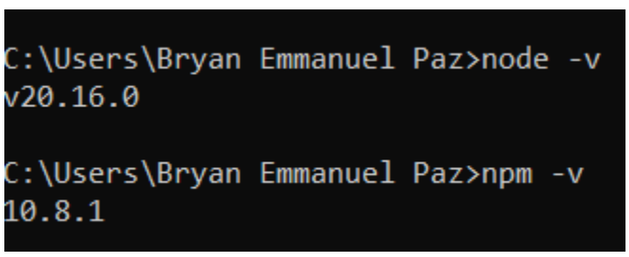
- Node.js
- Express.js
- MySQL
- MySQL Workbench or phpMyAdmin (GUI)
- Postman (for testing API)
- dotenv (for environment variables)
- nodemon (for auto-reloading server)

Installation & Setup

1. Install Node.js
Download from: https://nodejs.org/

Verify: After installing, open Terminal or CMD

➡ node -v
➡ npm -v



2. Install MySQL
Download from: https://dev.mysql.com/downloads/mysql/
Note: If you have xampp just start the mysql
Create a database (example: school_db)

3. **Initialize Project**

➡️ mkdir rest-api-node-mysql
➡️ cd rest-api-node-mysql
Then, initialize your project:
➡️ npm init -y
This creates a package.json file to manage dependencies.

4. Install Dependencies

- express: Handles routes and requests.
- mysql2: Connects Node.js to MySQL.
- dotenv: Loads .env file for secure credentials.
- nodemon: Automatically restarts server during development.

➡️ npm install express mysql2 dotenv
➡️ npm install --save-dev nodemon

Add this to package.json scripts:

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

Project Structure

```
rest-api-node-mysql/
├── config/
│   └── db.js
├── controllers/
│   └── studentController.js
├── routes/
│   └── studentRoutes.js
├── .env
├── index.js
└── package.json
```

College of Informatics and Computing Sciences

5. Create .env File insert the environment variable

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=yourpassword
DB_NAME=school_db
PORT=5000
```

Key Concepts Explained:

- dotenv Package:
  This package loads environment variables from a file named .env (which you need to create) so that sensitive information like the database password isn't hard-coded in your source files.

- Environment Variables:
  Think of them as configuration settings you can change without altering the code. In your .env file, you'll have lines like:

MySQL Connection:
The connection is established using the configuration provided. If something is misconfigured (like the wrong password), the application will throw an error.

6. config/db.js (MySQL Connection)

```
const mysql = require('mysql2');

require('dotenv').config();


const db = mysql.createConnection({
host:process.env.DB_HOST,user:
process.env.DB_USER,
password: process.env.DB_PASSWORD,
database: process.env.DB_NAME

});

db.connect(err => {
  if (err) throw err;
  console.log("MySQL Connected!");
});

module.exports = db;
```

7.**index.js (Main Server File)**

```
const express = require('express');
const dotenv = require('dotenv');
const studentRoutes = require('./routes/studentRoutes');

dotenv.config();
const app = express();

app.use(express.json());
app.use('/api/students', studentRoutes);

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
});
```

8. Routes/studentRoutes.js

```
const express = require('express');
const router = express.Router();
const {
  getStudents,
  getStudent,
  createStudent,
  updateStudent,
  deleteStudent
} = require('../controllers/studentController');

router.get('/', getStudents);
router.get('/:id', getStudent);
router.post('/', createStudent);
router.put('/:id', updateStudent);
router.delete('/:id', deleteStudent);

module.exports = router;
```

9. Controllers/studentController.js

```
const db = require('../config/db');

//Function 1: Get All Students

exports.getStudents = (req, res) => {
  db.query('SELECT * FROM students', (err, results) => {
    if (err) throw err;
    res.json(results);
  });
};


//GET /api/students
//This runs the SQL: SELECT * FROM students to get all students from the table.
//The result (list of students) is returned as JSON.


//Function 2: GetOneStudentbyID

exports.getStudent = (req, res) => {
  db.query('SELECT * FROM students WHERE id = ?', [req.params.id], (err, result) => {
    if (err) throw err;
    res.json(result[0]);
  });
};

// GET /api/students/:id
// Gets the id from the URL like /api/students/1
// SQL query: SELECT * FROM students WHERE id = ?
//? is replaced by the value in req.params.id
//Returns just one student (result[0]).
```

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph
College of Informatics and Computing Sciences

```
//Function 3: Create a New Student

exports.createStudent = (req, res) => {
  const { name, course } = req.body;
  db.query('INSERT INTO students (name, course) VALUES (?, ?)', [name, course], (err, result) => {
    if (err) throw err;
    res.json({ message: "Student added!" });
  });
};

// POST /api/students
// Gets data from the request body (name and course)
// SQL: INSERT INTO students (name, course) VALUES (?, ?)
// ? placeholders are replaced by [name, course]
// Sends a success message when inserted.


//Function 4: UpdateaStudent

exports.updateStudent = (req, res) => {
  const { name, course } = req.body;
  db.query('UPDATE students SET name = ?, course = ? WHERE id = ?', [name, course, req.params.id],
 (err, result) => {
    if (err) throw err;
    res.json({ message: "Student updated!" });
  });
};

//PUT /api/students/:id
//Updates the student with the given ID
//Data comes from request body (name, course)
//SQL: UPDATE students SET name = ?, course = ? WHERE id = ?
/ req.params.id targets the specific student to update.


//Function 5: Delete a Student

exports.deleteStudent = (req, res) => {
  db.query('DELETE FROM students WHERE id = ?', [req.params.id], (err, result) => {
    if (err) throw err;
    res.json({ message: "Student deleted!" });
  });
};


//DELETE /api/students/:id
//Deletes the student with the given id
//SQL: DELETE FROM students WHERE id = ?
//After deletion, it responds with a confirmation message.
```

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Endpoints Summary

Method	Endpoint	Description
GET	/api/students	Get all students
GET	/api/students/:id	Get single student
POST	/api/students	Create a new student
PUT	/api/students/:id	Update student by ID
DELETE	/api/students/:id	Delete student by ID

Notes

Concept	Explanation
req.params	Gets URL data like /api/students/1 (id = 1)
req.body	Gets data from user input (used in POST/PUT)
res.json()	Sends JSON data back to the user
throw err	Stops code and shows error in terminal if something breaks

How to Run and Test

Open terminal same directory with the project

npm run dev

You'll see:

```
[nodemon] starting `node index.js`
Server running at http://localhost:5000
MySQL Connected!
```

MySQL Connected!
Server running at http://localhost:5000

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Lipa Campus
A. Tanco Drive, Brgy. Marawoy, Lipa, Batangas, Philippines 4217
Tel Nos.: (+62 43) 980-0385; 980-0387; 980-0391 to 94 loc. 3130
E-mail Address: cics.lipa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

Use Postman to Test API:

1.GET all students
GET http://localhost:5000/api/students

2. GET one student
GET http://localhost:5000/api/students/1

3.POST a student
POST http://localhost:5000/api/students
Body > Raw > JSON:

```
{
  "name": "Async Xap ",
  "course": "BSCS"
}
```

4.PUT (update) a student
PUT http://localhost:5000/api/students/1
Body > Raw > JSON:

```
{
  "name": "BryL Zap",
  "course": "IT"
}
```

DELETE a student
DELETE http://localhost:5000/api/students/1

Hi everyone!

If you encounter any issues or difficulties while working on your project, feel free to clone this template repository to help you out:

https://github.com/bryanemmanuel/rest-api-node-mysql_temp.git

This repo can serve as a guide or starting point. Don't hesitate to explore and learn from it!