Gather Application Information:

1. name
2. version
3. purpose
4. and criticality to the business.
5. List the technologies, frameworks, and libraries used in the application (e.g., .NET Core, ASP.NET MVC, Entity Framework).
6. Document the architecture and
7. components of the application,
8. including frontend,
9. backend,
10. databases,
11. APIs,
12. and third-party integrations.

Capture Environment Details:

1. server infrastructure,
2. operating system,
3. network setup,
4. and dependencies.
5. List any software dependencies,
6. runtime environments (e.g., .NET Framework versions),
7. and third-party services required for the application to function properly.

Document Deployment Process:

1. build procedures,
2. deployment scripts,
3. and configuration management.
4. Specify deployment targets,
5. deployment tools (e.g., Azure DevOps, Jenkins),
6. and any manual steps or prerequisites required for deploying updates or patches.

Capture Monitoring and Logging:

1. monitoring and logging practices for the .NET application,
2. including metrics, alerts,
3. and log management tools used to monitor application health and performance.
4. Specify key performance indicators (KPIs),
5. error thresholds,
6. and escalation procedures for handling alerts and incidents.

Document Support Procedures:

1. Define support procedures and workflows for handling production incidents,
2. including incident reporting,
3. triage,
4. prioritization, and resolution.
5. Document communication channels,
6. contact information,
7. and escalation paths for coordinating with stakeholders,
8. development teams,
9. and third-party vendors.

Capture Troubleshooting Steps:

1. Document common troubleshooting steps and best practices for diagnosing and resolving issues in .NET applications,
2. including error handling,
3. logging strategies,
4. and debugging techniques.
5. Provide guidance on analyzing log files,
6. tracing requests,
7. and identifying root causes of performance bottlenecks or errors.

Document Disaster Recovery Plan:

1. Document disaster recovery procedures.
2. and contingency plans for mitigating risks
3. and restoring service in case of catastrophic events (e.g., server failures, data breaches, natural disasters).
4. Specify backup and recovery strategies,
5. data retention policies,
6. and failover mechanisms to ensure business continuity.

Provide Training and Knowledge Transfer:

1. Conduct training sessions or workshops to educate support teams on the .NET application's architecture,
2. functionality,
3. and support requirements.
4. Provide hands-on demonstrations,
5. walkthroughs,
6. and simulations to familiarize support staff with common tasks,
7. troubleshooting scenarios,
8. and operational procedures.

Review and Validate Documentation:

1. Review and validate the KT documentation with key stakeholders,

2. subject matter experts,
3. and support teams to ensure accuracy,
4. completeness, and relevance.
5. Update the documentation regularly to reflect changes in the application,
6. environment, or support processes.

Feedback and Follow-Up:

1. Provide feedback to the KT guys on the clarity,
2. completeness, and
3. usefulness of the KT documentation and training sessions.
4. Follow up with the KT guys for any additional questions, clarifications, or support needs that arise after the initial KT sessions.

Future Plans and Roadmap:

1. Are there any upcoming changes, upgrades, or enhancements planned for the .NET application?
2. How will these changes impact production support and maintenance activities?

Security and Compliance:

1. What security measures are implemented in the .NET application to protect against vulnerabilities and unauthorized access?
2. Are there any compliance requirements (e.g., GDPR, HIPAA) that need to be considered for production support?

Performance Optimization:

1. What strategies are employed for optimizing the performance of the .NET application?
2. Can you provide examples of performance tuning techniques used in the production environment?

Capacity Planning:

1. How is capacity planning handled for the .NET application to ensure scalability and resource allocation?
2. What are the key metrics used for monitoring resource utilization and capacity thresholds?

Change Management:

1. What is the process for managing changes and updates to the .NET application's configuration or infrastructure?
2. How are changes reviewed, approved, and documented before implementation?

Incident Response and Escalation:

1. What is the escalation path for handling critical incidents in the production environment?
2. Are there any predefined response SLAs (Service Level Agreements) for addressing incidents?

Documentation Accessibility:

1. How is KT documentation accessed and maintained by support team members?
2. Are there any tools or platforms used for centralized documentation management?

Training and Onboarding:

1. How are new team members onboarded and trained for supporting the .NET application?
2. Is there a formal training program or mentorship process in place for knowledge transfer?

Vendor and Third-Party Integration:

1. Are there any third-party vendors or external systems integrated with the .NET application?
2. How are dependencies managed and maintained for external integrations?

Continuous Improvement:

1. What mechanisms are in place for gathering feedback and identifying areas for improvement in production support processes?
2. How are lessons learned from incidents or operational challenges incorporated into future improvements?

Knowledge Retention:

1. How is institutional knowledge preserved and transferred within the support team?
2. Are there any measures in place to prevent knowledge silos or dependencies on individual team members?