Student ID No: \_\_\_\_\_\_\_\_\_\_Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_

Course & Year/Section: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Score: \_\_\_\_\_\_\_\_\_\_\_\_

**PERFORMANCE TASK 6**

**HIGH-LEVEL ARCHITECTURE DIAGRAM WITH INTEGRATION PATTERNS**

**General Directions**

* Read and follow each step carefully.
* Document your answers, screenshots, and reflections in your lab report.

**Objectives:**

By the end of this lab, students should be able to:

1. Identify and apply common system integration patterns (P2P, Hub-Spoke, Pub-Sub).
2. Develop a high-level architecture diagram for their proposed system project.
3. Demonstrate how modules interact and exchange data using appropriate integration patterns.
4. Document the rationale for selecting a specific integration pattern for their project.

**Tools/Resources Needed**

* Diagramming tool: Draw.io, Lucidchart, Miro, MS Visio (or equivalent)
* Project repository from Weeks 1 & 2
* Microsoft Word for documentation

**Lab Instructions**

**Step 1: Review Previous Work**

* Open your ProjectOverview.md and Context Diagram from Weeks 1 & 2.
* Identify the key modules and external systems/entities.

**Key Modules**

1. **Reservation Module** – for booking classrooms, labs, and offices.
2. **Equipment Module** – for reserving and tracking equipment (with QR/Barcode support).
3. **Maintenance Module** – for reporting, assigning, and updating maintenance requests.
4. **User Management Module** – for role-based accounts (faculty, students, admins, auxiliary staff).
5. **Admin Dashboard / Reporting Module** – for utilization reports, approvals, and monitoring system performance.

**External Systems / Entities**

1. **Faculty & Staff** – request facility and equipment reservations.
2. **Students** – limited access for certain reservations.
3. **Department Admins** – approve/monitor reservations, request maintenance.
4. **Auxiliary Staff** – receive and process maintenance/repair tasks.
5. **School Chairperson/Dean’s** – oversees all activities, final approval, and overall system management.
6. **Campus Single Sign-On**– external authentication service integrated into the system.

**Step 2: Select an Integration Pattern**

* Choose one integration pattern (P2P, Hub-Spoke, or Pub-Sub) that best fits your project.
* Justify your choice in a short-written explanation.

**Chosen Pattern:** **Hub-Spoke**

**Justification**

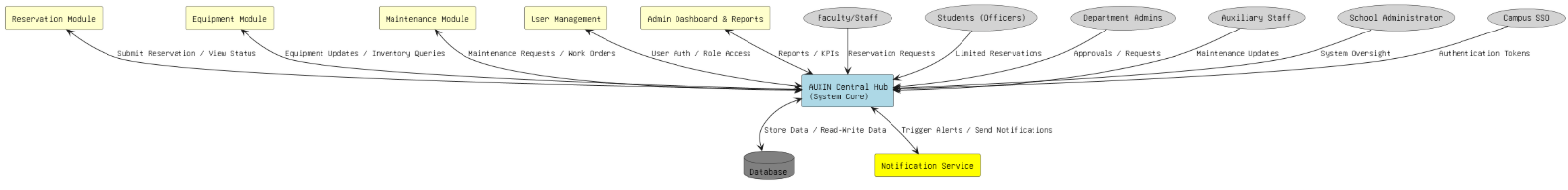
The Hub-Spoke integration pattern is the best fit for the AUXIN system because it provides a centralized hub that manages all communication between modules. Instead of each module directly connecting to every other module (as in P2P), all interactions flow through the hub.

* The Reservation Module, Equipment Module, and Maintenance Module only need to communicate with the AUXIN Hub, which simplifies the architecture.
* The Admin Dashboard retrieves consolidated data from the hub, while User Management and SSO authentication are also integrated at the hub level.
* This reduces system complexity, prevents data silos, and ensures better scalability and maintainability.
* It also helps with monitoring and security, since all requests and updates pass through a single point of control.

In short, the Hub-Spoke model ensures AUXIN can efficiently coordinate reservations, maintenance, and equipment tracking without modules becoming overly dependent on each other.

**Step 3: Create High-Level Architecture Diagram**

* Draw your system modules as components.
* Show communication links based on the chosen integration pattern.
  + P2P: Each module directly connects to others.
  + Hub-Spoke: A central hub routes communication.
  + Pub-Sub: A message broker handles publish/subscribe message.
  + Annotate each connection with type of communication (e.g., API call, database query, message queue).



**Step 4: Documentation**

Inside /docs, update ProjectOverview.md with a new section:

* Integration Pattern Applied – Name the pattern.
* Rationale – Explain why it fits your system.
* Diagram Reference – Insert your architecture diagram file (e.g., HighLevelArch.png).

**Integration Pattern Applied**

* Hub-Spoke Pattern

**Rationale**

The Hub-Spoke integration pattern was chosen for AUXIN because it provides a centralized hub that manages all communication between modules and external entities. This reduces complexity compared to a Point-to-Point (P2P) model, where each module must connect directly to others.

**By routing all interactions through the hub:**

**Scalability is improved** — new modules can be added without reconfiguring all connections.

**Maintainability is simplified** — changes in one module do not directly affect others.

**Reliability and Monitoring are enhanced** — the hub logs and manages all requests, making the system easier to monitor.

**Security is centralized** — all authentication and authorization requests pass through one controlled point.

This approach ensures that Reservation, Equipment, Maintenance, User Management, and Dashboard modules can communicate seamlessly with each other, while external entities (faculty, students, admins, auxiliary staff, and SSO) interact through the hub.

**Step 5: Version Control & Submission**

* Save your diagram as HighLevelArch.png in /docs.

**Deliverables**

1. High-Level Architecture Diagram (HighLevelArch.png in /docs)
2. Updated ProjectOverview.md with Integration Pattern & Rationale

**Grading Rubric for Lab Activity**

| **Criteria** | **Excellent (5)** | **Good (4)** | **Satisfactory (3)** | **Needs Improvement (2-1)** | **Weight** |
| --- | --- | --- | --- | --- | --- |
| Diagram Clarity | Well-structured, correct use of chosen pattern, clear labels | Mostly clear, minor missing details | Understandable but lacks clarity | Confusing or incorrect | 40% |
| Pattern Application | Correctly applies chosen pattern with justification | Mostly correct, justification minor | Pattern applied but weak explanation | Incorrect application or no justification | 30% |
| Documentation | Complete, well-written rationale & references | Mostly complete, minor gaps | Basic documentation only | Incomplete or missing | 20% |
| Repository Usage | Properly updated, file organized in /docs | Updated but disorganized | Incomplete updates | No repository update | 10% |

**Total: 100%**

**STUDENT WORKSHEET**

**Part 1 – System Information**

* Team Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* System Title: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* Selected Integration Pattern: (P2P / Hub-Spoke / Pub-Sub)🡨CHOOSE ONE ONLY

**Part 2 – Architecture Diagram**

* Insert or draw your High-Level Architecture Diagram here.

*(Screenshot or image should be saved as ArchitectureDiagram.png in the /docs folder of your repository.)*

A diagram of a payment system

AI-generated content may be incorrect.

*(Example Diagram if P2P is used)*

A diagram of a message broker

AI-generated content may be incorrect.

*(Example Diagram if Pub-Sub Integration is used)*

**Part 3 – Module Description**

* Fill in the table with at least 3 modules in your system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Module Name** | **Function/Responsibility** | **Interfaces (APIs, Middleware, etc.)** | **Integration Notes** |
| *Example* |  |  |  |
| Borrower Module | Handles borrower registration and profile management | REST API, DB | Sends borrower data to Loan Module via Hub |
| Loan Module | Manages loan applications, approvals, and records | REST API, DB | Receives borrower info, sends loan approval requests to Payment |
| Payment Module | Tracks payments and schedules | REST API | Communicates with Loan Module via Hub for updates |

**Part 4 – Data Flow Narrative**

* Write a short narrative (5–7 sentences) explaining how data flows between modules and external entities in your system under the chosen integration pattern.

*(Example Narrative)*

In the Lending System, the Hub-Spoke pattern ensures all communication passes through a central hub. When a borrower applies for a loan, the Borrower Module sends the request to the Hub, which forwards it to the Loan Module. Once the loan is approved, the Hub routes the approval details to the Payment Module to set up a repayment schedule. The Payment Module can also notify the Hub of missed or completed payments, which are relayed back to the Loan Module for record updates. This approach reduces direct dependencies between modules, making the system easier to maintain and scale.