Here is a step-by-step implementation process along with example code snippets to guide you through. This setup uses Python (Django/Flask for the API), Swagger for documentation, and Docker for containerization.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A white rectangular object with black text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A white rectangular object with a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Based on the requirements document and its context, here’s a structured **design** and a list of potential APIs that can be created, along with the necessary details for implementation.

**System Design**

**Objective**

The system facilitates device maintenance management, allowing:

1. Clients to check device status.
2. Maintenance tasks to be logged.
3. Compliance checks and activity tracking.

**Key Components**

1. **Authentication System**:
   * Authenticate the client using hardcoded values or token-based mechanisms initially.
   * Implement extensible authentication for future scalability.
2. **Device Management**:
   * APIs to query, create, update, and delete device records.
   * Ensure no conflicting activities occur on a single device simultaneously.
3. **Activity Tracking**:
   * Log and monitor maintenance tasks on devices.
   * Record actions for compliance reporting.
4. **Monitoring and Logging**:
   * Use ELK Stack for activity and error logging.
5. **Scalability**:
   * Future-proofing by defining clear payload structures and reusable modules.

**API List**

**1. Authentication**

**Purpose**: Authenticate the client and issue an access token.

* **Endpoint**: /auth/login
* **Method**: POST

A screenshot of a computer

Description automatically generated

* **Notes**:
  + Initial version can use hardcoded credentials.
  + Later versions can use OAuth2 or JWT for better security.

**2. Query Device**

**Purpose**: Check the status of a specific device.

* **Endpoint**: /device/query
* **Method**: GET
* **Query Params**:
  + device\_id

A white rectangular object with a black border

Description automatically generated

**3. Create Maintenance Task**

**Purpose**: Log a maintenance activity for a device.

* **Endpoint**: /activity/create
* **Method**: POST

A screenshot of a computer

Description automatically generated

**4. Update Maintenance Task**

**Purpose**: Modify details of an existing task.

* **Endpoint**: /activity/update
* **Method**: PUT
* **Payload**:

json

Copy code

{

"activity\_id": "ACT12345",

"status": "completed"

}

* **Response**:

json

Copy code

{

"message": "Activity updated successfully"

}

**5. Delete Maintenance Task**

**Purpose**: Remove a maintenance record.

* **Endpoint**: /activity/delete
* **Method**: DELETE
* **Payload**:

json

Copy code

{

"activity\_id": "ACT12345"

}

* **Response**:

json

Copy code

{

"message": "Activity deleted successfully"

}

**6. Get Device Details**

**Purpose**: Retrieve information about a device.

* **Endpoint**: /device/details
* **Method**: GET
* **Query Params**:
  + device\_id
* **Response**:

json

Copy code

{

"device\_id": "ABCD",

"status": "available",

"last\_maintenance": "2024-11-20T15:30:00Z"

}

**7. Get Activity Logs**

**Purpose**: Fetch logs for compliance or monitoring.

* **Endpoint**: /logs/activities
* **Method**: GET
* **Query Params**:
  + device\_id
  + date\_range
* **Response**:

json

Copy code

[

{

"activity\_id": "ACT12345",

"device\_id": "ABCD",

"activity": "maintenance",

"status": "completed",

"timestamp": "2024-11-20T15:30:00Z"

}

]

**8. Device Onboarding**

**Purpose**: Add a new device to the system.

* **Endpoint**: /device/add
* **Method**: POST
* **Payload**:

json

Copy code

{

"device\_id": "XYZ123",

"description": "New wireless router",

"status": "available"

}

* **Response**:

json

Copy code

{

"message": "Device onboarded successfully"

}

**9. Device Deletion**

**Purpose**: Remove a device from the system.

* **Endpoint**: /device/delete
* **Method**: DELETE
* **Payload**:

json

Copy code

{

"device\_id": "XYZ123"

}

* **Response**:

json

Copy code

{

"message": "Device deleted successfully"

}

**10. System Health Check**

**Purpose**: Ensure the service is running.

* **Endpoint**: /health
* **Method**: GET
* **Response**:

json

Copy code

{

"status": "OK"

}

**11. Compliance Reporting**

**Purpose**: Generate a compliance report.

* **Endpoint**: /report/compliance
* **Method**: POST
* **Payload**:

json

Copy code

{

"start\_date": "2024-11-01",

"end\_date": "2024-11-25"

}

* **Response**:

json

Copy code

{

"report\_url": "http://reporting-system-url/report123.pdf"

}

**API Security and Requirements**

1. **Authentication**:
   * Use JWT tokens issued by the /auth/login endpoint.
   * Include the token in the Authorization header for secured endpoints.
2. **Logging**:
   * Log all requests and responses to the ELK Stack.
   * Include metadata (timestamp, client\_id, etc.) in logs.
3. **Error Handling**:
   * Implement consistent error responses:

json

Copy code

{

"error": "Invalid request",

"details": "Device ID not found"

}

1. **Validation**:
   * Validate all payloads using a library like Pydantic or Marshmallow.
2. **Extensibility**:
   * Design modular endpoints to add functionality easily.