**LOVELY PROFESSIONAL UNIVERSITY**

**Academic Task-2**

**1. Project Overview**

The Live Process Monitoring Dashboard is a web-based application that collects and displays real-time data for a variety of processes. It displays system metrics in an interactive and visual format, including CPU utilization, memory consumption, network activity, and other vital performance indicators. The purpose is to allow administrators and users to efficiently monitor live operations, receive notifications for abnormalities, and analyze patterns over time.

**Expected Outcomes**

A real-time dashboard that monitors active system processes.  
  
Interactive data visualization improves insights.  
  
Alerts and notifications for threshold breaches.  
  
Historical data collection for analysis and decision-making.

**Scope**

The project will focus on real-time monitoring of system processes and data visualization. The dashboard will support integration with various data sources, including APIs, databases, and IoT devices.

**2. Module-Wise Breakdown**

The project consists of three core modules:

**1. Data Collection & Processing Module**

**Purpose:** Collects real-time system data and processes it for visualization.

**Roles:**

Retrieves live system metrics (CPU, memory, network, etc.).  
  
Stores historical data for analysis.  
  
Detects irregularities and sends alerts.

**2. Data Visualization Module**

**Purpose:** Converts raw data into meaningful visual representations.

**Roles:**

Shows real-time graphs and charts.  
  
Allows for filtering and historical comparison.  
  
Creates user-friendly UI components.

**3. User Interface & Alerting Module**

**Purpose:** Allows users to interact with the dashboard and receive notifications. **Roles:**

Provides an easy-to-use monitoring interface.

Implement alert systems (email, SMS, and pop-ups).

Provides user authentication and role-based access control.

**3. Functionalities**

**Data Collection & Processing Module**

Obtain real-time system metrics.  
  
Data can be stored and retrieved for historical examination.  
  
Implement WebSocket’s to provide real-time updates.

**Data Visualization Module**

Interactive charts and graphs (CPU, memory, and network utilization).  
  
Customizable dashboards with multiple data views.

Heatmaps for anomaly detection.

**User Interface & Alerting Module**

User-friendly UI with a responsive design.  
  
Alert system for threshold crossings.  
  
Role-based access control and authentication.

**4. Technology Used**

**Programming Languages:**

* Python (Flask/Django) for backend.
* JavaScript (React.js) for frontend.

**Libraries and Tools:**

* Flask-SocketIO for real-time updates.
* PostgreSQL for structured data storage.
* Chart.js/D3.js for data visualization.

**Other Tools:**

* GitHub for version control.
* Postman for API testing.
* Docker for deployment.
* **5. Flow Diagram**