**Debugging:**

1. **System.debug Statements:**
   * Use the **System.debug** statements in your Apex code to print log statements. These statements help you understand the flow of your code, variable values, and identify any issues during execution.

apexCopy code

System.debug('Variable Value: ' + myVariable);

1. **Developer Console:**
   * Salesforce provides a web-based Developer Console that allows you to monitor and debug your code. You can execute anonymous Apex, inspect logs, set breakpoints, and view variable values.
2. **Debug Logs:**
   * Enable debug logs for specific users to capture detailed information during execution. You can configure log levels and monitor the logs in the Salesforce Setup under Monitoring > Debug Logs.
3. **Apex Debugger:**
   * Use the Apex Debugger in Salesforce Extensions for Visual Studio Code or other Salesforce development environments that support debugging. This allows you to set breakpoints, inspect variables, and step through your code.

**Traceability:**

1. **Logging Ids and Timestamps:**
   * Include unique identifiers and timestamps in your logs to trace the execution flow. This helps in associating logs with specific transactions or events.
2. **Custom Logging Framework:**
   * Develop a custom logging framework that records key events and details during the execution of your code. This can include method entry/exit, important decisions, and error conditions.
3. **Exception Handling and Logging:**
   * Implement robust exception handling in your code and log exceptions with relevant details. This ensures that when errors occur, you can trace back to the root cause quickly.

apexCopy code

try { // Code block } catch (Exception e) { System.debug('Exception: ' + e.getMessage()); throw e; }

1. **Unique Identifiers on Records:**
   * Ensure that critical records in your school management system have unique identifiers. This makes it easier to trace the flow of data across different components.
2. **Audit Trails:**
   * Leverage Salesforce's audit trails to monitor changes made to the setup configuration. This helps in tracking changes to classes, triggers, or other metadata that may impact your application.
3. **Integration Monitoring:**
   * If your school management system integrates with external systems, implement logging and monitoring for integration transactions. This includes recording requests, responses, and any errors that may occur during integration.
4. **Change Sets and Version Control:**
   * Use change sets or version control systems (such as Git) to track changes made to your Salesforce application. This ensures traceability across different releases and helps in rolling back changes if needed.
5. **Documentation:**
   * Maintain comprehensive documentation that describes the architecture, data models, and business processes of your school management application. This documentation serves as a reference for developers and administrators, aiding in traceability.