Project Development Phase

**Debugging & Traceability**

|  |  |
| --- | --- |
| TEAM LEAD | R.Sree Bharathi |
| NM ID | BF65628C5F4C176991E4180F64CA6B6F |
| PROJECT NAME | Create Brand Name, Brand Email and Brand Logo |

Debugging and traceability are essential when implementing to ensure that data is collected accurately and that you can trace issues back to their source. Here are some steps to achieve this:

**Version Control System:** Use a version control system (e.g., Git) to track changes in your code and project files. This allows you to identify when and why certain changes were made, making it easier to trace back to the source of issues.

**Issue Tracking**: Implement an issue tracking system (e.g., Jira, Trello) to log and prioritize bugs, feature requests, and tasks. Assign issues to team members and keep a history of changes and discussions related to each issue.

**Logging and Monitoring**: Integrate robust logging and monitoring into your project. This allows you to track the behavior of your software and identify issues in real-time. Popular tools for logging and monitoring include ELK Stack, Prometheus, and Grafana.

**Automated Testing**: Develop automated tests for your software, including unit tests, integration tests, and end-to-end tests. Automated tests help catch and trace issues early in the development process.

**Continuous Integration/Continuous Deployment (CI/CD**): Implement a CI/CD pipeline to automate the build, testing, and deployment of your software. CI/CD ensures that changes are thoroughly tested before they are deployed, reducing the chances of introducing new issues.

**Code Reviews:** Conduct code reviews as part of your development process. Code reviews help identify potential issues, improve code quality, and provide traceability for changes made to the codebase.

**Documentation:** Maintain comprehensive documentation for your project, including architecture diagrams, API documentation, and user guides. Documentation helps with traceability by providing context for how different components work.

**Error Handling:** Implement robust error handling in your code to catch and log exceptions or unexpected behavior. This helps in diagnosing and debugging issues when they occur.

**Change Management Process**: Establish a clear change management process for making modifications to the project. This should include code reviews, testing procedures, and a rollback plan in case of issues.

**Root Cause Analysis:** When issues arise, conduct root cause analysis to understand why they occurred. Use tools like log analysis, debugging tools, and monitoring data to trace the problem back to its source.