**Software Construction and Development Lab**

**Assignment # 01**

**Student Name:** Muhammad Sameer Khan  
**Student ID:** 65135  
**Instructor:** Engr. Muhammad Rayan Shaikh

**Question 01: SDLC Model Recommendation**

**Case 1: Real-Time Tracking System – *Logistics Company***

**Recommended SDLC Model: Agile**

**Why Agile?**

**1. Flexibility for Evolving Requirements**

The project should have the flexibility to adopt numerous updates as well as user-initiated modifications. Teams using Agile design their projects through brief development phases called sprints which let them enhance their work structures according to client feedback.

**2. Collaboration Across Borders**

Agile promotes complete teamwork among distributed global teams through its implementation of scheduled stand-ups and sprint evaluations and project review activities.

**3. Continuous Risk Management**

Each sprint delivers operational product versions through which Agile enables teams to recognize and solve problems at an early stage. When implemented correctly these practices lower the possibility of significant project failures appearing at late stages.

**4. Faster Time to Market**

An organization can introduce its Minimum Viable Product (MVP) through early deployment. Regular enhancement and addition of new features helps the business gain value earlier in the process.

**Case 2: Tax Calculation System – *Government Tax Authority***

**Recommended SDLC Model: Waterfall**

**Why Waterfall?**

**1. Clear, Fixed Requirements**

Waterfall succeeds for this development due to the stable requirements that developers confront before their work begins. The system requires finishing one stage before advancing to the next to guarantee all requirements fulfill upfront.

**2. Regulatory and Compliance Needs**

All government projects require complete documentation together with required approvals. Waterfall offers its strength in managing formal planning and documentation together with system reviews which suits regulatory projects.

**3. Predictable Timeline and Scope**

The project manager maintains full control of resources through scheduling as well as deadlines and deliverables execution during the linear approach. The careful nature of this approach stands vital in government work since any project delays or budget overruns would cause significant problems.

**4. Thorough Testing Before Release**

The delivery of a comprehensive tested system as a single release is achievable through Waterfall which proves suitable for critical financial systems that need complete error-free systems.

**Question 02: UML Diagrams for Smart City Traffic Control System**

An intelligent traffic control system requires clear understanding of all its interactions during the design stage. I will assume the lead role as system architect to direct all teams between AI engineers and infrastructure developers and city planners into an aligned structure for modular and scalable system development. The UML diagrams will receive the following order of importance for implementation:

**1. Use Case Diagram: High Priority**

**Why It's Important:**

A Use Case diagram serves as an essential tool to establish system functionality from user point of view. The diagram shows the UML system allows us to see which users like sensors and responders and city planners will utilize it along with its supported functions like traffic control adjustment and signal overrides.

**Why Prioritize It:**

* A Use Case diagram provides an essential first step for building because it presents a fundamental framework for understanding functional requirements to all stakeholders.
* The diagram simplifies technological and non-technical understanding of system boundaries so the city planners can match their vision with the design objectives.

**2. Sequence Diagrams: High Priority**

**Why It's Important:**

Real-time component interaction becomes possible to monitor because Sequence Diagrams serve as essential documentation tools. The system requires precise understanding of real-time information processing since it depends on sensor data together with dynamic traffic management for emergency response coordination.

**Why Prioritize It:**

* Sequence Diagrams display both critical decision processes between system elements such as AI signal signal control based on input sensors along with emergency vehicle protocols.
* AI engineers and backend developers will gain benefit from using this document for understanding both the logic flow and implementation of correct component interactions.

**3. Component Diagram: Medium-High Priority**

**Why It's Important:**

Through the Component Diagram the system gets segmented into its main functional components. The system includes three main functional modules: Data Collection Module and AI Decision-Making Engine and Emergency Override System. Each modular development component needs proper understanding of both component structure and functional interaction.

**Why Prioritize It:**

* The diagram demonstrates the complete system structure thus each development team receives precise directions about their assigned system areas.
* The diagram benefits development stages while enabling scalability improvements because it allows each part to be developed and tested as independent units.

**4. Deployment Diagram: Medium-High Priority**

**Why It's Important:**

The physical deployment of traffic sensors cameras and traffic lights appears in the Deployment Diagram of the system as it displays both local server locations and cloud distribution. The deployment diagram holds special importance during operations of internet-connected systems.

**Why Prioritize It:**

* A proper diagram implementation enables infrastructure managers to grasp how system components should be physically set up (e.g., in cloud or traffic controllers and beyond).
* The system analysis phase at this stage identifies limitations or bottlenecks in infrastructure which leads to early detection of potential problems that will make the system sturdy and scalable.

**5. Activity Diagram: Optional but Valuable**

**Why It's Helpful:**

Activity Diagram deploys workflow methods to display decision-making processes together with parallel actions. Sequence Diagram provides information regarding interaction ordering. The Activity Diagram promotes understanding of how systems calculate congestion predictions and manage traffic light emergency control procedures throughout time.

**Why Prioritize It (Optional):**

* Decision clarity is obtained through the use of these diagrams when dealing with complex systems that include AI-driven logic.
* The documentation approach assists in documenting both rare situations and parallel operations which AI systems need to process concurrently.

**Question 04: Centralized Hotel Booking System (For ODD Roll Numbers)**

The design of a centralized booking system for global luxury hotels requires essential requirements that involve lot traffic management and data consistency maintenance and scalable security measures and independent front-end business logic for future enhancements. The lead developer must propose an operative architecture that effectively handles these requirements through modules and distributed networks with microservices and secure security implementations. My selected design architecture has the following background rationale:

**1. High Traffic Handling and Scalability**

**Design Approach: Microservices Architecture**

The hotel chain's worldwide growth requires an efficient system design that enables thousands of transactions per second performance particularly at peak tourism times. The recommended solution for this system is microservices architecture because a monolithic design struggles to accommodate growth requirements.

**How Microservices Help:**

* The application achieves distributed load capacity when broken down into four essential services (Room Management, Customer Management, Reservation Service, Payment Service) because each service operates independently for load handling.
* The system can increase its capacity by running services over multiple servers or containers which enables automatic resource distribution for traffic spikes.
* Multiple region deployments of microservices ensure both failover capabilities and redundancy through the same environment. System availability and reliability can be maintained through the capability of unaffected regions to assume control when any region experiences heavy traffic.

**2. Preventing Data Inconsistencies and Overbooking**

**Design Approach: Event-Driven Architecture with Distributed Databases**

During busy peak seasons the system needs to use an event-driven architecture and distributed database systems to prevent overbooking and keep real-time data consistency.

**How Event-Driven Architecture Helps:**

* The system updates itself in real time throughout all systems when customers perform any of these actions: room searches and booking modifications and reservations.
* The event queues paired with message brokers like Kafka and RabbitMQ enable proper sequence processing and access from various services. The Room Availability Service can instantly update the booking status for reservations to prevent booking overlaps during the confirmation process.

**How Distributed Databases Help:**

* The Global Data Synchronization subsystem makes use of distributed databases such as Cassandra or Google Spanner because it allows real-time synchronization of replicated data across multiple regions. A real-time consistency of room bookings persists throughout all locations through this protection scheme.
* Two-phase commits together with eventual consistency protocols function in regions that do not need real-time promises to deliver performance vs consistency flexibility.

**3. Secure Authentication and Data Protection**

**Design Approach: OAuth2, Role-Based Access Control (RBAC), and Data Encryption**

Security remains essential because the system treats all customer information and payment data along with booking details with utmost care.

**How OAuth2 Helps:**

* In order to provide secure user access to resources the standard authentication choice is OAuth2 Authentication. Through different OAuth2 authentication flows such as authorization code for users and client credentials for staff members the system enables secure token-based authentication for all parties including customers and hotel personnel and administrators.
* User authentication results in JWT tokens used by users to access data without frequent reentry of their credentials.

**How RBAC Helps:**

Every user group at the hotel will receive access permissions which match their assigned role. For example:

* Bookings service users get access to their booking details along with customer support applications through the system but nothing else.
* The check-in/check-out system and room management system will be accessible by staff users.
* Admin Access: Full access to financial reports, occupancy data, and promotions.
* Through RBAC only users who have authorization rights are able to complete specific system actions therefore preventing unauthorized entry.

**Data Encryption and PCI DSS Compliance:**

* All encoding processes will cover both stored and communicated sensitive data with encryption protocols (TLS/SSL will ensure security for data sent across networks).
* The system will maintain PCI DSS standard compliance to secure financial transactions along with payment data protection via compliance with PCI DSS regulations since this is the highest possible standard.

**4. Independent Business Logic and Front-End Flexibility**

**Design Approach: Separation of Concerns with RESTful APIs**

A business-independent front end requires fundamental concern separation to maintain logical independence.

**How APIs Help:**

* The system will show all business data and business logic through RESTful APIs. The APIs will manage operations that involve hotel searches in addition to reservation check status as well as payment processing workflows.
* All APIs provided by the system will be accessible through its front-end interface which can be either web or mobile. Any update or redesign of the front-end does not interferewith business logic or backend functionality because of this separation methodology.
* The system will deploy microservices to create separate functional units that handle particular functions (such as reservation support and payment processing). The front-end can develop independently because it only needs to invoke required services via API.

**5. Long-Term Growth and Maintainability**

**Design Approach: Continuous Integration and Deployment (CI/CD), Monitoring, and Logging**

**How CI/CD Helps:**

The system deployment will rely on a comprehensive CI/CD pipeline which enables effortless system updates. Updates and new features developers push through the system will not disrupt service functions nor affect what is currently operating.

**How Monitoring Helps:**

The system health along with its performance in real time can be monitored through Prometheus or Grafana tools. The tools enable programmers to view peak visitor times alongside the detection of performance bottlenecks to quickly fix problems during periods when customers remain unaffected.

**How Logging Helps:**

A centralized logging system using ELK Stack will capture every system event along with errors and user actions which ensures easier user activity tracking for debugging and monitoring during security incidents and system failure situations.

**Question 05: Refactored Python Code**

* **By using python built-in sorting function:**

nums = [5, 2, 9, 1, 5, 6]

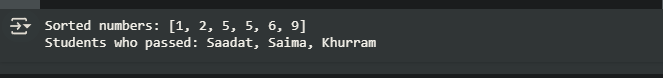
nums = sorted(nums)

student\_data = [("Saadat", 85), ("Areeb", 42), ("Saima", 91), ("Khurram", 75)]

passed\_students = [student[0] for student in student\_data if student[1] >= 50]

print("Sorted numbers:", nums)

print("Students who passed:", ", ".join(passed\_students))



* **Optimizing without built-in function:**

def bubble\_sort(nums):

    n = len(nums)

    for i in range(n):

        for j in range(0, n - i - 1):

            if nums[j] > nums[j + 1]:

                nums[j], nums[j + 1] = nums[j + 1], nums[j]

    return nums

def get\_passed\_students(student\_data):

    return [name for name, score in student\_data if score >= 50]

nums = [5, 2, 9, 1, 5, 6]

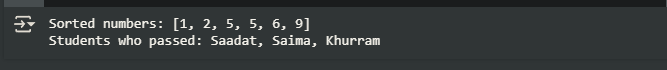
student\_data = [("Saadat", 85), ("Areeb", 42), ("Saima", 91), ("Khurram", 75)]

sorted\_nums = bubble\_sort(nums)

passed\_students = get\_passed\_students(student\_data)

print("Sorted numbers:", sorted\_nums)

print("Students who passed:", ", ".join(passed\_students))

****