Applied System Design

Course Name: Applied System Design	Course Code	L-T-P	Credits
	ETCCSD275	4-0-0	4
Type of Course:	Major		
Pre-requisite(s):	Software Engineering principles, Operating Systems, and familiarity with system architecture concepts.		

Course Perspective:

This course provides an in-depth exploration of system design principles and practices in building scalable, robust, and maintainable software systems. It emphasizes design patterns, architectural styles, performance considerations, and integration strategies for distributed systems. Learners will analyze real-world case studies and design blueprints to enhance their ability to design production-ready systems in various domains such as e-commerce, IoT, finance, and cloud-native platforms.

The Course Outcomes (COs).

COs	Statements
CO 1	Describing the principles of applied system design and architectural styles for scalable systems.
CO 2	Applying design patterns and modeling techniques to develop modular and maintainable architectures.
CO 3	Analyzing trade-offs in system design related to scalability, availability, performance, and cost.

CO 4	Designing end-to-end system blueprints based on use cases, functional requirements, and constraints.

CO = Course outcomes. A student is expected to have learnt concepts anddemonstrated/developed abilities or skills related to software engineering/ management of the software at the end of the course.

Course Outline:

Unit Number: 1	**Introduction to System Design and Architectural Thinking**	**No. of hours: 15**
Topics Covered: • System design vs software design • Key quality attributes: scalability, availability, reliability, latency, maintainability • Monolithic vs microservices architecture • Real-World Use Case: Scaling a monolithic e-commerce application to microservices		
Unit Number: 2	**Design Patterns and Modeling for Large Systems**	**No. of hours: 15**
Topics Covered: • Object-oriented and component-based design principles • Design patterns: Singleton, Factory, Observer, Proxy, Adapter, Circuit Breaker • UML diagrams: Class, Sequence, Component, Deployment • Real-World Use Case: Designing a payment gateway integration system using design patterns		
Unit Number: 3	**Scalable and Distributed System Design**	**No. of hours: 15**

Topics Covered: • Load balancing, caching strategies (LRU, LFU), CDN • Database sharding, replication, and eventual consistency • CAP Theorem and distributed transaction patterns • Real-World Use Case: Designing a real-time collaborative document editing platform		
Unit Number: 4	**System Design Strategies for Performance, Security, and Maintainability**	**No. of hours: 15**
Topics Covered: • Performance tuning: profiling, benchmarking, bottleneck identification • Security by design: authentication, authorization, encryption • API design best practices: REST, GraphQL, throttling, rate-limiting • Maintainability: logging, monitoring, documentation • Real-World Use Case: Designing a secure and observable ride-sharing system backend		

Text and Reference Books:

- Martin Kleppmann – Designing Data-Intensive Applications
- Sam Newman – Building Microservices
- Eric Evans – Domain-Driven Design
- Mark Richards – Software Architecture Patterns
- Bass, Clements & Kazman – Software Architecture in Practice

Learning Outcomes

Inside the Classroom:

1. **Conceptual Understanding:**

- Students will gain deep knowledge of key principles such as modularity, abstraction, scalability, and reliability in system design.
- Analyze system design trade-offs (e.g., consistency vs availability, cost vs performance).

2. **Architectural Thinking:**
   - Apply architectural patterns like layered, microservices, client-server, and event-driven designs through interactive lectures and case-based discussions.
3. **Modelling and Design Patterns:**
   - Construct system models using UML diagrams and apply appropriate design patterns in case scenarios discussed in class.
4. **Analytical and Problem-Solving Skills:**
   - Evaluate the impact of different design choices through classroom design exercises and real-world problem walkthroughs.
5. **Communication and Collaboration:**
   - Develop the ability to communicate system architecture clearly using diagrams, flowcharts, and design documentation during group activities.

Outside the Classroom Learning:

1. **Application of Concepts:**
   - Apply architectural thinking and design strategies to small-scale projects or prototypes as homework or term assignments.
2. **Real-World Case Study Analysis:**
   - Analyze well-known system design case studies (e.g., YouTube, Netflix, Uber) through research-based tasks and reflective write-ups.
3. **Self-Directed Learning:**

- Explore current industry practices in system architecture, distributed systems, and cloud design patterns via online learning platforms (e.g., System Design Primer on GitHub, YouTube lectures by Gaurav Sen).

4. **Technical Communication:**
    - Prepare and present system design proposals outside class hours, improving technical documentation and presentation skills.
5. **Peer Learning and Collaboration:**
    - Collaborate in small groups to critique and improve each other's system design blueprints through informal study groups and peer review sessions.