# MODEL QUESTION PAPER

# Advanced Software Engineering

## Q1

### (a) Objective Type (5 × 1 = 5 marks)

i. Which SDLC model works best when requirements are expected to change frequently?
**Answer: C. Agile**

ii. Agile development delivers software in:
**Answer: B. Sprints**

iii. Microservices communicate mainly using:
**Answer: B. APIs**

iv. Which diagram shows external users and systems in C4?
**Answer: C. Context**

v. Architecture Decision Records (ADR) are used to:
**Answer: B. Record design decisions**

---

### (b) Understanding (7 marks)

**Software Development Life Cycle (SDLC)** is a systematic process used to develop high-quality software in a structured and organized manner. It defines the different phases involved in software development from start to end.

**Main phases of SDLC are:**

1. **Requirement Analysis** – Understanding user needs and system requirements.
2. **System Design** – Creating architecture and design specifications.
3. **Implementation** – Actual coding of the software.
4. **Testing** – Verifying and validating the software for defects.
5. **Deployment** – Releasing the software to users.
6. **Maintenance** – Fixing bugs and adding enhancements after deployment.

**Agile methodology improves adaptability** compared to traditional models like Waterfall in the following ways:

- Agile works in **short iterations called sprints**, allowing frequent feedback.
- Changes in requirements can be easily accommodated even in later stages.
- Continuous testing and integration reduce risk and improve quality.
- Customer involvement is regular, ensuring the product meets real needs.

Thus, Agile provides more flexibility, faster delivery, and better response to changing requirements than traditional SDLC models.

---

### (c) Application (8 marks)

**Assumptions:**

- The University Student Portal will be used by students, faculty, and administrators.
- Requirements such as course registration, attendance tracking, and result display may change frequently.
- Continuous improvements and updates are expected.

**Recommended SDLC Model: Agile Model**

**Justification:**

- The Agile model supports **frequent requirement changes**, which is essential for a student portal.
- Features like course registration and results can be developed and delivered **incrementally**.
- Regular feedback from users (students and staff) helps improve usability.
- Issues can be identified and fixed early through continuous testing.
- Faster delivery of updates ensures the system stays relevant and efficient.

**Conclusion:**
Agile SDLC is the most suitable model for a University Student Portal because it provides flexibility, faster releases, continuous improvement, and better alignment with user needs.

## Q2 (Unit I)

### (a) Objective Type (5 × 1 = 5 marks)

i. Cloud-native applications focus on:
**Answer: B. Scalability and automation**

ii. API-first approach helps teams to:
**Answer: B. Develop in parallel**

iii. UML Deployment diagram represents:
**Answer: B. Runtime environment**

iv. Microservices usually follow:
**Answer: B. Database per service**

v. Which model supports early partial delivery?
**Answer: B. Incremental**

---

## (b) Understanding (7 marks)

**Microservices architecture** is a software architectural style in which an application is built as a collection of small, independent services. Each service focuses on a specific business functionality, runs independently, and communicates with other services using lightweight APIs (usually REST).

**Key characteristics of microservices architecture:**

- Each service is **independently deployable**
- Services communicate through **APIs**
- Each service can have its **own database**
- Services are loosely coupled

**Advantages of Microservices over Monolithic Architecture:**

1. **Scalability:** Individual services can be scaled independently.
2. **Flexibility:** Different technologies can be used for different services.
3. **Faster development:** Teams can work on services in parallel.
4. **Fault isolation:** Failure of one service does not crash the entire system.
5. **Easy maintenance:** Smaller codebases are easier to update and manage.

Thus, microservices provide better scalability, resilience, and agility compared to monolithic systems.

---

## (c) Application (8 marks)

**Assumptions:**

- The University Library System is used by students, librarians, and faculty.
- The system must support online access and high availability.
- Different library functions can work independently.

**High-level Microservices Architecture for a University Library System:**

**Proposed Microservices:**

1. **User Management Service** – Manages students, faculty, and librarian accounts.
2. **Book Catalog Service** – Handles book details, search, and availability.
3. **Issue & Return Service** – Manages book issuing, returns, and due dates.
4. **Fine Management Service** – Calculates and tracks late return fines.
5. **Notification Service** – Sends email/SMS alerts for due dates and fines.

**Architecture Explanation:**

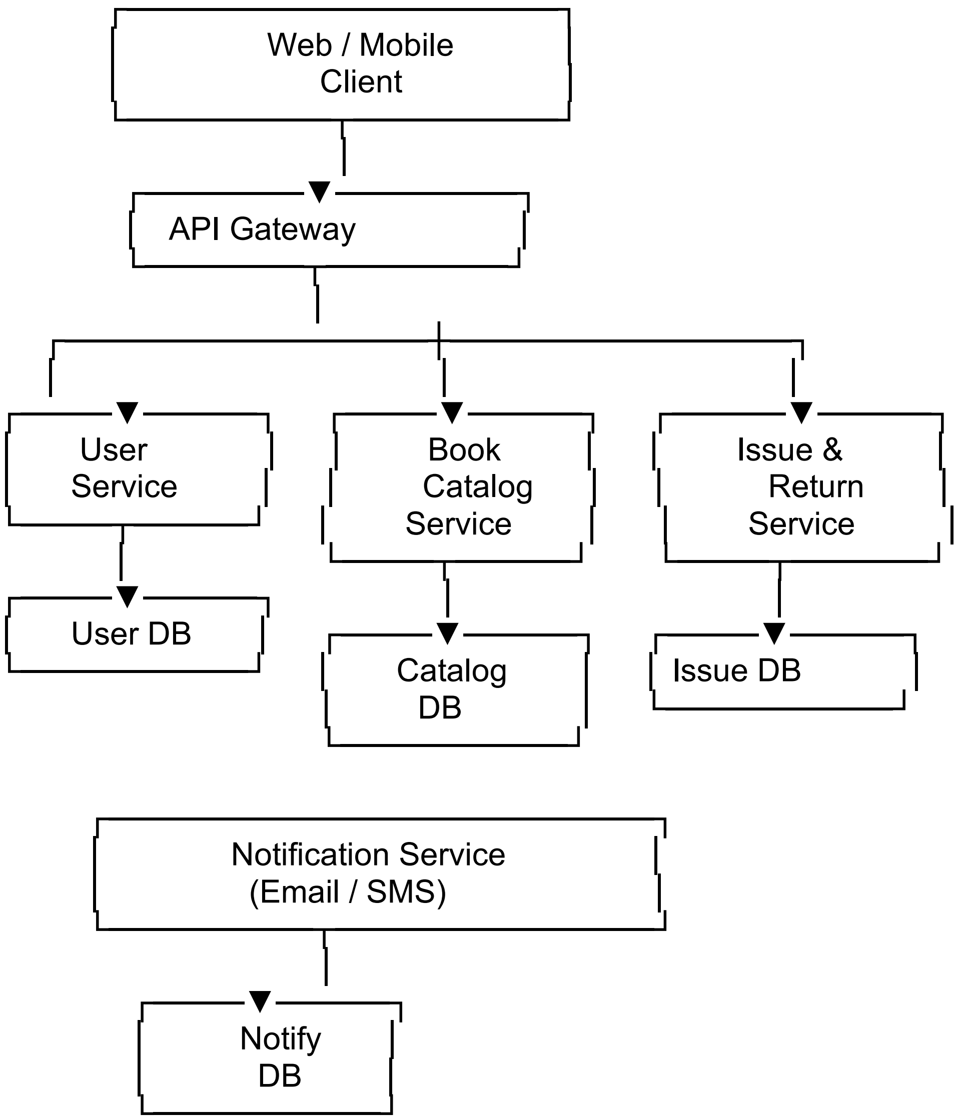
- Each service has its **own database** to avoid tight coupling.

- Services communicate using **REST APIs**.
- An **API Gateway** handles requests from web or mobile applications.
- Services can be independently deployed and scaled.

**Justification:**

- Independent services allow easier maintenance and future expansion.
- Faults in one service do not affect the entire system.
- Scalability ensures smooth operation during peak usage (exam periods).

**Conclusion:**
A microservices architecture makes the University Library System scalable, flexible, and reliable while supporting future enhancements.



## Q3 (Unit I)

### (a) Objective Type (5 × 1 = 5 marks)

i. Planning Poker is used for:
**Answer: C. Estimation**

ii. Story points represent:
**Answer: C. Effort and complexity**

iii. Estimation uncertainty is highest during:
**Answer: B. Early stages**

iv. Function Point Analysis measures:
**Answer: B. User functionality**

v. Agile planning happens in:
**Answer: B. Sprints**

---

## (b) Understanding (7 marks)

**Software project estimation** is the process of predicting the effort, time, cost, and resources required to complete a software project. Accurate estimation helps in proper planning, budgeting, scheduling, and risk management.

**Importance of software project estimation:**

- Helps in **realistic project planning and scheduling**
- Supports **resource allocation**
- Reduces risk of **cost and time overruns**
- Improves **customer confidence and decision-making**

**Estimation Techniques (any two):**

1. **Planning Poker**
    - Used mainly in Agile projects.
    - Team members estimate tasks using story points.
    - Encourages team discussion and consensus.
    - Reduces individual bias in estimation.
2. **Function Point Analysis (FPA)**
    - Measures software size based on **user-visible functionality**.
    - Independent of programming language.
    - Useful during early project stages.
    - Helps estimate effort and cost accurately.

Thus, estimation techniques help in managing project scope and ensuring successful project delivery.

---

## (c) Application (8 marks)

**Assumptions:**

- The Online Examination System is a new project.
- No previous similar project data is available.
- Requirements may evolve during development.

**Recommended Estimation Technique: Planning Poker (Agile Estimation)**

**Justification:**

- Planning Poker does **not require historical data**, making it suitable for new projects.
- Estimation is based on **team experience and understanding**.
- Story points capture **effort, complexity, and uncertainty** effectively.
- Encourages discussion among developers, testers, and analysts.
- Estimates can be refined after each sprint as requirements become clearer.

**Conclusion:**
Planning Poker is the most suitable estimation technique for a new Online Examination System because it handles uncertainty well and supports continuous improvement in estimates during Agile development.

## Q4 (Unit II)

### (a) Objective Type (5 × 1 = 5 marks)

i. Which design pattern ensures only one instance of a class exists?
**Answer: B. Singleton**

ii. Circuit Breaker pattern is used to handle:
**Answer: B. Cascading failures**

iii. Hexagonal architecture separates:
**Answer: B. Business logic and infrastructure**

iv. CQRS separates:
**Answer: A. Read and write models**

v. BFF pattern supports:
**Answer: A. Multiple frontends**

---

### (b) Understanding (7 marks)

**GoF (Gang of Four) design patterns** are a set of reusable solutions to commonly occurring software design problems. These patterns were introduced by four authors and provide best practices for object-oriented software design.

**Types of GoF Design Patterns:**

1. **Creational Patterns** – Deal with object creation (e.g., Singleton, Factory).

2. **Structural Patterns** – Deal with object composition (e.g., Adapter, Facade).
3. **Behavioral Patterns** – Deal with object interaction and communication (e.g., Observer, Strategy).

**Benefits of GoF Design Patterns:**

- Promote **reusability and maintainability**
- Improve **code flexibility and scalability**
- Provide **standard solutions**, making code easier to understand
- Reduce **tight coupling** between components
- Help in building **robust and extensible systems**

Thus, GoF design patterns help developers create well-structured and easily maintainable software systems.

---

## (c) Application (8 marks)

**Assumptions:**

- The University Notification System sends messages through email, SMS, and mobile apps.
- Notifications must reach multiple users simultaneously.
- New notification channels may be added in the future.

**Suitable Design Patterns:**

1. **Observer Pattern**
   - Used to notify multiple subscribers at the same time.
   - Email, SMS, and Mobile App act as observers.
   - When a notification is generated, all channels are automatically updated.
2. **Factory Pattern**
   - Used to create different notification objects (Email, SMS, App).
   - Allows easy addition of new notification types without modifying existing code.
3. **Singleton Pattern**
   - Ensures only one instance of the Notification Manager exists.
   - Helps in centralized control of notification delivery.

**Justification:**

- Observer pattern ensures real-time notification delivery.
- Factory pattern improves extensibility and reduces code changes.
- Singleton pattern ensures consistent notification management.

**Conclusion:**
Using Observer, Factory, and Singleton patterns makes the University Notification System scalable, flexible, and easy to maintain.

# Q5 (Unit II)

## (a) Objective Type (5 × 1 = 5 marks)

i. Adapter pattern is mainly used to:
**Answer: B. Convert interfaces**

ii. Retry pattern should be avoided when failures are:
**Answer: C. Permanent**

iii. Facade pattern provides:
**Answer: B. Simplified interface**

iv. Decorator pattern is used to:
**Answer: A. Add responsibilities dynamically**

v. Bulkhead pattern improves:
**Answer: B. Fault isolation**

---

## (b) Understanding (7 marks)

**Hexagonal Architecture**, also known as **Ports and Adapters Architecture**, is an architectural pattern that separates the core business logic from external systems such as databases, user interfaces, and third-party services.

**Key concepts of Hexagonal Architecture:**

- **Core Domain (Business Logic):** Contains application rules and use cases.
- **Ports:** Interfaces that define how the core interacts with the outside world.
- **Adapters:** Implementations of ports that connect to external systems like UI or databases.

**Advantages of Hexagonal Architecture:**

- Improves **testability** of business logic.
- Reduces **tight coupling** with external systems.
- Makes the system **technology-independent**.
- Easier maintenance and future enhancements.
- Supports multiple interfaces (web, mobile, API) easily.

Thus, Hexagonal Architecture results in a clean, flexible, and maintainable system design.

---

## (c) Application (8 marks)

**Assumptions:**

- The University ERP system includes modules such as admissions, attendance, exams, and finance.
- The system interacts with databases, web interfaces, and external services.
- High test coverage is required to ensure reliability.

**How Hexagonal Architecture Improves Testability:**

1. **Business Logic Isolation**
   - Core ERP logic is separated from infrastructure.
   - Business rules can be tested without databases or UI.
2. **Mocking External Dependencies**
   - Ports allow use of mock adapters during testing.
   - External services (payment, email) can be simulated.
3. **Independent Testing**
   - Unit tests focus only on application logic.
   - Faster and more reliable test execution.
4. **Easy Integration Testing**
   - Real adapters can be plugged in when needed.
   - Changes in technology do not affect core tests.

**Conclusion:**
By isolating business logic and using ports and adapters, Hexagonal Architecture significantly improves the testability, reliability, and maintainability of a University ERP system.

## Q6 (Unit II)

### (a) Objective Type (5 × 1 = 5 marks)

i. Flyweight pattern helps in reducing:
**Answer: B. Memory usage**

ii. Proxy pattern is used for:
**Answer: A. Access control**

iii. CQRS is best suited when:
**Answer: A. Reads dominate writes**

iv. Microservice patterns address:
**Answer: B. Resilience and scalability**

v. Which pattern isolates failures between services?
**Answer: B. Bulkhead**

---

### (b) Understanding (7 marks)

**CQRS (Command Query Responsibility Segregation)** is an architectural pattern that separates the system's **write operations (commands)** from **read operations (queries)**.
Commands update data, while queries only retrieve data, often using different models or databases.

**Benefits of CQRS:**

- Improves performance and scalability
- Optimizes read and write workloads independently
- Suitable for systems with heavy read operations

**Backend-for-Frontend (BFF)** is a pattern where a **dedicated backend service is created for each frontend** (web, mobile, etc.).
Each BFF provides APIs tailored to the specific needs of that frontend.

**Benefits of BFF:**

- Reduces complexity on the frontend
- Improves performance by returning only required data
- Allows independent evolution of frontends

Thus, CQRS improves scalability, while BFF improves frontend efficiency and flexibility.

---

## (c) Application (8 marks)

**Assumptions:**

- The Library Management System is used by students, librarians, and administrators.
- The system must be highly available and fault tolerant.
- Failures in one module should not affect the entire system.

**Resilient Architecture Design:**

**Key Components:**

1. **API Gateway**
   - Entry point for all client requests
   - Handles routing, authentication, and rate limiting
2. **Microservices**
   - Book Catalog Service
   - Issue & Return Service
   - User Management Service
   - Notification Service
3. **Resilience Patterns Used:**
   - **Circuit Breaker:** Prevents cascading failures when a service is down
   - **Bulkhead:** Isolates services so failure in one does not impact others
   - **Retry (with limits):** Handles temporary failures
   - **CQRS:** Improves performance for heavy read operations (search, availability)
4. **Independent Databases**
   - Each service has its own database for fault isolation

**Justification:**

- Service isolation ensures system stability
- Circuit breaker and bulkhead patterns improve reliability
- CQRS enhances performance during peak usage

- Independent services allow faster recovery and scaling

**Conclusion:**
By using microservices with resilience patterns like Circuit Breaker, Bulkhead, and CQRS, the Library Management System becomes highly reliable, scalable, and fault tolerant.

## Q7 (Unit III)

### (a) Objective Type (5 × 1 = 5 marks)

i. Typical Sprint duration is:
**Answer: B. 1–4 weeks**

ii. Kanban limits work using:
**Answer: B. WIP limits**

iii. CI/CD automates:
**Answer: B. Build and testing**

iv. Shift-left testing means:
**Answer: B. Testing early**

v. Unit tests are written by:
**Answer: B. Developers**

---

### (b) Understanding (7 marks)

**Scrum** is an Agile framework used for managing and developing complex software products. It focuses on iterative development, continuous feedback, and team collaboration.

*Scrum Roles:*

1. **Product Owner**
   - Defines product vision and priorities.
   - Manages and prioritizes the product backlog.
2. **Scrum Master**
   - Facilitates Scrum events.
   - Removes impediments and ensures Scrum practices are followed.
3. **Development Team**
   - Cross-functional team responsible for delivering increments of the product.

*Scrum Events:*

- **Sprint:** Fixed time-boxed iteration (1–4 weeks).
- **Sprint Planning:** Decide what to build in the sprint.

- **Daily Scrum:** Short daily meeting for progress tracking.
- **Sprint Review:** Demonstration of completed work.
- **Sprint Retrospective:** Discuss improvements for next sprint.

*Scrum Artifacts:*

- **Product Backlog**
- **Sprint Backlog**
- **Increment**

Thus, Scrum provides transparency, inspection, and adaptation throughout development.

---

## (c) Application (8 marks)

**Assumptions:**

- The University Attendance Management System is web-based.
- Frequent updates and bug fixes are required.
- High reliability and accuracy are critical.

**CI/CD Pipeline Design:**

**Pipeline Stages:**

1. **Source Code Management**
   - Developers push code to a Git repository.
2. **Continuous Integration**
   - Automated build is triggered.
   - Unit tests and code quality checks are executed.
3. **Automated Testing**
   - Integration and regression tests are run.
   - Ensures attendance logic works correctly.
4. **Build & Packaging**
   - Application is packaged (e.g., containerized).
5. **Deployment**
   - Automatically deployed to staging environment.
   - After approval, deployed to production.
6. **Monitoring**
   - Logs and performance metrics are monitored.
   - Alerts generated for failures.

**Justification:**

- Early testing reduces defects.
- Automation ensures faster and reliable releases.
- Continuous monitoring improves system reliability.

**Conclusion:**
A well-designed CI/CD pipeline ensures faster delivery, better quality, and high availability of the University Attendance Management System.

## Q8 (Unit III)

### (a) Objective Type (5 × 1 = 5 marks)

i. SAFe is used for:
**Answer: B. Agile at enterprise scale**

ii. Definition of Done ensures:
**Answer: B. Quality completeness**

iii. Regression testing ensures:
**Answer: B. Existing features still work**

iv. Chaos Engineering tests:
**Answer: B. System resilience**

v. DevOps aims to reduce:
**Answer: C. Silos**

---

### (b) Understanding (7 marks)

**CI/CD (Continuous Integration and Continuous Delivery/Deployment)** is a DevOps practice that automates the process of building, testing, and deploying software frequently and reliably.

**Continuous Integration (CI):**

- Developers regularly merge code into a shared repository.
- Automated builds and tests are triggered.
- Helps detect defects early.

**Continuous Delivery/Deployment (CD):**

- Code changes are automatically prepared for release.
- Deployment to staging or production is automated.
- Ensures faster and reliable releases.

**Role of CI/CD in Agile and DevOps:**

- Supports **Agile principles** by enabling frequent and incremental delivery.
- Enhances **collaboration** between development and operations teams.
- Reduces manual errors and deployment risks.

- Improves software quality through continuous testing.

Thus, CI/CD acts as the backbone of Agile and DevOps practices.

---

## (c) Application (8 marks)

**Assumptions:**

- The University Result Processing System is accessed by a large number of users simultaneously.
- Peak load occurs during result declaration.
- High availability and accuracy are critical.

**Suggested Testing Strategies:**

1. **Load Testing**
    - Tests system behavior under expected peak user load.
2. **Stress Testing**
    - Evaluates system performance beyond normal limits.
3. **Regression Testing**
    - Ensures existing functionalities work after updates.
4. **Chaos Testing**
    - Simulates failures to test system resilience.

**Suggested DevOps Practices:**

1. **Auto-scaling**
    - Automatically scales resources during peak load.
2. **CI/CD Pipeline**
    - Enables quick fixes and reliable deployments.
3. **Monitoring and Alerting**
    - Detects failures in real time.
4. **Blue-Green or Canary Deployment**
    - Reduces downtime during releases.

**Conclusion:**
By combining proper testing strategies with DevOps practices such as auto-scaling, CI/CD, and continuous monitoring, the University Result Processing System can achieve high reliability and handle peak load effectively.