Core Objectives

1. **Statistics Management System (SMS):**
    - Store and manage statistical data related to baggage handling and operations.
    - Generate and manage daily, weekly, monthly, and annual reports.
    - Provide a user-friendly interface for operators to view, edit, and export data.
2. **Information Management System (IMS):**
    - Manage and display flight schedules, baggage data, and operational instructions dynamically.
    - Interface with external systems for data exchange and synchronization.
    - Operate 24/7 with redundancy for reliability.

Key Functional Requirements

Statistics Management System (SMS)

- **Data Handling and Storage:**
    - Maintain large-capacity databases for handling baggage statistics and operational data.
    - Automate data compilation and formatting for reports (e.g., baggage counts, flow rates).
- **Reporting:**
    - Generate customizable reports for baggage statistics (daily, weekly, monthly, annually).
    - Ensure easy formatting and printing for various operational needs.

Information Management System (IMS)

- **Flight Management:**
    - Handle flight schedules (arrival, departure, and transfer) dynamically.
    - Support manual and automatic data updates for flight allocation.
- **Baggage Management:**
    - Track and allocate baggage based on flight data.
    - Interface with systems like FIDS, CUPPS, and security for real-time tracking.
- **System Monitoring:**
    - Provide operational status and fault information for the BHS.
    - Offer redundancy and fault-tolerant design to ensure uninterrupted operation.
- **Interfaces with External Systems:**
    - Integrate with multiple external systems (e.g., FIS, security, fire alarm, BMS, CCTV).

Implementation Approach

1. Database Design

- **Database Type:** Use a local relational database management system (RDBMS) like **PostgreSQL** or **SQLite** for data storage.
- **Structure:**
    - **Baggage Statistics Table:** Track counts, flow rates, and operational data.
    - **Flight Schedule Table:** Store flight and airline information.
    - **Baggage Tracking Table:** Map baggage ID to flight details and sorting destinations.

2. User Interface (UI)

- **Framework:** Develop the UI using a desktop application framework like **Electron.js**, **Qt**, or **Tkinter**.
- **Features:**
    - Data input and editing forms.
    - Real-time operational dashboards for monitoring.
    - Report generation and export tools (e.g., CSV, PDF).

3. Redundancy

- Implement a dual-system setup:
    - Active server for real-time operations.
    - Standby server for failover.

4. External System Integration

- Develop APIs or middleware to interface with external systems.
- Ensure real-time communication for flight and baggage updates.

5. Security and Redundancy

- Implement user authentication and role-based access control.
- Ensure local backups and redundancy to prevent data loss.

Technology Stack

- **Backend:** Python (Django/Flask) or Java (Spring Boot).
- **Frontend:** HTML/CSS/JavaScript or Python-based GUI frameworks.
- **Database:** PostgreSQL or SQLite (local setup).
- **Communication:** RESTful APIs for system integration.
- **Reporting Tools:** Pandas (Python) or JasperReports.

Project Phases

1. **Requirement Analysis:** Finalize data structure, reporting needs, and system integration points.
2. **Database Development:** Create schemas for SMS and IMS data.
3. **UI Development:** Build an operator-friendly interface for data interaction.
4. **Integration:** Establish connections with external systems (e.g., FIS, CUPPS).
5. **Testing:** Conduct functional, integration, and load testing.
6. **Deployment:** Set up the system in a local environment with redundancy.