1. Error Identification: Exception handling starts with the identification of potential errors or exceptional situations that can occur within the system. This can include scenarios like database connection failures, input validation errors, file upload issues, or unexpected data formats.
2. Exception Classes: ATS developers typically define a set of exception classes that represent different types of errors or exceptional situations. Each class can contain information about the error, its severity, and details that can help in debugging and troubleshooting.
3. Try-Catch Blocks: Developers enclose the code that might generate exceptions within "try-catch" blocks. The "try" block contains the code that may cause an exception, while the "catch" block is responsible for handling the exception when it occurs.
4. Logging: When an exception is caught, it is important to log the error details. This includes information like the type of exception, the time it occurred, the user or application context, and any relevant data. Logging helps in diagnosing and resolving issues after they occur.
5. Graceful Degradation: Exception handling aims to ensure that the ATS can continue to function in a degraded mode or gracefully recover from errors. For example, if a database connection error occurs, the system might switch to a read-only mode or provide a user-friendly error message instead of crashing.
6. User-Friendly Messages: When exceptions occur, it's important to provide clear and user-friendly error messages to both applicants and system administrators. These messages should explain the issue and, if possible, suggest actions to resolve it.
7. Notification: In some cases, it is necessary to notify administrators or support teams when critical exceptions occur. This allows them to respond promptly to resolve the issue and minimize any disruption to the recruitment process.
8. Escalation: In certain situations, exceptions might require escalation to a higher level of support or involve manual intervention. Exception handling mechanisms should include protocols for such cases.
9. Retry Mechanism: In cases where an exception is temporary (e.g., a network timeout), the ATS may implement a retry mechanism. This automatically attempts the failed operation again after a short delay to see if the issue resolves itself.
10. Contingency Plans: Exception handling should be part of a broader contingency plan that outlines how the system behaves during major

failures or disasters. This plan may include data backup and recovery procedures to safeguard applicant information.