**Experiment 9**

**On the basis of required validations map the different test cases to handle all possible critical cases which may arise during the life cycle of the software.**

To map different test cases for handling critical cases in the Faculty Contribution Management system, we need to identify potential failure points and scenarios where the system might behave unexpectedly. Here are some critical cases to consider along with corresponding test cases:

1. **Input Validation:**

* Test case: Verify that all input fields (e.g., textboxes, dropdowns) have proper validation for data type, length, and format.
* Test case: Test for boundary values, including maximum and minimum allowed values for input fields.
* Test case: Test for handling of special characters and invalid input formats.

1. **Authentication and Authorization:**

* Test case: Verify that only authorized users can access sensitive functionalities.
* Test case: Test different user roles (e.g., admin, faculty) and ensure that each role has appropriate access permissions.
* Test case: Verify that unauthorized access attempts are appropriately logged and denied.

1. **Data Integrity:**

* Test case: Verify that data entered by users is stored accurately in the database.
* Test case: Test for data consistency across different modules (e.g., contributions linked to the correct faculty member).
* Test case: Test data encryption and decryption mechanisms to ensure data security.

1. **Concurrency and Load Handling:**

* Test case: Simulate multiple users accessing and updating data simultaneously to ensure system stability under load.
* Test case: Verify that the system can handle a large number of concurrent requests without performance degradation.
* Test case: Test for race conditions and ensure proper synchronization mechanisms are in place.

1. **Error Handling and Recovery:**

* Test case: Test for graceful handling of unexpected errors, such as database failures or network interruptions.
* Test case: Verify that error messages are clear, concise, and provide guidance to users on how to resolve the issue.
* Test case: Test system recovery mechanisms, such as automatic rollback of transactions in case of failure.

1. **Integration Testing:**

* Test case: Test integration with external systems or APIs to ensure smooth data exchange.
* Test case: Verify proper error handling and recovery in case of integration failures.
* Test case: Test for compatibility with different browsers, operating systems, and devices.

1. **Performance Testing:**

* Test case: Measure system response times for common operations and ensure they meet acceptable thresholds.
* Test case: Test scalability by gradually increasing the load on the system and monitoring performance metrics.
* Test case: Identify and optimize any performance bottlenecks.

1. **Security Testing:**

* Test case: Perform penetration testing to identify potential vulnerabilities in the system.
* Test case: Test for SQL injection, cross-site scripting (XSS), and other common security threats.
* Test case: Verify proper implementation of authentication, authorization, and data encryption mechanisms.

1. **Backup and Recovery:**

* Test case: Test backup procedures to ensure data can be successfully restored in case of data loss or corruption.
* Test case: Verify that backup files are stored securely and can only be accessed by authorized personnel.
* Test case: Test recovery procedures to ensure minimal downtime in case of system failure.

1. **Usability Testing:**

* Test case: Evaluate the user interface for intuitiveness and ease of use.
* Test case: Verify that common workflows can be completed efficiently without unnecessary steps.
* Test case: Solicit feedback from actual users to identify areas for improvement.

By covering these critical cases with appropriate test cases, you can ensure the Faculty Contribution Management system is robust, secure, and reliable throughout its lifecycle.