CA-2 | CSR210 | LPU | Advanced Programming and Database Systems

Continuous Assessment Question Paper

Subject: Advanced Programming and Database Systems

Units Covered: Unit-1: REST API Development with FastAPI, Unit-2: Dynamic Web Application Development with Flask, Unit-3: Integrating databases with SQLAlchemy and PostgreSQL, Unit-4: NoSQL databases with MongoDB and Cassandra, Unit-5: Building analytics dashboards using 'plotly' and 'dash'.

Total Marks: 50

ASSIGNMENT OBJECTIVE

Design and implement a full-stack, data-driven dashboard application that demonstrates comprehensive integration of backend APIs, database management, and interactive visualization. The application must feature:

1. RESTful APIs enabling real-time data manipulation (JWT authentication optional)
2. An analytics dashboard that dynamically reflects database changes
3. Local deployment (cloud deployment optional)

Recommended Application Themes

Students may select one of the following domains or propose an alternative theme with instructor approval:

- Personal Finance Tracker
- University Courses Tracker
- Student Performance Tracker
- Health & Fitness Monitor
- Social Media Sentiment Explorer
- E-commerce Sales Dashboard
- Library Inventory System
- Student Attendance Tracker

TECHNICAL REQUIREMENTS

1. Backend Implementation

- Develop using either FastAPI or Flask
- Implement CRUD operations via RESTful endpoints
- Include proper error handling and HTTP status codes
- Database integration via SQLAlchemy (PostgreSQL) or native drivers (MongoDB)

2. Database Layer

- Use either PostgreSQL (relational) or MongoDB (NoSQL)
- Design normalized schema (SQL) or appropriate document structure (NoSQL)
- Include sample dataset with at least 50 records

3. Frontend/Dashboard

- Create interactive visualizations using Plotly and Dash
- Minimum of three distinct visualization types (bar, line, pie, scatter, etc.)
- Implement filtering/querying capabilities
- Ensure responsive design principles

4. System Integration

- APIs must modify database records
- Dashboard must reflect changes without manual refresh
- Clear separation of concerns between components

DEMONSTRATION PROTOCOL

Each student will present their project in 5 minutes following this sequence:

Step 1: Repository Overview (1 minute)

- Share GitHub repository link
- Highlight README.md with:
    - Project description
    - Technology stack
    - Setup instructions
    - API documentation

Step 2: System Architecture (1 minute)

- Explain chosen architecture pattern
- Diagram component interactions
- Justify technology selections

Step 3: Live Demonstration (2 minutes)

- Show pre-loaded dashboard with sample data
- Execute ADD operation via API (Postman/curl/custom interface)
- Demonstrate real-time update in dashboard visualizations
- Highlight additional CRUD operations

Step 4: Video Presentation (1 minute)

- Play pre-recorded 60-90 second explainer video covering:
    - Application features
    - Key technical implementations
    - User workflow

EVALUATION RUBRIC

Criteria	Maximum Marks	Assessment Parameters
Framework & Concepts	15	• Appropriate use of FastAPI/Flask features • Database design and optimization • Code organization and modularity • Error handling and validation
Visualization Quality	10	• Relevance of visualizations to data • Interactivity and user experience • Aesthetic design and clarity • Information density and readability
Demonstration	15	• System functionality and completeness • Real-time update demonstration • Presentation clarity and timing • Technical justification of choices

GitHub Repository	5	• Code organization and documentation • Commit history and version control • README completeness and clarity • Setup reproducibility
Explainer Video	5	• Content coverage and conciseness • Production quality and clarity • Demonstration of key features • Professional presentation
Total | 50 |		