PROJECT DEVELOPMENT PHASE

EXCEPTIONAL HANDLING

Date	23 October2023
Team ID	NM2023TMID06472
Project Name	Project –Building A Website Using Canva

EXCEPTIONAL HANDLING:

1. Identify Exceptions:
   - Recognize potential exceptional situations in your project, such as input errors, resource unavailability, or unexpected data.

2. Use Try-Catch Blocks:
   - Encapsulate the code that might raise exceptions within try-catch blocks. This allows you to catch and handle exceptions gracefully.

3. Choose Appropriate Exception Types:
   - Select or create specific exception types that accurately describe the nature of the problem. Custom exceptions can be used to represent project-specific issues.

4. Logging and Messaging:
   - Implement logging mechanisms to record exception details, including error messages, timestamps, and stack traces. This information is invaluable for debugging and tracing the source of issues.

5. Graceful Error Handling:
   - Handle exceptions gracefully by providing user-friendly error messages or fallback mechanisms to minimize disruptions to the user experience.

6. Cleanup Resources:
   - Use the `finally` block to ensure that resources, such as database connections or file handles, are properly released regardless of whether an exception is thrown.

7. Rethrow or Propagate:
   - Depending on the situation, you can choose to rethrow exceptions or propagate them up the call stack for higher-level error handling.

8. Define Clear Error Handling Policies:
   - Establish and document clear error handling policies that specify how different types of exceptions should be handled within your project.

9. Unit Testing for Exception Scenarios:
   - Create unit tests that specifically target exceptional scenarios to verify that your exception handling mechanisms work as expected.

10. Preventing Unhandled Exceptions:
    - Implement proactive measures to minimize the occurrence of exceptions, such as input validation and checking for resource availability before use.

11. Use Checked Exceptions (if applicable):
    - In languages that support checked exceptions, leverage them to enforce exception handling, ensuring that exceptions are acknowledged and addressed in the code.