Debugging & traceability

1. **Logging and Error Handling:** Implement comprehensive logging to capture errors and events. Use appropriate log levels to differentiate between informational, warning, and error messages. This helps in tracking issues and understanding the application flow.
2. **Debugging Tools:** Integrate debugging tools that allow developers to identify and resolve issues efficiently. Use debugging libraries, code analyzers, and debugging interfaces to trace the execution path and pinpoint problematic areas.
3. **Version Control System:** Utilize a version control system such as Git to manage source code changes. Encourage the use of meaningful commit messages to track the changes made during the development process. This aids in maintaining a clear history of code modifications.
4. **Error Monitoring and Reporting:** Set up error monitoring and reporting mechanisms to automatically detect and report application errors. This ensures that the development team is promptly notified of any critical issues, enabling them to respond quickly and efficiently.
5. **Unit Testing and Test Automation:** Develop a robust suite of unit tests to verify the application's components and functionalities. Automate these tests to ensure that the application maintains expected behavior throughout the development lifecycle.
6. **Code Documentation:** Document the code thoroughly, including inline comments, function descriptions, and API documentation. This promotes a better understanding of the codebase and assists in identifying potential issues during the development process.
7. **Traceability through Unique Identifiers:** Assign unique identifiers to each request and transaction within the application. Maintain a clear traceability mechanism to track the flow of requests and approvals, ensuring that every action is linked to a specific identifier for easy auditing and debugging.