# **Key Account Manager (KAM) Lead Management System**

[Key Account Manager (KAM) Lead Management System](#_atah7t5px16m)

[Introduction](#_khly0hqokiy)

[Requirements](#_auogxvsfs8qb)

[Core Requirements](#_3yh9514klop2)

[Technical Requirements](#_jarrx1xidbcy)

[Implementation Guide](#_gdvkw4uxyv0y)

[System Design Questions](#_qs5qwjj37s4l)

[Data Modeling Questions](#_3n7xe8uepcam)

[Implementation Details](#_56cfa4mvwgfc)

[Edge Cases](#_rv3zgyfmjlpr)

[Submission Guidelines](#_kq6wm9ny58zh)

[Code Requirements](#_5qj4zvoxqhl1)

[Documentation Requirements](#_rajj90yfl35t)

[Video Demonstration](#_dxwqoli7k2q7)

[Submission Format](#_ntr1zbwgccw0)

[Evaluation Criteria](#_n97ui5yfwebz)

[Bonus Features](#_ub0r0w1cs99c)

[Project Timeline](#_kc8dtpmyhgms)

[Submission Process](#_ucukgs19e4ag)

[Important Notes](#_25ul58euzyti)

## **Introduction**

Udaan, a B2B e-commerce platform, requires a Lead Management System for Key Account Managers (KAMs) who manage relationships with large restaurant accounts. This system will help track and manage leads, interactions, and account performance.

## **Requirements**

### Core Requirements

1. Lead Management
   * Add new restaurant leads
   * Store basic restaurant information
   * Track lead status
2. Contact Management
   * Multiple Points of Contact (POCs) per restaurant
   * Store contact details (name, role, contact information)
   * Support multiple POCs with different roles
3. Interaction Tracking
   * Record all calls made to leads
   * Track orders placed
   * Store interaction dates and details
4. Call Planning
   * Set call frequency for each lead
   * Display leads requiring calls today
   * Track last call made
5. Performance Tracking
   * Track well-performing accounts
   * Monitor ordering patterns and frequency
   * Identify underperforming accounts

### Technical Requirements

1. Data Models
   * Database schema/data structures design
   * Entity relationship management
   * Efficient querying capabilities
2. API Design
   * RESTful APIs for all operations
   * Error handling
   * Authentication and authorization
3. Business Logic
   * Logic for determining today's calls
   * Account performance metrics calculation
   * Lead status transition handling

## **Implementation Guide**

### System Design Questions

1. How would you design the overall architecture?
2. What database(s) would you choose and why?
3. How would you ensure scalability?

### Data Modeling Questions

1. Design the database schema or class structure
2. How would you handle relationships between entities?
3. What indexes would you create for optimal performance?

## **Implementation Details**

Write code that implements above requirements.

### Edge Cases

1. How would you handle change in KAM?
2. How would you handle timezone differences for call scheduling?

## **Submission Guidelines**

### Code Requirements

1. Working condition code
2. Clear dependency specification
3. Sample data inclusion

### Documentation Requirements

README.md file containing:

1. Project overview
2. System requirements
3. Installation instructions
4. Running instructions
5. Test execution guide
6. API documentation
7. Sample usage examples

### Video Demonstration

5-10 minute video showing:

1. Code setup process
2. Application running
3. Major features demonstration
4. Sample inputs/outputs

* MP4 format
* Accessible location link in README

### Submission Format

ZIP file containing:

1. Source code directory
2. README.md
3. requirements.txt or package.json
4. Demonstration video

## **Evaluation Criteria**

1. Modularity
   * Separation of concerns
   * Design pattern usage
   * Component reusability
   * Well-defined interfaces
2. Extensibility
   * New feature addition ease
   * Component configurability
   * Interface/abstract class usage
3. Completeness
   * All required features
   * Error handling
   * Edge case coverage
   * Comprehensive testing
4. Code Readability
   * Clear naming conventions
   * Proper documentation
   * Consistent coding style
   * Appropriate comments

### Bonus Features

1. Unit Testing
   * Comprehensive test coverage
   * Scenario and edge case testing
   * Integration tests
   * Test documentation
2. REST API Implementation
   * RESTful endpoint design
   * API documentation
   * HTTP status codes
   * Request/Response validation
   * Authentication/Authorization
3. Functional User Interface
   * Clean, intuitive design
   * Responsive layout
   * Interactive features
   * Error handling
   * Cross-browser compatibility
4. Deployment
   * Online accessibility
   * Deployment documentation
   * Environment configuration
   * CI/CD pipeline

## **Project Timeline**

* Submission Deadline: January 5th, 2025
* Early submissions encouraged
* Rolling basis evaluation

## **Submission Process**

1. Complete implementation
2. Record demonstration video
3. Create ZIP file with all components
4. Verify all requirements
5. Submit via email

## **Important Notes**

1. Incomplete submissions not considered
2. Original work required
3. Third-party libraries allowed with justification
4. Plagiarism results in disqualification