Requirement Analysis Specification Document

Project Team

2025 年 4 月 5 日

目录

1 Document Overview

1.1 Document Purpose

This document is the requirement analysis specification document for the project, aiming to provide a detailed description of the system's functional requirements, non-functional requirements, use case analysis, and module interaction relationships, serving as the foundation for subsequent design and development work. It clearly defines user requirements and transforms these requirements into implementable technical specifications. Additionally, through use case diagrams, it describes the responsibilities of the System group in the project and its interaction relationships with other groups (UI group, Data group, and Analysis group), providing clear role definitions and collaboration guidelines for all project stakeholders.

1.2 Project Background

This project aims to develop a comprehensive system with four main components working together: the System group, User Interface group (UI), Data group, and Analysis group. These four groups each have their specific responsibilities, working together to build a complete solution to meet user needs and business objectives. The System group, as the technical infrastructure provider of the project, is the designer and implementer of the project's technical architecture, responsible for the development of core system functionalities.

1.3 Target Audience

This document is intended for the following readers:

- Project managers: To understand the overall project requirements and scope
- Development team members: To understand specific technical requirements and implementation details
- Testers: To develop test plans and test cases
- User representatives: To confirm whether the system meets business requirements

1.4 References

1. Claude 3.7
2. QwQ

2 System Overview

2.1 System Objectives

This system aims to [describe the main objectives and problems to be solved here]. By integrating user interface, system architecture, data processing, and analysis functions, it provides users with a comprehensive solution to improve work efficiency and decision-making quality.

2.2 System Scope

This system includes but is not limited to the following functional areas:

- User interface display and interaction
- System core architecture and services
- Data collection, storage, and management
- Data analysis and visualization

2.3 System Architecture Overview

The system consists of four main components with the following responsibilities:

- **System Group**: Responsible for core architecture design, API implementation, third-party service integration, system deployment, performance optimization, security assurance, and monitoring systems
- **User Interface Group (UI)**: Responsible for user interface design and implementation, user experience optimization, and front-end interaction with backend APIs
- **Data Group**: Responsible for data model design, database implementation, data processing workflows, and data quality assurance
- **Analysis Group**: Responsible for data analysis algorithms, report generation, data visualization, and decision support models

These components interact through well-defined interfaces, as shown in the following figure:

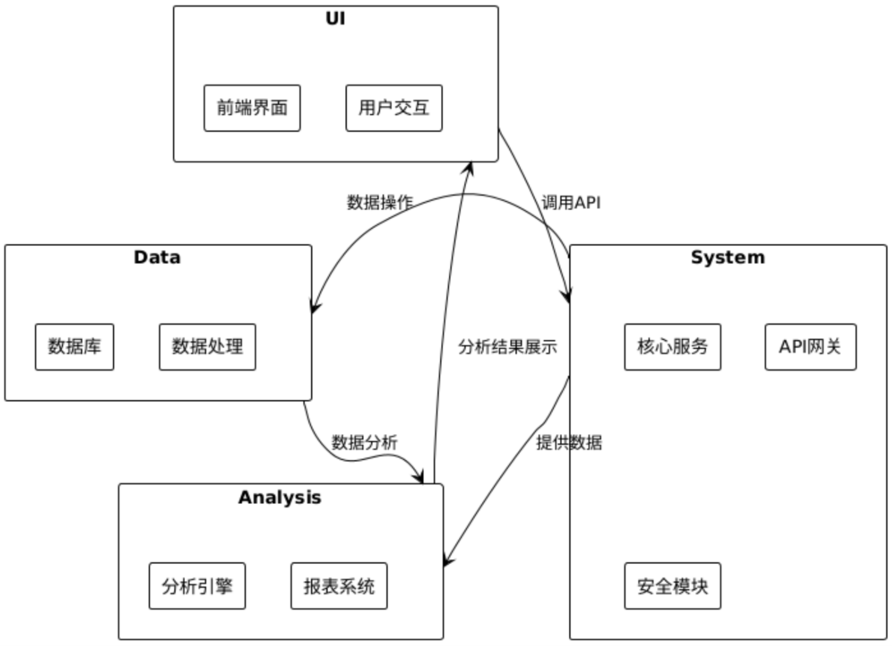

图 1: System Architecture Overview

3 User Requirements

3.1 User Role Definition

The system identifies the following main user roles:

1. **System Administrator**: Responsible for system management and maintenance
2. **Doctor User**: Uses the main functions of the system
3. **Patient User**: Uses some functions of the system

3.2 User Scenarios

The following is an overview of the main user scenarios:

1. User login and authentication
2. Data entry and editing
3. Data query and retrieval
4. Data analysis and report generation
5. System management and configuration
6. Security auditing and monitoring

4 Functional Requirements

4.1 UI Group Functional Requirements

1. **User Interface Design**
   - Design interface layouts that comply with user experience principles
   - Implement responsive design to adapt to different devices
   - Provide consistent visual style and component library
2. **User Interaction Implementation**
   - Implement basic functions such as user login, registration, and password reset
   - Provide personalization settings and theme customization
   - Implement multi-language support
3. **Frontend Data Display**
   - Implement data tables, charts, and other display components
   - Provide data filtering, sorting, and pagination functions
   - Support data export and printing

4.2 System Group Functional Requirements

The System group's main responsibility in the project is to serve as the technical infrastructure provider, responsible for the development of core system functionalities. The following are the specific functional requirements for the System group:

1. **Build and Deployment Process**

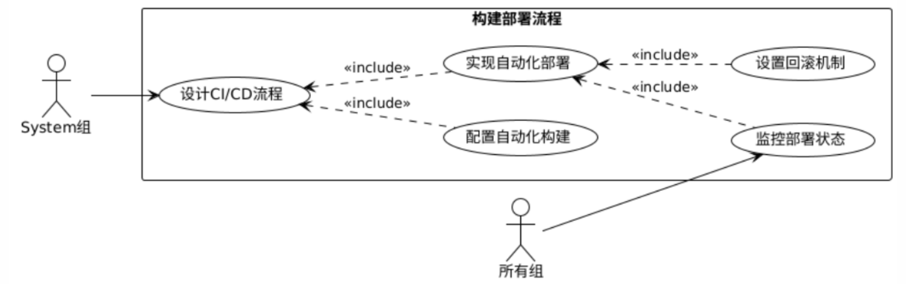

图 2: Build and Deployment Process Use Case Diagram

   - Design and implement complete CI/CD processes
   - Configure automated build and test environments

- Provide deployment status monitoring
- Establish rollback mechanisms for deployment failures

2. **Integrate Third-party Services**

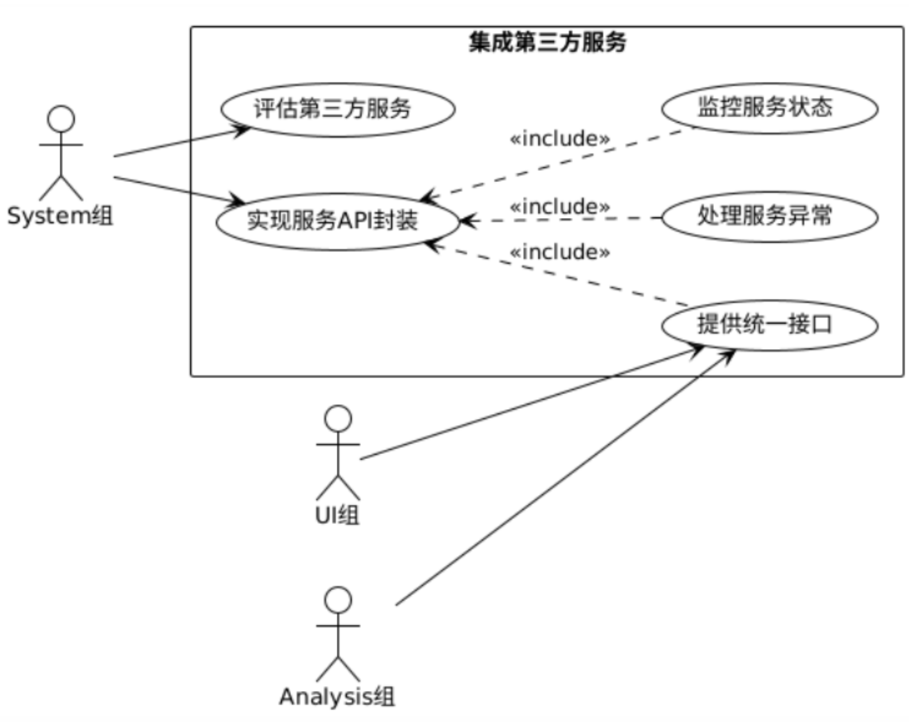

图 3: Third-party Service Integration Use Case Diagram

- Evaluate and select third-party services suitable for project requirements
- Implement encapsulation of third-party APIs
- Provide unified interface specifications
- Handle third-party service exceptions
- Monitor third-party service availability and performance

3. **Collaborative Database Development**

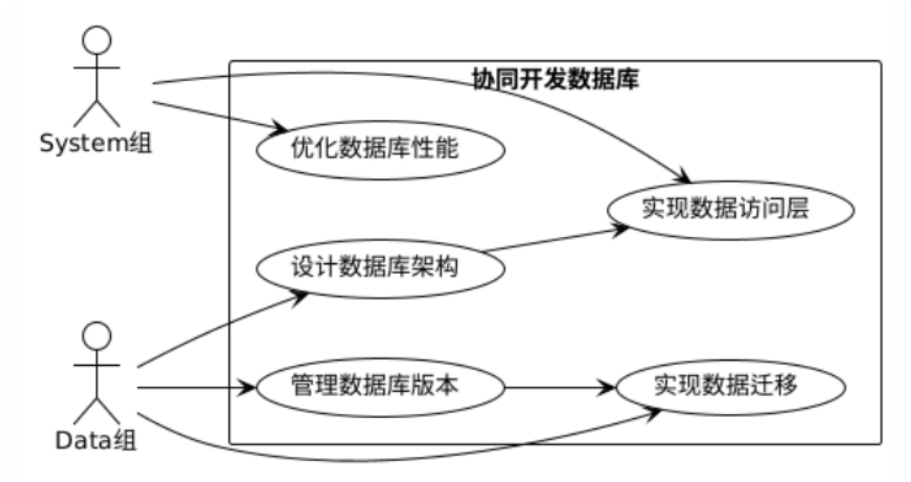

图 4: Collaborative Database Development Use Case Diagram

- Collaborate with the Data group to implement data access layer
- Optimize database performance
- Implement data caching mechanisms

4. **API Design and Implementation**

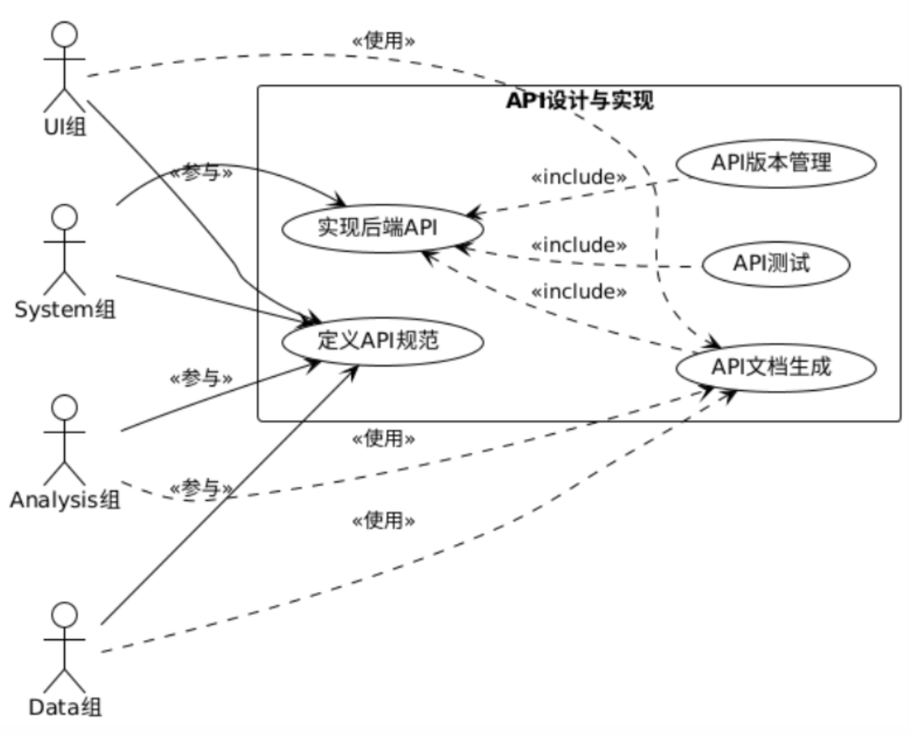

图 5: API Design and Implementation Use Case Diagram

- Develop API design standards and specifications
- Implement all backend API functionalities

- Generate complete API documentation
- Conduct API unit testing and integration testing
- Manage API versions and compatibility

5. **Performance Optimization**

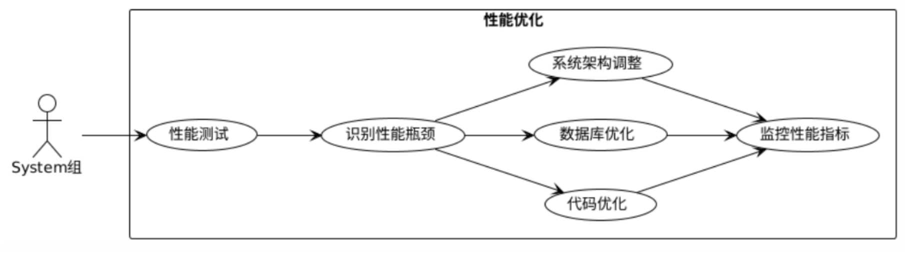

图 6: Performance Optimization Use Case Diagram

- Design and execute performance testing plans
- Analyze and identify system performance bottlenecks
- Optimize code execution efficiency
- Collaborate with the Data group on database performance optimization
- Adjust system architecture when necessary
- Continuously monitor system performance metrics

6. **Security Assurance**

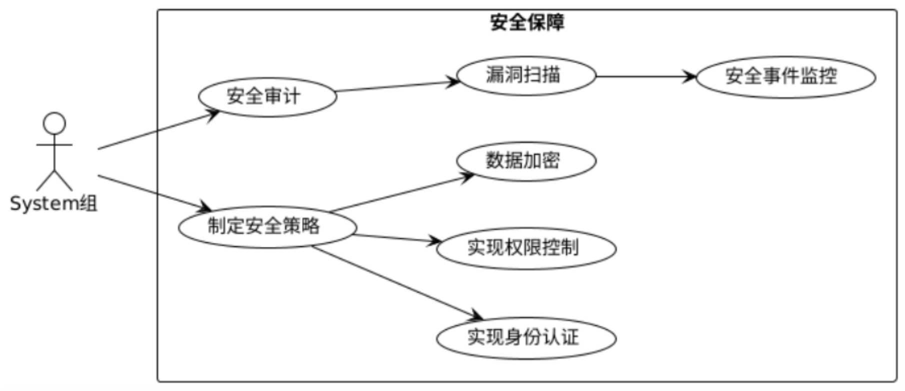

图 7: Security Assurance Use Case Diagram

- Develop system security policies and standards
- Implement identity authentication and authorization mechanisms
- Implement fine-grained permission control

- Encrypt sensitive data storage and transmission
- Prevent common security threats (SQL injection, XSS, etc.)
- Conduct regular security audits and vulnerability scanning
- Implement security event monitoring and response mechanisms

7. **Monitoring and Logging System**

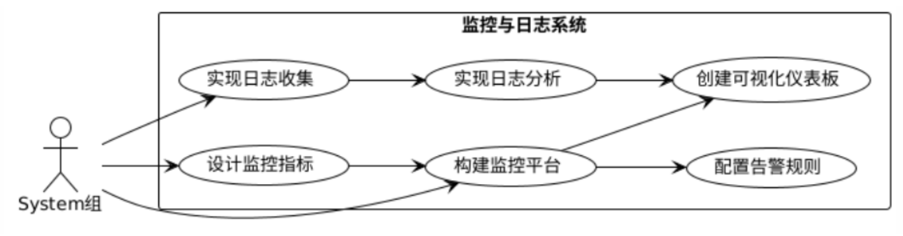

图 8: Monitoring and Logging System Use Case Diagram

- Design system key monitoring metrics
- Implement comprehensive log collection mechanisms
- Build unified monitoring platform
- Configure reasonable alert rules
- Implement log aggregation and analysis functions
- Create intuitive visualization dashboards

4.3 Data Group Functional Requirements

1. **Database Design**
   - Design database models and table structures
   - Define data constraints and relationships
   - Implement data version control
2. **Data Processing Workflow**
   - Implement data ETL (Extract, Transform, Load) processes
   - Provide data batch import and export functions
   - Implement data cleaning and validation mechanisms
3. **Data Quality Management**
   - Implement data quality check rules

- Monitor data integrity and consistency
- Provide data repair tools

4.4 Analysis Group Functional Requirements

1. **Data Analysis Algorithms**
    - Implement statistical analysis functions
    - Provide predictive analysis models
    - Support custom analysis rules
2. **Report Generation**
    - Design standard report templates
    - Support custom reports
    - Implement scheduled report generation and distribution
3. **Data Visualization**
    - Provide various charts and visualization components
    - Support interactive data exploration
    - Implement data drilling and aggregation analysis

5 Non-functional Requirements

5.1 Performance Requirements

1. **Response Time**
    - Page loading time should not exceed 3 seconds
    - API interface response time should not exceed 1 second
    - Report generation time should not exceed 5 seconds
2. **Concurrent Processing**
    - The system should support at least 100 concurrent users
    - The database should handle 50 concurrent transactions simultaneously

5.2 Reliability Requirements

1. **Data Backup and Recovery**
    - Perform daily data backups
    - Data recovery time should not exceed 4 hours
2. **Error Handling**
    - The system should provide clear error messages
    - Critical operations should provide rollback mechanisms

5.3 Security Requirements

1. **Identity Authentication and Authorization**
    - Implement multi-factor authentication
    - Role-based access control
    - Regular password update policies
2. **Data Security**
    - Encrypt sensitive data storage
    - Communication encryption (HTTPS)

5.4 Maintainability Requirements

1. **Modular Design**
    - The system should follow loose coupling and high cohesion design principles
    - Support independent module updates and deployment
2. **Complete Documentation**
    - Provide detailed system architecture documentation
    - Write complete API documentation
    - Maintain code comments and key algorithm descriptions
3. **Testability**
    - Support automated unit testing and integration testing
    - Provide testing environments and test data

6 Use Case Analysis

6.1 System Group Use Case Overview

The following use case diagram shows the System group's main responsibilities and interaction relationships with other groups:

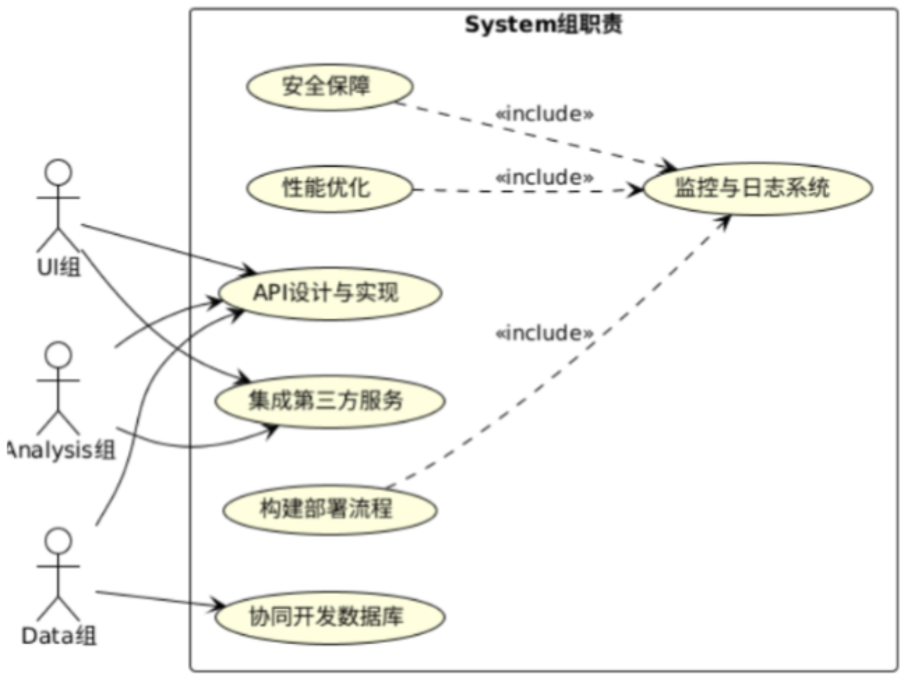

图 9: System Group Use Case Overview

6.2 System Group Use Case Analysis

6.2.1 Build and Deployment Process

Use Case Name: Build and Deployment Process
Actor: System Group
Preconditions:

- Project code repository has been established
- Deployment environment is ready

Postconditions:

- Automated CI/CD process can run normally
- All groups can use the deployment process to deploy code

Basic Flow:

1. System group designs CI/CD process
2. Configure automated build environment
3. Implement automated deployment mechanism
4. Set up deployment monitoring and rollback mechanism
5. All groups use the deployment process to deploy code

Alternative Flow:

1. When deployment fails, automatically trigger rollback mechanism
2. When environment is unavailable, provide manual deployment options

Deliverables:

- CI/CD configuration files
- Deployment scripts
- Deployment documentation

6.2.2 Integrate Third-party Services

Use Case Name: Integrate Third-party Services
Actor: System Group
Preconditions:

- Third-party services to be integrated have been identified
- Related API documentation and access permissions have been obtained

Postconditions:

- Third-party services are successfully integrated into the system
- Provide unified interfaces for other groups to use

Basic Flow:

1. Evaluate and select third-party services suitable for project requirements
2. Implement encapsulation of third-party APIs
3. Provide unified interface specifications

4. Handle third-party service exceptions
5. Monitor third-party service availability and performance

Alternative Flow:

1. When third-party services are unavailable, provide degradation strategies
2. When interfaces change, provide compatibility handling

Deliverables:

- Third-party service integration documentation
- API encapsulation code
- Service configuration guide

6.2.3 Collaborative Database Development

Use Case Name: Collaborative Database Development
Actor: System Group and Data Group
Preconditions:

- System requirements have been clarified
- Data requirements have been collected

Postconditions:

- Database architecture design is completed
- Data version management mechanism is established

Basic Flow:

1. Data group designs database architecture
2. System group implements data access layer
3. Data group manages database versions
4. Data group implements data migration solutions

Alternative Flow:

1. When architecture design has issues, System group proposes modification suggestions

2. When performance issues are serious, jointly redesign data models

Responsibility Division:

- System Group: Implement data access layer and database performance optimization
- Data Group: Responsible for database architecture design, version management, and data migration

Deliverables:

- Database design documentation
- Database version control scripts
- Data access layer code

6.2.4 API Design and Implementation

Use Case Name: API Design and Implementation
Actor: System Group
Preconditions:

- System functional requirements have been clarified
- Data models have been defined

Postconditions:

- API interface design is completed and implemented
- API documentation is generated and published
- API testing is passed

Basic Flow:

1. Develop API design standards and specifications
2. Implement all backend API functionalities
3. Generate complete API documentation
4. Conduct API unit testing and integration testing
5. Manage API versions and compatibility

Alternative Flow:

1. When requirements change, update API design and notify relevant parties
2. When security vulnerabilities are discovered, promptly fix and update

Deliverables:

- API design documentation
- API implementation code
- API test cases
- Swagger/OpenAPI documentation

6.2.5 Monitoring and Logging System

Use Case Name: Monitoring and Logging System
Actor: System Group
Preconditions:

- Basic system functions have been implemented
- Monitoring requirements and metrics have been determined

Postconditions:

- Monitoring and logging system is established and running
- Exception alert mechanism is effective

Basic Flow:

1. Design system key monitoring metrics
2. Implement comprehensive log collection mechanisms
3. Build unified monitoring platform
4. Configure reasonable alert rules
5. Implement log aggregation and analysis functions
6. Create intuitive visualization dashboards

Alternative Flow:

1. When log volume is too large, implement log grading and sampling strategies

2. When false alarms are too many, adjust alert thresholds and rules

Deliverables:

- Monitoring system configuration
- Log analysis platform
- Alert rule configuration
- Operation manual

6.3 Other Key Use Cases

This section supplements detailed descriptions of other key use cases as needed.

7 Data Requirements

7.1 Data Entities

The main data entities involved in the system include:

1. **User**: Stores system user information
2. **Role**: Defines user roles and permissions
3. **Business Data**: [Supplement according to specific project]
4. **Configuration Data**: System configuration and parameters
5. **Log Data**: System operation and event logs

7.2 Data Dictionary

The following is the field definition of the system's main data entities:

Entity	**Field**	**Type**	**Description**
User	id	Integer	User unique identifier
	username	String	Username
	password	String	Encrypted password
	email	String	User email
	status	Enum	User status (Active/Disabled)
Role	id	Integer	Role unique identifier

Entity	**Field**	**Type**	**Description**
	name	String	Role name
	permissions	String array	Permission list

Note: Supplement the complete data dictionary according to specific project requirements.

7.3 Data Flow

The main data flows of the system are as follows:

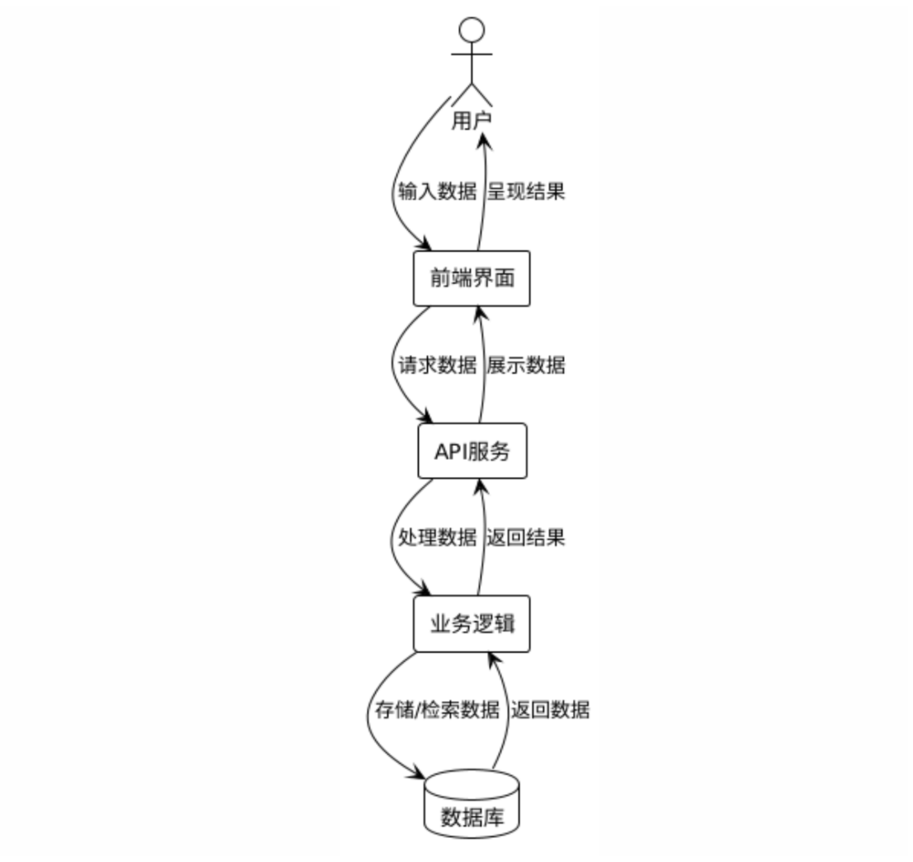

图 10: Data Flow Diagram

8 Interface Requirements

8.1 System Group and Other Groups Interface Overview

As the technical infrastructure provider of the project, the System group needs to establish clear interfaces with the UI group, Data group, and Analysis group to ensure smooth collaboration among all groups. The following diagram shows the main interface relationships between the System group and other groups:

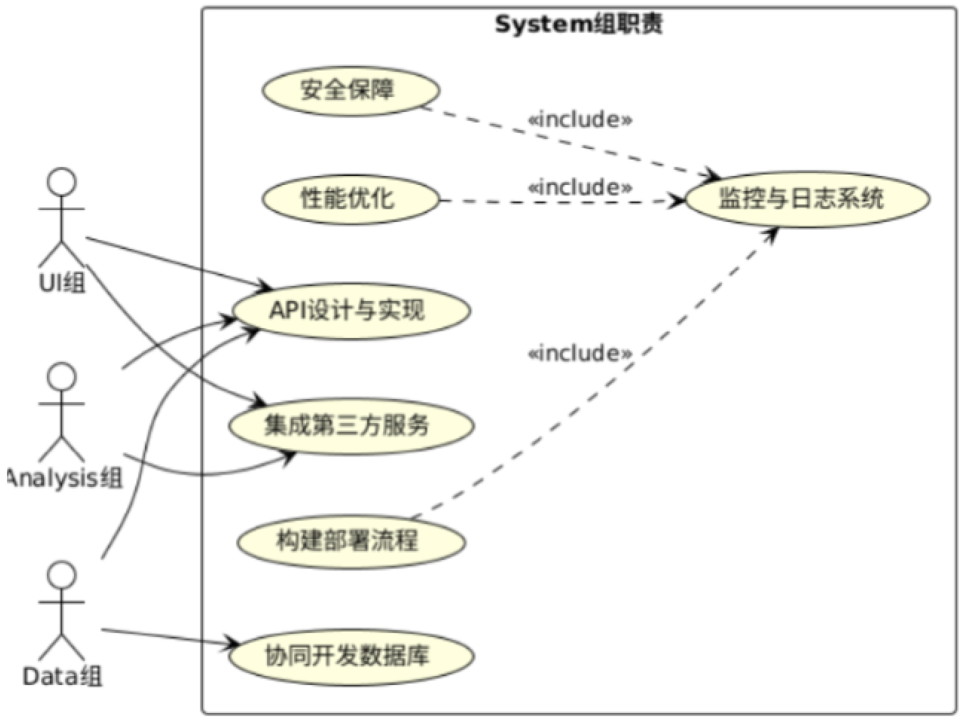

图 11: System Group and Other Groups Interface Relationship Diagram

8.2 System Group and UI Group Interface

The System group provides backend service support for the UI group, with interfaces mainly manifested at the API level.

8.2.1 API Interface Specifications

- **Interface Style**: RESTful API
- **Data Format**: JSON
- **Authentication Method**: JWT (JSON Web Token)
- **Status Code Usage**: Follow HTTP standard status codes
- **Version Control**: Include version number in URL path, e.g., /api/v1/users
- **Error Handling**: Unified error response format
- **Pagination Mechanism**: Support limit/offset and cursor pagination

8.2.2 Core API Interfaces

Interface Name	**Request Method**	**Path**	**Description**
User Authentication	POST	/api/v1/auth/login	UI group calls this interface for user login authentication
Get User Information	GET	/api/v1/users/profile	Get detailed information of the current logged-in user
Data Query	GET	/api/v1/data	Support various query parameters, return paginated data
Data Operations	POST/PUT/DELETE	/api/v1/data	Create, update, delete data
File Upload	POST	/api/v1/files	Support multi-file upload, return file URL
System Configuration	GET	/api/v1/config	Get system configuration parameters required by UI

8.2.3 Frontend-Backend Interaction Process

1. **User Authentication Process**
    - UI group collects user credentials (username/password)
    - Call authentication API to obtain access token
    - Subsequent requests carry token for identity verification
2. **Data Interaction Process**
    - UI group constructs requests according to API specifications
    - System group processes requests and returns appropriate responses
    - UI group updates interface based on responses
3. **Error Handling Process**
    - System group returns standard error codes and detailed error messages
    - UI group displays appropriate error prompts based on error type

8.3 System Group and Data Group Interface

The System group and Data group closely collaborate in database development and data processing, with interface design considering data access efficiency and security.

8.3.1 Database Access Interface

- **Data Access Pattern**: Repository pattern
- **ORM Framework**: Provide object-relational mapping capabilities
- **Transaction Support**: Support distributed transactions
- **Caching Mechanism**: Multi-level caching strategy
- **Connection Pool Management**: Optimize database connection resources

8.3.2 Main Data Interaction Scenarios

Interaction Scenario	**Interface Method**	**Description**
Data Query	findById(), findByCondition()	Support single record query and conditional query, Data group responsible for SQL optimization
Data Writing	save(), update(), delete()	Provide unified data writing interface, System group handles concurrency control
Batch Operations	batchInsert(), batchUpdate()	High-performance batch data processing interface
Data Migration	migrateData()	Data migration interface for version upgrades
Data Validation	validate()	Data business rule validation to ensure data consistency

8.3.3 Responsibility Division

- **Data Group Responsibilities**:
  - Design database structure (tables, indexes, constraints)
  - Optimize database query performance
  - Write database change scripts
  - Manage database versions
- **System Group Responsibilities**:
  - Implement data access layer code
  - Handle data business logic
  - Ensure data access security

- Implement data caching mechanism
- Manage database connection resources

8.4 System Group and Analysis Group Interface

The System group provides data services and computing resources for the Analysis group, supporting the Analysis group in conducting various data analyses.

8.4.1 Data Analysis API

- **Data Retrieval API**: Provide structured data query interface
- **Data Stream API**: Support stream data processing
- **Computing Task API**: Support asynchronous analysis task submission and result retrieval
- **Result Storage API**: Provide analysis result persistence capability

8.4.2 Main Interaction Scenarios

Interaction Scenario	**Interface Method**	**Description**
Raw Data Retrieval	fetchRawData()	Get raw data under specified conditions, support pagination and filtering
Submit Analysis Task	submitTask()	Submit asynchronous analysis task, return task ID
Get Task Status	getTaskStatus()	Query execution status of analysis task
Get Analysis Result	getTaskResult()	Get analysis result of completed task
Register Data Callback	registerCallback()	Automatically notify Analysis group after analysis result is generated

8.4.3 Data Format Specifications

- **Input Data Format**: JSON, CSV, or binary format
- **Metadata Description**: Provide data structure and field description
- **Output Result Format**: Unified analysis result format
- **Error Message Format**: Detailed error codes and descriptions

8.5 Cross-group Collaboration Interface

Some functionalities require collaboration among multiple groups, and the System group needs to provide interfaces to coordinate multi-group work.

8.5.1 Notification and Event System

- **Event Publishing Interface**: System group publishes system events
- **Event Subscription Interface**: Other groups subscribe to events of interest
- **Message Queue**: Ensure event asynchronous processing does not block main process
- **Event Types**: System status changes, task completion, data updates, etc.

8.5.2 Integration Test Interface

- **Test Environment Configuration**: Provide independent test environment interface
- **Test Data Generation**: Generate test data interface
- **State Reset**: Reset system state after testing interface
- **Simulation Interface**: Simulate third-party service interface

8.6 Interface Documentation and Version Management

8.6.1 Interface Documentation Specifications

- **Documentation Format**: Adopt OpenAPI (Swagger) specification
- **Required Fields**: Interface URL, request method, parameter description, response format, error codes
- **Example Code**: Call examples in various languages
- **Update Record**: Interface change history

8.6.2 Interface Version Management

- **Version Naming**: Major version.Minor version.Revision version (e.g., 1.2.3)
- **Compatibility Principle**: Minor versions and revision versions must be backward compatible

- **Deprecation Process**: Announce interface deprecation in advance, retain transition period

8.6.3 Interface Change Management

- **Change Review**: Important interface changes require multi-group review
- **Change Notification**: Notify relevant parties before interface changes
- **Change Testing**: New interfaces must pass automated testing
- **Rollback Mechanism**: Quick rollback when problems are found after interface launch

9 Constraints and Assumptions

9.1 Technical Constraints

1. The system should be developed using modern Web technologies
2. Database selection should consider performance and scalability
3. The system should support mainstream browsers
4. Development should follow secure coding standards

9.2 Business Constraints

1. The system should comply with relevant industry regulations and standards
2. Data processing should follow data protection principles
3. System functions should be compatible with existing business processes

9.3 Assumptions

1. Users have basic computer operation skills
2. System operating environment meets minimum hardware requirements
3. Network connection is stable and reliable
4. User data can be migrated from existing systems

10 Acceptance Criteria

10.1 Functional Acceptance Criteria

System acceptance should meet the following functional standards:

1. Users can complete all core business processes
2. Data processing results are accurate and error-free
3. Report generation function operates normally
4. System management functions are complete and usable

10.2 Non-functional Acceptance Criteria

The system should also meet the following non-functional standards:

1. System response time meets performance requirements
2. Security testing shows no serious vulnerabilities
3. System reliability testing passes
4. User interface complies with usability principles

11 Appendix

11.1 Glossary

Term	**Definition**
API	Application Programming Interface
UI	User Interface
ETL	Extract, Transform, Load
CI/CD	Continuous Integration/Continuous Deployment
RESTful	Representational State Transfer, an architectural style for API design principles

11.2 Use Case Relationship Explanation

- **Include Relationship («include»)**: Indicates that a base use case is included in another use case and is a mandatory part
- **Extend Relationship («extend»)**: Indicates that a use case extends another use case's behavior under specific conditions
- **Use Relationship**: Indicates that an actor uses the functionality provided by a use case
- **Participation Relationship**: Indicates that an actor participates in the definition or discussion of a use case but is not the main implementer

11.3 Revision History

Version	**Date**	**Reviser**	**Revision Content**
1.0	March 31, 2025	Cai Xu	Initial document draft

表 6: Document Revision History