**System Design Document (SDD) Template**

**Project Name:** [Enter Project Name]  
**Version:** [Enter Version]  
**Author(s):** [Your Name(s)]  
**Date:** [DD/MM/YYYY]  
**Status:** [Draft/Final]

**1. Introduction & Overview**

**1.1 Purpose**

[Briefly describe the purpose of the system and this document.]

**1.2 Scope**

[Define the system’s boundaries, key functionalities, and intended users.]

**1.3 Stakeholders**

[List all key stakeholders (Product Owner, Developers, Testers, Business Users, etc.).]

**1.4 Assumptions & Constraints**

[Highlight any known technical or business limitations.]

**2. Architecture Decisions & Principles**

**2.1 Design Principles**

[List key architectural principles such as Microservices, Event-Driven, Domain-Driven Design, etc.]

**2.2 Technology Stack**

| **Component** | **Technology** | **Justification** |
| --- | --- | --- |
| Backend | [e.g., Node.js, Java Spring Boot] | [Reason for choosing] |
| Frontend | [e.g., React, Angular] | [Reason for choosing] |
| Database | [e.g., PostgreSQL, MongoDB] | [Reason for choosing] |
| Cloud | [e.g., AWS, GCP, Azure] | [Reason for choosing] |

**2.3 Trade-offs & Justifications**

[Discuss trade-offs between different architecture choices.]

**2.4 Security Considerations**

[List security measures such as authentication, authorization, encryption, compliance, etc.]

**3. System Context & Use Cases**

**3.1 Context Diagram**

[Insert a high-level diagram showing the system and external interactions.]

**3.2 Actors & Use Cases**

[List actors and describe how they interact with the system.]

**3.3 Business Scenarios**

[Provide real-world workflows or user stories.]

**4. High-Level Architecture**

**4.1 Component Diagram**

[Include a visual representation of system components and their interactions.]

**4.2 Deployment Architecture**

[Describe the hosting model: On-premise, Cloud, Hybrid, etc.]

**4.3 System Boundaries & Dependencies**

[Explain integrations with third-party tools and services.]

**5. Detailed Design**

**5.1 Microservices / Modular Design**

[List services/modules and their responsibilities.]

**5.2 Class Diagrams & Data Models**

[Provide entity-relationship diagrams (ERD) for databases.]

**5.3 Sequence Diagrams**

[Illustrate interactions between components over time.]

**5.4 State Diagrams**

[Show lifecycle transitions of key objects in the system.]

**6. Non-Functional Requirements (NFRs)**

**6.1 Performance & Scalability**

[Define expected system load, caching strategies, database sharding, etc.]

**6.2 Availability & Reliability**

[Discuss SLAs, failover mechanisms, and disaster recovery plans.]

**6.3 Security & Compliance**

[List compliance requirements (e.g., GDPR, HIPAA) and security best practices.]

**6.4 Observability & Logging**

[Describe monitoring, alerting, and log management strategies.]

**7. DevOps & CI/CD Strategy**

**7.1 Build & Deployment Pipelines**

[Describe CI/CD workflows using Jenkins, GitHub Actions, etc.]

**7.2 Infrastructure as Code (IaC)**

[Specify Terraform, CloudFormation, or Kubernetes deployment strategies.]

**7.3 Testing Strategy**

[Define Unit, Integration, Performance, and Security testing approaches.]

**8. Risks & Mitigation Strategies**

**8.1 Known Risks**

[List risks such as performance bottlenecks, security vulnerabilities, vendor lock-in, etc.]

**8.2 Contingency Plans**

[Describe failover, backup strategies, and incident response plans.]

**9. Future Considerations & Roadmap**

**9.1 Scalability Plans**

[Outline how the system can scale as demand grows.]

**9.2 Feature Enhancements**

[Suggest potential feature upgrades or system refinements.]

**9.3 Technology Evolution**

[Discuss upcoming changes in tech stack or cloud infrastructure.]

**10. Appendices & References**

**10.1 Glossary**

[List technical terms and definitions.]

**10.2 Reference Documents**

[Provide links to relevant API documentation, SDKs, and other resources.]

**How to Use This Template?**

* **Modify it as needed** to suit your project’s complexity.
* **Use diagrams** (draw.io, Lucidchart, PlantUML) for visual clarity.
* **Keep it concise**—avoid excessive detail unless necessary.