Financial Risk Assessment System Report

Yujie Han, Weikang Fu

April 24, 2025

Contents

1 Background

In recent years, the rapid advancement of internet finance has profoundly reshaped the financial sector, markedly enhancing service accessibility, accelerating transaction processes, and expanding the scale of financial operations. Nevertheless, this rapid development has concurrently given rise to a spectrum of novel and increasingly sophisticated risks, including but not limited to fraud, identity theft, and credit default. Traditional risk control approaches, predominantly based on manual review and predefined rules, are proving insufficient in coping with the vast volume, heterogeneity, and real-time demands of contemporary financial activities.

To effectively mitigate these emerging challenges, it is imperative for financial institutions to embrace intelligent and automated risk management solutions capable of real-time risk detection and adaptive decision-making. A well-designed risk control system not only minimizes potential financial losses but also ensures regulatory compliance, bolsters customer confidence, and supports the sustainable growth of financial enterprises.

In response to this pressing need, the present project aims to develop an internet finance risk control system characterized by a modular architecture and the integration of intelligent technologies, thereby achieving high levels of efficiency, scalability, and flexibility in risk governance.

2 Project Objectives and Requirements

2.1 Project Objectives

The primary goal of this project is to develop a risk control system tailored for internet finance, utilizing the Model-View-Controller (MVC) architectural pattern. The system will incorporate essential functional modules such as user management, risk control lists, rule engine, model management, and log monitoring. These modules are analyzed using use case diagrams, and the project provides a comprehensive overview of the functional and technical requirements, aiming to offer financial institutions an integrated and effective risk management solution.

The specific objectives of the project include:

1. **Enhance Risk Identification Capability**: Build a layered risk control framework that combines blacklist/whitelist filtering, rule-based assessments, and machine learning models to improve the precision and coverage of risk detection.

2. **Improve Decision Efficiency**: Enable automated risk evaluation in critical scenarios such as loan applications, minimizing the need for manual review and significantly accelerating decision-making processes.

3. **Enhance Risk Control Flexibility**: Allow for flexible and real-time configuration of rules and models, empowering institutions to quickly adapt to policy changes, emerging risks, and evolving business needs.

4. **Strengthen Monitoring and Alerts**: Establish a detailed logging system coupled with real-time alert mechanisms to ensure prompt detection and resolution of anomalies or high-risk events.

5. **Optimize User Experience**: Design an intuitive user interface with transparent decision explanations to facilitate effective interaction and understanding by business users and risk managers.

Through the implementation of this project, financial institutions can effectively control credit risk, reduce default rates, and improve decision efficiency, thus supporting healthy business growth. Meanwhile, the modular design and good scalability of the system also lay a foundation for future functional enhancement and technical evolution.

2.2 Core Requirements

Based on an in-depth analysis of financial risk control scenarios and business needs, the system must meet the following core requirements:

2.2.1 Functional Requirements

User Authentication and Access Control With the continuous expansion of application scenarios in financial risk control systems, the need for fine-grained access management has become increasingly prominent. In high-risk financial environments, regulatory compliance and security auditing demand a robust mechanism for user authentication and access control. This system enforces strict authentication and authorization processes to ensure that users with different roles obtain access permissions that match their responsibilities, thereby safeguarding system security and operational compliance.

The system adopts a Role-Based Access Control (RBAC) model, classifying users into two primary roles: administrators and general users. Each role is granted differentiated permissions tailored to its functions. Additionally, the system integrates a comprehensive logging mechanism that records user logins and critical operations for future audits and troubleshooting. This design meets the stringent security requirements of financial risk control while also providing efficient management tools for system administrators.

User Management The system supports the creation, retrieval, modification, and deletion of user accounts, and provides batch operation interfaces to enhance administrative efficiency. Through role configuration, the system enforces access control strategies based on permissions, effectively securing access to system resources.

Risk Control List Management The system enables unified management of blacklists, whitelists, and graylists. It provides functionalities for list addition, retrieval, and status updates, and allows rapid checks for list matches to support real-time risk decision-making scenarios.

Risk Control Rule Configuration The system supports the full lifecycle management of risk control rules, including rule creation, modification, and deletion. It incorporates a priority setting and status control mechanism to ensure precise and flexible rule triggering. Additionally, it offers batch import/export functions to facilitate multi-environment deployment and rule migration. Built-in analytics on rule trigger frequency further aid in rule evaluation and optimization.

Risk Model Management The system facilitates the upload, approval, and deployment of risk models, providing version control and runtime status management. It also supports consistency validation and version rollback to improve the reliability and controllability of model releases. Continuous monitoring of model runtime status allows timely detection of anomalies and errors, ensuring high availability of model services.

Loan Risk Assessment The system supports the input and processing of loan application data. Based on predefined rules, it automatically scores applications and determines their risk levels, enabling automated approvals and risk interpretation. After assessment, the system displays the results along with details of the triggered rules, enhancing the explainability and transparency of decision-making.

Log Monitoring and Alerts To ensure observability and maintainability, the system includes a comprehensive log monitoring and alerting module. It records both operational and runtime logs and supports multi-dimensional querying and analysis. This module integrates anomaly detection and real-time alert features, as well as log status tracking and management tools to support routine maintenance and root cause analysis.

2.2.2 Non-functional Requirements

Security The system implements an access control mechanism based on user authentication and authorization, combined with encrypted storage of sensitive data and secure communication channels, effectively ensuring data privacy and interaction security. In addition, comprehensive operation logging and audit tracking, along with transactional management mechanisms, guarantee traceability of critical operations and consistency of data.

Reliability The system is built upon a robust data processing flow and runtime framework, featuring exception detection and error recovery mechanisms. Periodic data backups and disaster recovery strategies enhance the system's resilience to risks. Real-time system status monitoring and alert functionalities further strengthen its ability to respond to abnormal conditions.

Performance The system optimizes database access and risk assessment procedures to improve response speed and concurrent processing capabilities, achieving efficient resource scheduling and utilization. Meanwhile, the user interface is designed with intuitive interaction and usability in mind, providing clear workflows and functional guidance to enhance overall usability and user experience.

Scalability The system is capable of integrating new features and adapting to diverse business scenarios, supporting applications of various scales and meeting evolving functional requirements.

Maintainability With a well-structured codebase and comprehensive logging and monitoring framework, the system ensures ease of future maintenance, configuration adjustments, and deployment upgrades. This contributes to high maintainability and long-term evolvability of the system.

3 Module Requirements Analysis

Based on an understanding of the overall system requirements, this section provides an in-depth analysis of each functional module to clarify their specific functionalities and key implementation points.

3.1 Authentication Module

The authentication module serves as the security gateway of the system, responsible for verifying user identity and controlling access permissions. It forms the first line of defense in the security architecture of the risk control system. In financial risk control scenarios, the module not only ensures the authenticity of user identities but also enables precise control of access permissions based on user roles, thereby preventing risks of data leakage or tampering caused by unauthorized access.

3.1.1 User Login

The system features a login interface developed using PyQt5, requiring users to input their username and password for identity verification. The interface offers two login modes: administrator and standard user. During login, users select the appropriate role, and the system validates the provided credentials against the identity information stored in the database. It also verifies whether the user possesses the necessary privileges associated with the selected role.

The authentication mechanism is implemented through the ‘login‘ method in the ‘AdminController‘ class. This method first invokes ‘UserModel.authenticate‘ to verify the user’s credentials, followed by a role check to ensure consistency between the requested role and the one recorded in the system. If a standard user attempts to log in as an administrator, the system will deny the request, thereby maintaining strict role-based access control. Upon successful authentication, the system logs the event—recording the username, login time, and selected role—to support subsequent auditing and security analysis.

3.1.2 Access Control

Strictly distinguishes the access scope between administrators and regular users. Administrators have full access to all system functions, while regular users are limited to viewing and managing their own personal information.

3.2 User Management Module

The user management module handles the full lifecycle of system users, including creation, modification, deletion, and retrieval of user information.

A unified data model manages basic attributes (e.g., username, password, contact details) and role-based permissions centrally.

The interface displays user data in a table format with support for filtering, sorting, and pagination, improving operation and search efficiency. Both single and batch operations are supported, with immediate feedback on execution results. All actions are logged to ensure traceability and audit compliance.

The module is implemented using PyQt5 for the GUI and follows the Model-View-Controller (MVC) architecture to separate logic from presentation. Input validation mechanisms ensure data integrity and consistency.

3.3 Model Management Module

The risk control model is a core component of decision-making, and the standardization and efficiency of its management directly affect the accuracy and timeliness of risk control decisions. The model management module is responsible for the lifecycle management of risk control models, including creation, versioning, approval workflows, deployment, and monitoring, ensuring a controllable and traceable process from development to production.

3.3.1 Model Metadata Management

The system manages metadata such as model name, version number, environment type (development, testing, or production), and creation time through the `ModelManagementModel` class. Users can conveniently view basic information of existing models via the graphical interface and are provided with the ability to add new models or modify attributes of existing ones as needed.

Regarding model file management, the system supports both uploading and storing of model files. Users can select files using a file dialog, and the system securely stores these files in binary format within the database, ensuring both the integrity and confidentiality of model assets. Additionally, a version control mechanism is implemented for model files, allowing rollback to previous versions when necessary.

In terms of model state management, the system tracks both validation status and rollback state. Using the `toggle_verified` and `toggle_rollback` methods, administrators can mark the validation status of models and indicate whether a rollback is required, while all such operations are logged for auditing purposes.

Furthermore, the system offers a comprehensive set of model querying capabilities. Users can filter models by name, environment, validation status, and other criteria to efficiently locate relevant model information. A dedicated error query feature is also available, which displays all models that have encountered errors along with their corresponding error details.

3.3.2 Model Approval Workflow

Includes consistency checks and multi-level approvals with full status tracking and operation history. Supports model deployment and rollback, with runtime monitoring and error logging to ensure stability. Users can query models via a GUI with multi-criteria filters and access detailed error information.

3.4 Auxiliary Modules

In addition to the core modules of authentication, user management, and model management, the system also includes the following auxiliary modules to provide a more comprehensive risk control capability.

3.4.1 Risk Control List Management Module

In financial risk control systems, watchlists—including blacklists, whitelists, and greylists—serve as critical instruments for implementing precise risk management strategies. The watchlist management module is responsible for the centralized administration of these lists, supporting operations such as addition, querying, status modification, and batch import/export.

This system's watchlist management is implemented via the `NameListModel` class, which provides comprehensive functionality for managing list data. Through the user interface, new list entries can be added by specifying key attributes such as rule ID, risk level, list type, business line, risk label, risk domain, value, and value type. Upon successful addition, the system automatically generates operation logs to ensure full traceability.

For querying purposes, the system offers a multidimensional search function, enabling users to filter records by criteria such as RCID, ID, status, type, risk domain, and business line. Query results are presented in a tabular format, displaying key information and the current status of each record. To enhance readability, the system employs color-coded indicators for status fields—for instance, valid entries are highlighted with a green background.

In terms of status management, users are permitted to update the status of list entries, such as marking them as valid or invalid. Deletion of records is also supported; however, a confirmation prompt is enforced prior to deletion to prevent accidental data loss.

One of the core features of the module is the hit-checking mechanism, implemented via the `check_hit` method. Users can input a value and its corresponding type to determine whether it matches any entries in the risk control lists. The system returns all matched rule details, including rule ID, name, and expression, aiding users in understanding the reason for each hit.

Regarding batch operations, the system includes an interface for batch import, laying the groundwork for future functionality expansion, although this feature is not yet fully implemented in the current version.

3.4.2 Risk Control Rule Configuration Module

The system's risk control rule management is implemented based on the `RuleModel` class, which provides comprehensive functionality for managing rule-related data. Through the user interface, users can add new risk control rules by specifying attributes such as rule name, rule expression, external rule flag, and priority level. Upon creation, the system automatically generates a unique rule ID and logs the operation to ensure full traceability.

In terms of rule expressions, the system supports a variety of types, including credit score rules (e.g., `credit_score < 550`), overdue count rules (e.g., `overdue_count > 5`), maximum overdue days rules (e.g., `max_overdue_days > 90`), debt ratio rules (e.g., `debt_ratio > 0.6`), and multi-loan conditions (e.g., `has_mortgage and has_car_loan`).

For rule status management, the system allows enabling and disabling of rules, enabling users to dynamically adjust rule activity according to business requirements. The interface also displays the operational status of each rule, such as "Active," "Disabled," or "Error," providing at-a-glance system insights.

The system conducts statistical analysis of rule hits through the `get_rule_hit_count` and `get_rule_hit_by_business` methods. Users can examine the number of times each rule has been triggered to assess its effectiveness, or view hit statistics by business line to analyze risk distribution across operational units. These statistics are presented in a tabular format, offering a clear visual reference for data-driven decision-making by risk analysts.

The rule management interface is designed with user experience in mind, adopting a tabbed layout that separates rule listings, hit statistics, and business-line-based analytics. It provides an intuitive and organized environment for filtering and managing rules. Rule statuses are visually differentiated through colored indicators—for instance, green dots for active rules, gray for disabled, and red for error states.

3.4.3 Loan Evaluation Module

This module integrates risk control rules and watchlists to implement a complete workflow that spans from application data entry to risk assessment and presentation of decision outcomes.

The loan evaluation functionality is realized through the `evaluate_loan_application` method within the `RiskControlController` class. This method begins by collecting the applicant's basic information—such as name, ID number, phone number, and address—alongside loan-specific details including amount, term, and purpose. It also gathers a set of risk control indicators, such as credit score, number of overdue instances, maximum overdue days, debt ratio, and whether the applicant holds a mortgage or car loan. Subsequently, the system retrieves all active risk control rules and evaluates each rule to determine whether the application data triggers any rule conditions.

The core logic of rule evaluation is implemented in the `_check_rule_triggered` method, which parses the rule expressions and applies them to the input data to ascertain whether a rule has been triggered. For each triggered rule, the system deducts a corresponding score based on the severity of the rule. Upon completion of the evaluation, the system calculates a final score—starting from a base of 100 points, with deductions applied for each triggered rule—and makes a loan approval decision accordingly, typically approving applications that score above 60.

The system logs the evaluation results in detail, including applicant information, assessment score, final decision, and the list of triggered rules. In cases where the loan is denied, the applicant's information is added to the risk control watchlist, serving as a reference for future decision-making processes.

3.4.4 Log Monitoring Module

The log monitoring module is responsible for recording both operational and runtime logs, offering multidimensional querying and analytical capabilities to facilitate anomaly detection and alerting, as well as supporting log handling and status management.

This system's logging functionality is implemented using the `LogMonitoringModel` and `LogModel` classes. Two categories of logs are maintained: operational logs and runtime logs. Operational logs capture user interactions within the system, such as login events, rule additions, and model status modifications. Runtime logs, on the other hand, document events during system execution, including rule evaluations and model execution results, with particular emphasis on errors and exceptions.

For log retrieval, the system provides advanced filtering options. Users can filter log records based on keywords, log type (operational/runtime), time range, and other criteria. The query results are displayed in tabular format, detailing information such as log ID, timestamp, user, operation description, error messages, exception details, warning status, and processing status.

Anomaly detection constitutes a key aspect of the log monitoring module. The system employs the `get_pending_warnings` method to identify unresolved warning logs, enabling administrators to detect and address system anomalies promptly. For each detected anomaly, the system provides a resolution tracking feature; users can mark logs as processed using the `mark_as_done` method to prevent redundant handling.

4 Database Design

4.1 Database Tables and Schema

Users Table

Field Name	Key	Data Type	Description
id	PK	INTEGER	Auto-increment primary key
username	UNIQUE	TEXT	Unique username
password		TEXT	Hashed password (salted)
is_admin		BOOLEAN	Admin flag (TRUE/FALSE)
full_name		TEXT	Full name of the user
email	UNIQUE	TEXT	Email address
phone	UNIQUE	TEXT	Phone number
created_at		TIMESTAMP	Record creation time
updated_at		TIMESTAMP	Last updated time

Table 1: Users Table Schema

Log Monitoring Table

Field Name	Key	Data Type	Description
id	PK	INTEGER	Auto-increment primary key
operator_id	FK	INTEGER	Reference to `Users(id)`
operation		TEXT	Operation description
error_info		TEXT	Error information (if any)
exception_info		TEXT	Exception details (if any)
is_warning		BOOLEAN	Whether it's a warning
is_done		BOOLEAN	Whether it's processed
warning_type		TEXT	Type of warning (optional)
created_at		TIMESTAMP	Log creation timestamp

Table 2: Log Monitoring Table Schema

Model Management Table

Field Name	Key	Data Type	Description
id	PK	INTEGER	Auto-increment primary key
log_id	FK	INTEGER	Reference to `Log Monitoring(id)`
model_name		TEXT	Name of the model
model_file_path		TEXT	Path or URL to the model file
model_version_id		INTEGER	Version of the model
environment		ENUM	Environment (dev/test/prod)
caller_id	FK	INTEGER	Reference to `Users(id)`
approver_id	FK	INTEGER	Reference to `Users(id)`
verified		BOOLEAN	Whether the model is verified
rolled_back		BOOLEAN	Whether the model has been rolled back
created_at		TIMESTAMP	Creation time
updated_at		TIMESTAMP	Last updated time

Table 3: Model Management Table Schema

Rule Management Table

Field Name	Key	Data Type	Description
id	PK	INTEGER	Auto-increment primary key
log_id	FK	INTEGER	Reference to `Log Monitoring(id)`
rule_name		TEXT	Name of the rule
rule_expression		TEXT	Logical expression
is_external		BOOLEAN	Whether it is an external rule
priority		ENUM	Priority (High/Medium/Low)
creator_id	FK	INTEGER	Reference to `Users(id)`
created_at		TIMESTAMP	Creation time
updated_at		TIMESTAMP	Last updated time

Table 4: Rule Management Table Schema

Credit Report Table

Field Name	Key	Data Type	Description
id	PK	INTEGER	Auto-increment primary key
user_id	FK	INTEGER	Reference to `Users(id)`
rule_id	FK	INTEGER	Reference to `Rule Management(id)`
log_id	FK	INTEGER	Reference to `Log Monitoring(id)`
status		BOOLEAN	Report status (Valid/Invalid)
data_source		TEXT	Source of credit data
risk_domain		TEXT	Category of risk (e.g., financial, fraud, etc.)
value_type		INTEGER	Type of the values (e.g., phone number, ID)
value		TEXT	Matched value
created_at		TIMESTAMP	Creation time

Table 5: Credit Report Table Schema

Watch List Table

Field Name	**Key**	**Data Type**	**Description**
id	PK	INTEGER	Primary key, auto-increment
user_id	FK	INTEGER	Reference to `Users(id)`
rule_id	FK	INTEGER	Reference to `Rule Management(id)`
log_id	FK	INTEGER	Reference to `Log Monitoring(id)`
report_id	FK	INTEGER	Reference to `Credit Report(id)`
risk_level		INTEGER	Risk level of this record
list_type		INTEGER	List type (e.g., 1 = blacklist, 2 = whitelist)
business_line		INTEGER	Code of the associated business line
risk_label		INTEGER	Risk label category code
risk_domain		INTEGER	Risk domain classification (e.g., personal, enterprise)
creator	FK	TEXT	Reference to `Users(username)`
create_time		DATE	Record creation date

Table 6: Watch List Table Schema

4.2 Database ER Diagram

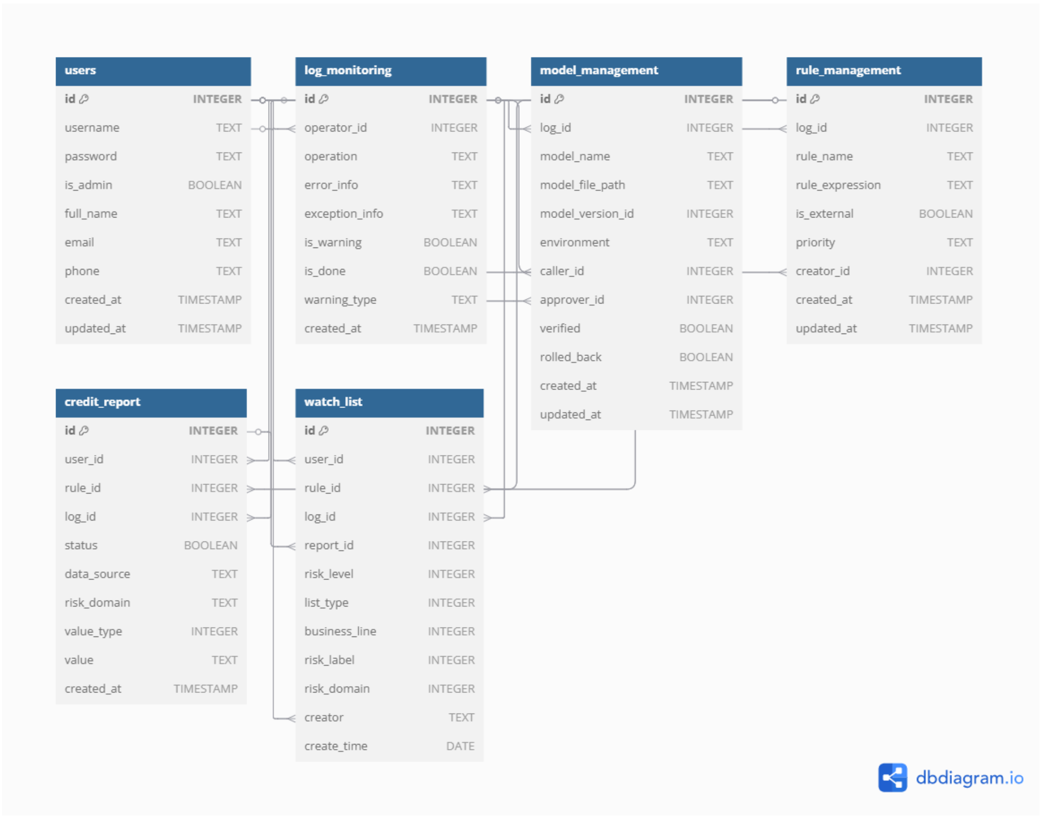


Figure 1: Entity-Relationship Diagram for the Core Database Schema

4.3 Technical Highlights

- **Connection Pooling**: A lightweight connection pool is implemented via `CONNECTION_POOL` to reduce the overhead of frequent database connections. The pool enforces a maximum size (`MAX_POOL_SIZE`); excess connections are closed to optimize resource usage.
- **Context Manager**: The `get_db_cursor` function provides a context-managed database cursor. Using `with` statements simplifies error handling and resource cleanup, ensuring atomic transactions with automatic commit or rollback.
- **Unified DB Interface**: The `execute_query` and `execute_update` functions abstract low-level database operations, offering a consistent

API for querying and updating. This improves readability and maintainability.

- **Batch Operation Support**: The `load_csv_to_table` function enables bulk data import from CSV files. It commits every 100 records to balance performance and memory usage.
- **Dual Database Engine Support**: The `DB_TYPE` configuration allows switching between SQLite and PostgreSQL. The system dynamically creates connections and schemas based on the selected engine, offering flexible deployment for varied use cases.

5 Software Design

5.1 Architecture Overview

This financial risk control system adopts the classic MVC (Model-View-Controller) architecture, effectively separating data access, business logic, and user interface presentation responsibilities.

5.1.1 Architectural Layers

- **Model Layer**: Responsible for data access and business entity definitions. It includes models like `UserModel`, `CreditReportModel`, `RuleModel`, `ModelManagementModel`, and `LogMonitoringModel`. It encapsulates database operations and provides unified data access interfaces.
- **View Layer**: Implements the user interface using PyQt5, including main windows (`AdminWindow`, `UserInfoWindow`) and functional tabs (`UserManagementTab`, `RiskControlTab`, etc.). It presents data and captures input but doesn't contain business logic.
- **Controller Layer**: Processes user requests, coordinates between model and view layers, and implements business logic. Controllers include `UserController`, `AdminController`, `RiskControlController`, etc.
- **Utility Layer**: Provides common functionalities like database connection management, logging, and data formatting. This layer supports modular reuse and reduces system complexity.

5.1.2 Technology Stack

- **Language**: Python 3.10 for its strengths in data processing and rapid development.
- **GUI Framework**: PyQt5 is used to build cross-platform graphical interfaces.
- **Database**: Supports SQLite (for lightweight use) and PostgreSQL (for production). The system can switch databases via configuration.
- **DB Access**: Custom connection pooling and context managers ensure efficient resource use and transactional safety.

5.1.3 Module Division

Based on business functionalities and data processing requirements, the system is structured into several key modules: the `Authentication Module` handles user login and access control; the `User Management Module` manages user account creation, deletion, updating, and retrieval; the `Watchlist Management Module` maintains information on blacklists, whitelists, and greylists; the `Risk Rule Configuration Module` supports the definition and management of complex risk control rules; the `Model Management Module` oversees version control and deployment of predictive models; the `Loan Evaluation Module` integrates rules and watchlist data to perform risk scoring; and the `Log Monitoring Module` records operational and runtime logs, enabling real-time auditing and alerting.

5.2 View Design

Based on PyQt5, the UI is layered into main windows, tab pages, and dialogs.

5.2.1 Login View

Implemented in `LoginWindow`:

1. Input for username and masked password
2. Role selection via radio buttons
3. Validation using controller logic
4. Success/failure feedback through signals

The login interface is designed to be clean and intuitive, offering clear user guidance. A sample implementation is shown below:

```
Algorithm: GUI Initialization for Login Interface

Input: None
Output: Initialized login interface with input fields and login button

1: Set window title to "User Management System - Login"
2: Set window size to (350×250) at position (300, 300)
3: Create vertical layout container
4: Create form layout with:
5:     └ Username input field
6:     └ Password input field (masked)
7: Create login role group with:
8:     └ Radio button: Regular User (default selected)
9:     └ Radio button: Administrator
10: Create login button and bind click event to login handler
11: Assemble components into vertical layout
12: Apply layout to window
```

5.2.2 User Management View

Implemented in `UserManagementTab`:

1. Table view showing user ID, role, and contact info
2. Filters for username and role with real-time updates
3. Inline edit/delete buttons with confirmation prompts
4. Form for adding/editing users with full attribute control

The user management interface is designed with a focus on operational convenience. All functionalities are integrated into a single tab to avoid the inefficiency of frequent interface switching. The key implementations are as follows:

```
Input: None
Output: Complete user management interface with filters, table, and form controls

1: Set window styles and parameters
2: Create main vertical layout container

3: ▷ Construct filter area
4:    Create horizontal layout for filters
5:    Add username search field with placeholder "Search username" (width: 200)
6:    Add role filter label and dropdown with items ["All", "Admin", "Regular user"] (width: 120)
7:    Add "Search" button and bind click event to user search function

8: ▷ Add user table
9:    Create table with 7 columns: ['ID', 'Username', 'Is Admin', 'Full Name', 'Email', 'Phone', 'Actions']

10: ▷ Construct user form frame and layout
11:    Create form frame named "statusFrame"
12:    Create vertical layout for form components

13: ▷ Add form control buttons
14:    Create horizontal button layout
15:    Add "Add" button → bind to add_user()
16:    Add "Update" button → bind to update_user()
17:    Add "Delete" button (id: deleteButton) → bind to delete_user()
18:    Add "Clear Form" button (id: secondaryButton) → bind to clear_form()

19: Assemble filter area, table, form, and buttons into main layout
20: Apply main layout to window
```

5.2.3 Model Management View

Implemented in `ModelManagementTab`:

1. Detailed table view with alternating row colors and state coloring
2. Action buttons that reflect current status (e.g., "Verify" or "Unverify")
3. Filters for name, environment, and verification status
4. Separate error tab with red highlight for failed models
5. Grouped input form for new models with file upload and approver

The model management interface is designed to provide visual control over the entire lifecycle of risk models. The key implementation is as follows:

```
Input: None
Output: Interface with model management and error tracking tabs

1: Apply window styles and properties

2: Create main vertical layout container

3: ▷ Create top-level tabs for switching views
4:    Create horizontal layout for tab buttons
5:    Add "Model Management" button
6:    Add "Model Errors" button
7:    Configure tab button properties

8: ▷ Construct filter area for model table
9:    Create horizontal filter layout
10:    Add model name search field with placeholder "Search model name" (width: 200)
11:    Add additional filter components (e.g., dropdowns, checkboxes)

12: ▷ Add main model table
13:    Create table with 10 columns:
14:        ['ID', 'Model Name', 'Version', 'Environment', 'Caller',
           'Approver', 'Verified', 'Rolled Back', 'Created At', 'Actions']

15: ▷ Add model error table
16:    Create table with 8 columns:
17:        ['ID', 'Model Name', 'Version', 'Environment', 'Approver',
           'Error Message', 'Exception', 'Log Time']

18: ▷ Create form area for adding new models
19:    Initialize QFrame for add-model form
20:    Add input components and layout for model registration

21: ▷ Assemble all components into main layout
22:    Add tab layout, filter layout, table widgets, and form area
23: Apply main layout to the window
```

5.2.4 Risk Rules and Lists View

1. **List Management**: Table with RCID, type, hits, status, etc. Colored status indicators, full filtering, and actions

2. **Rule Management**: Table with rule ID, status (green/gray/red), add/enable/disable options

3. **Rule Statistics**: Tables for hit counts per rule and business line, with colored rows

4. **Loan Evaluation**: Form sections for applicant, loan info, risk indicators, and result. Green for approval, red for rejection, with triggered rules highlighted

5.2.5 Log Monitoring View

1. Multi-criteria filtering by keyword, type, and date

2. Table with logs (ID, time, operator, content, errors, warnings)
3. Detail panel for full log view
4. Mark-as-done button with audit logging
5. Service status indicator (”Log service online”) and refresh + pagination

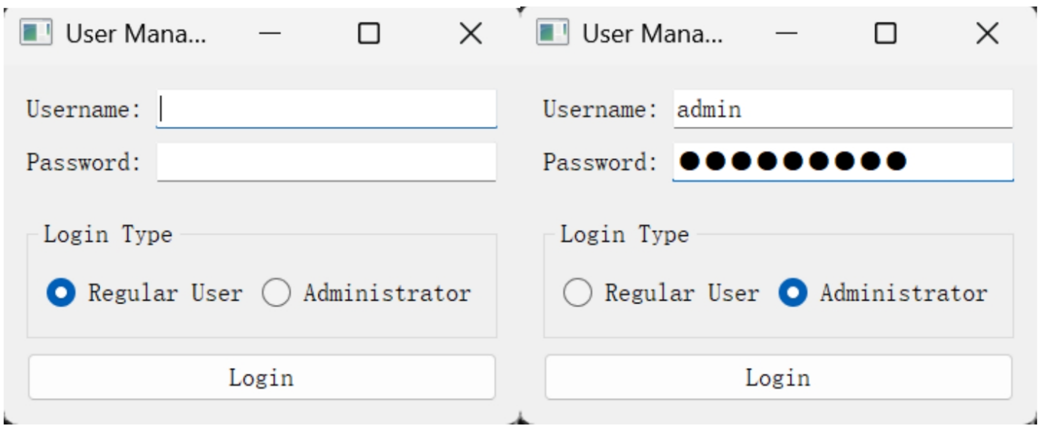
Three-part layout: top filters, middle log table, bottom details. Emphasizes usability and real-time monitoring.

6 System Implementation

This chapter provides a visual and textual overview of the system’s functional modules, illustrating the actual execution logic and user interaction processes. All features are driven by a graphical user interface, while backend logic is implemented using a controller–model architecture. Key workflows include authentication, security control, risk evaluation, and model management.

6.1 Authentication Process

Users log in by entering a username, password, and selecting a role via the login interface. The system invokes the user model through the authentication controller to validate credentials and then loads the appropriate interface based on the verification result. Two permission levels are defined: administrators have full access, while regular users are limited to viewing their personal information.



Username:
Password:
Login Type
[✓ Regular User] Administrator

Username: admin
Password: ●●●●●●●●●
Login Type
Regular User [✓ Administrator]

All authentication actions are logged for audit purposes, ensuring a complete loop of identity verification and access control. After entering login credentials and selecting the role (administrator or regular user), the system authenticates the user and loads the corresponding functional modules based on their role.

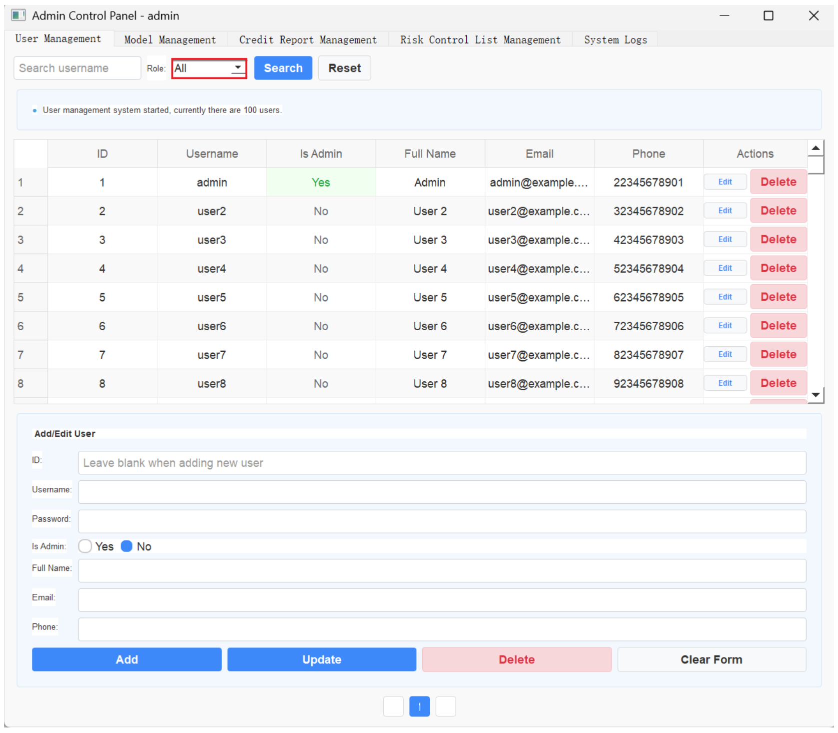
6.2 User Management Process

The user management module is a fundamental component of the system, enabling the creation, editing, deletion, and viewing of user accounts. It supports multi-user management, role assignment, and batch operations, serving as the core tool for administrators to maintain system access rights.

Administrators can perform user-related operations in the "User Management" interface, including search, add, edit, and delete. A filter area at the top allows role-based filtering (administrator or regular user), and keyword-based fuzzy search by username facilitates quick user location.

User data is displayed in a tabular format, including fields such as username, role, email, and phone number. For existing users, clicking the "Edit" button opens the edit mode, allowing modification of user attributes; clicking "Delete" triggers a removal operation.

A form area labeled "Add/Edit User" is located below the table, supporting both new user entry and existing user updates. Before submitting, required fields such as username, administrator flag, full name, email, and phone number must be completed. Upon submission, the system updates the database via controller–model interaction and reflects the changes in the interface for real-time feedback.



Admin Control Panel - admin

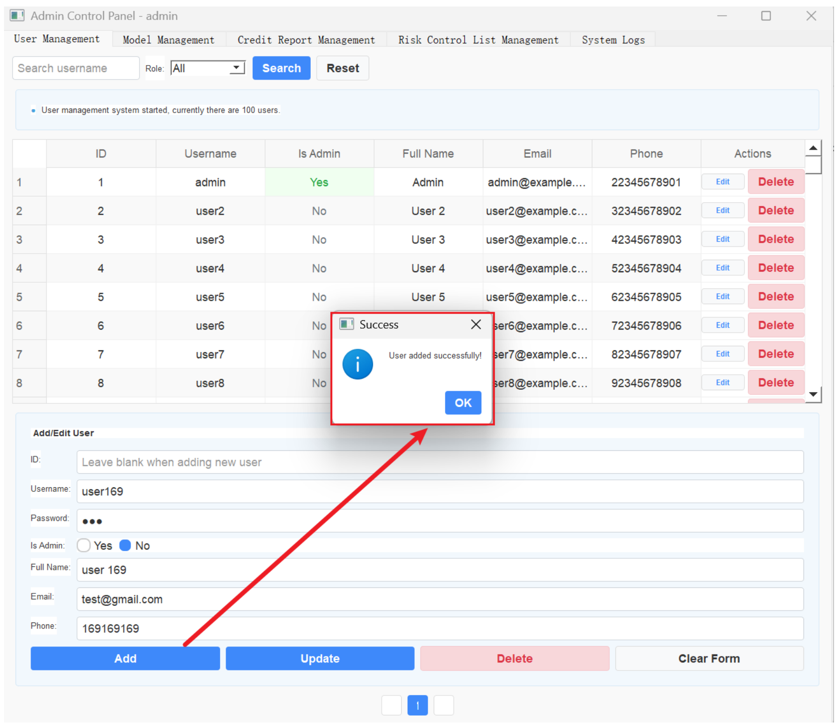
Role: All

• User management system started, currently there are 100 users.

	ID	Username	Is Admin	Full Name	Email	Phone	Actions
1	1	admin	Yes	Admin	admin@example....	22345678901	Edit Delete
2	2	user2	No	User 2	user2@example.c...	32345678902	Edit Delete
3	3	user3	No	User 3	user3@example.c...	42345678903	Edit Delete
4	4	user4	No	User 4	user4@example.c...	52345678904	Edit Delete
5	5	user5	No	User 5	user5@example.c...	62345678905	Edit Delete
6	6	user6	No	User 6	user6@example.c...	72345678906	Edit Delete
7	7	user7	No	User 7	user7@example.c...	82345678907	Edit Delete
8	8	user8	No	User 8	user8@example.c...	92345678908	Edit Delete

Add/Edit User

ID:	Leave blank when adding new user
Username:	
Password:	
Is Admin:	Yes [✓ No]
Full Name:	
Email:	
Phone:	



Admin Control Panel - admin

Role: All

• User management system started, currently there are 100 users.

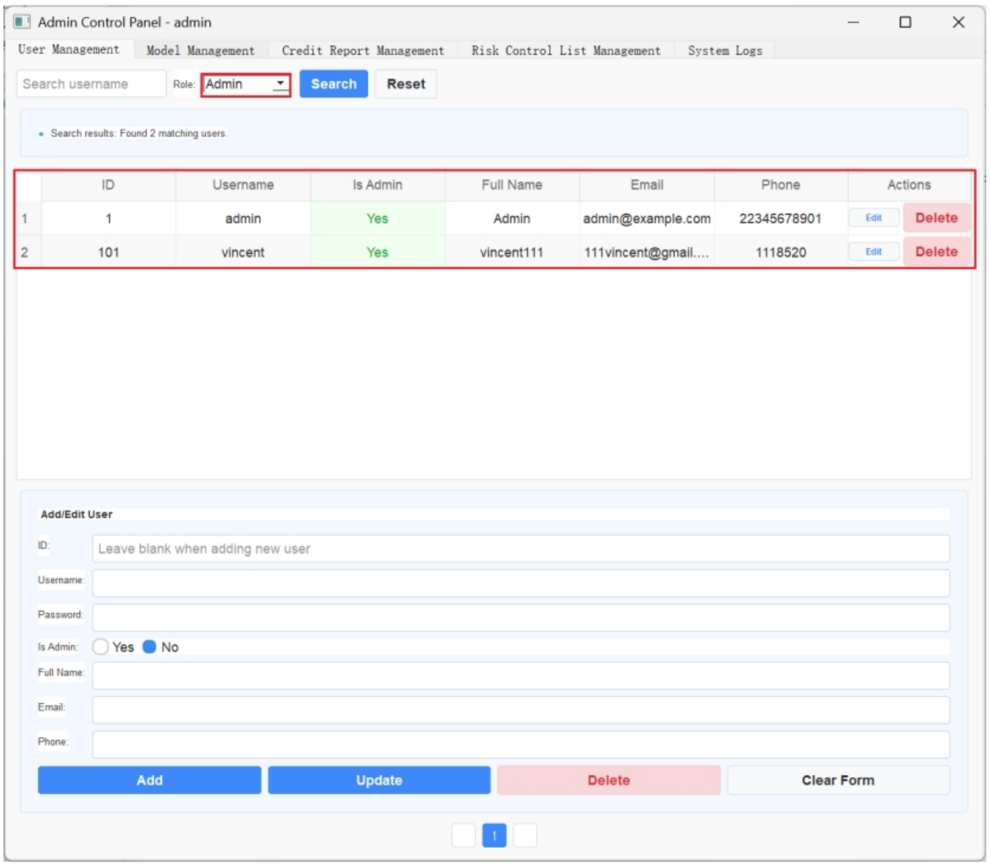
	ID	Username	Is Admin	Full Name	Email	Phone	Actions
1	1	admin	Yes	Admin	admin@example....	22345678901	Edit Delete
2	2	user2	No	User 2	user2@example.c...	32345678902	Edit Delete
3	3	user3	No	User 3	user3@example.c...	42345678903	Edit Delete
4	4	user4	No	User 4	user4@example.c...	52345678904	Edit Delete
5	5	user5	No	User 5	user5@example.c...	62345678905	Edit Delete
6	6	user6	No		ser6@example.c...	72345678906	Edit Delete
7	7	user7	No		ser7@example.c...	82345678907	Edit Delete
8	8	user8	No		ser8@example.c...	92345678908	Edit Delete

Success

User added successfully!

Add/Edit User

ID:	Leave blank when adding new user
Username:	user169
Password:	•••
Is Admin:	Yes [✓ No]
Full Name:	user 169
Email:	test@gmail.com
Phone:	169169169


Admin Control Panel - admin

User Management | Model Management | Credit Report Management | Risk Control List Management | System Logs

Search username | Role: Admin | Search | Reset

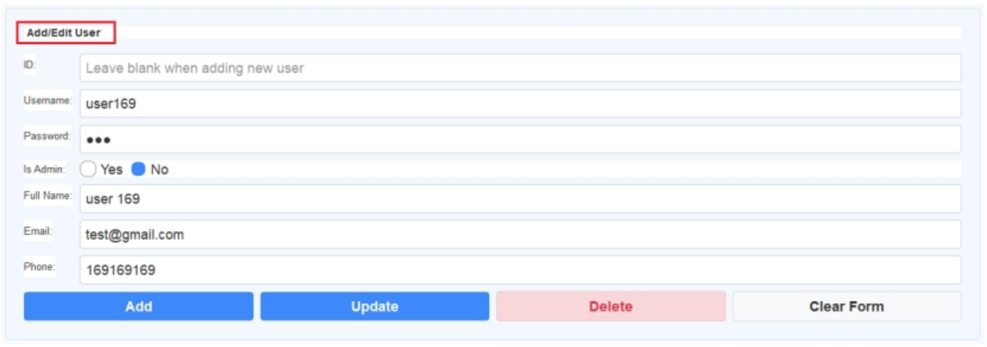
- Search results: Found 2 matching users.

	ID	Username	Is Admin	Full Name	Email	Phone	Actions
1	1	admin	Yes	Admin	admin@example.com	22345678901	Edit Delete
2	101	vincent	Yes	vincent111	111vincent@gmail....	1118520	Edit Delete

Add/Edit User

ID:	Leave blank when adding new user
Username:	
Password:	
Is Admin:	Yes [✓ No]
Full Name:	
Email:	
Phone:	

Add | Update | Delete | Clear Form


Add/Edit User

ID:	Leave blank when adding new user
Username:	user169
Password:	●●●
Is Admin:	Yes [✓ No]
Full Name:	user 169
Email:	test@gmail.com
Phone:	169169169

Add | Update | Delete | Clear Form

As shown in the figure, user data is displayed in a table with filtering by username or role. The form area below provides streamlined access to add, update, or delete users.

6.3 Model Management Process

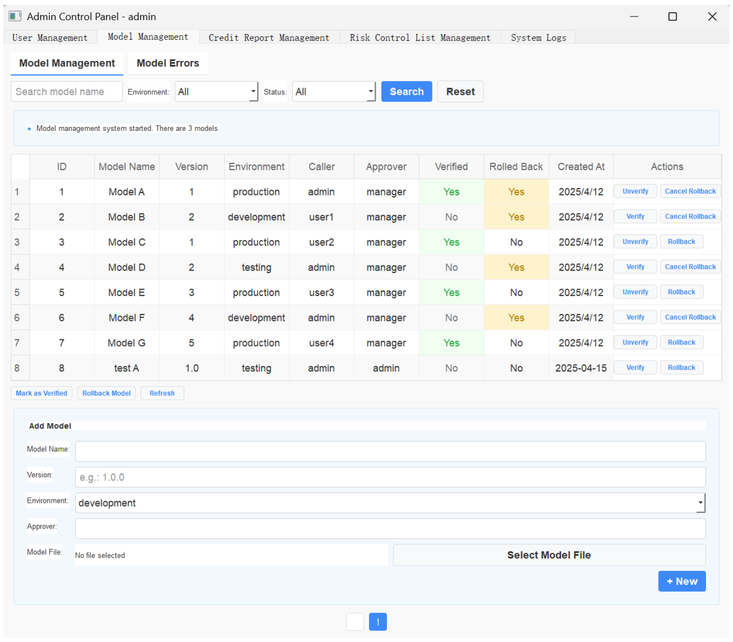
The model management module is a key component for maintaining the full lifecycle of risk control models. It supports model creation, version control, verification status management, rollback handling, and exception tracking. This module connects the front-end interface with the backend database and model logic, enabling users to visually manage models, and serves as the foundation for intelligent risk control decisions.

Administrators can create and import models via the "Model Management" interface, view model metadata and status, and perform actions such as verification and rollback. Users can filter models by name, environment (e.g., development, testing, production), and verification status through the top search panel, allowing quick identification of target models.

Model information is displayed in a table format with fields such as model name, version, caller, approver, verification status, rollback status, and creation time. Statuses are color-coded for clarity.

On the right side of the table, action buttons are provided, allowing users to execute context-aware operations such as "Verify," "Unverify," or "Rollback." All operations are logged and reflected in the model's state. At the bottom of the interface, an "Add Model" form allows manual entry of model details including name, version, environment, and approver, and supports model file upload via the "Select Model File" button.

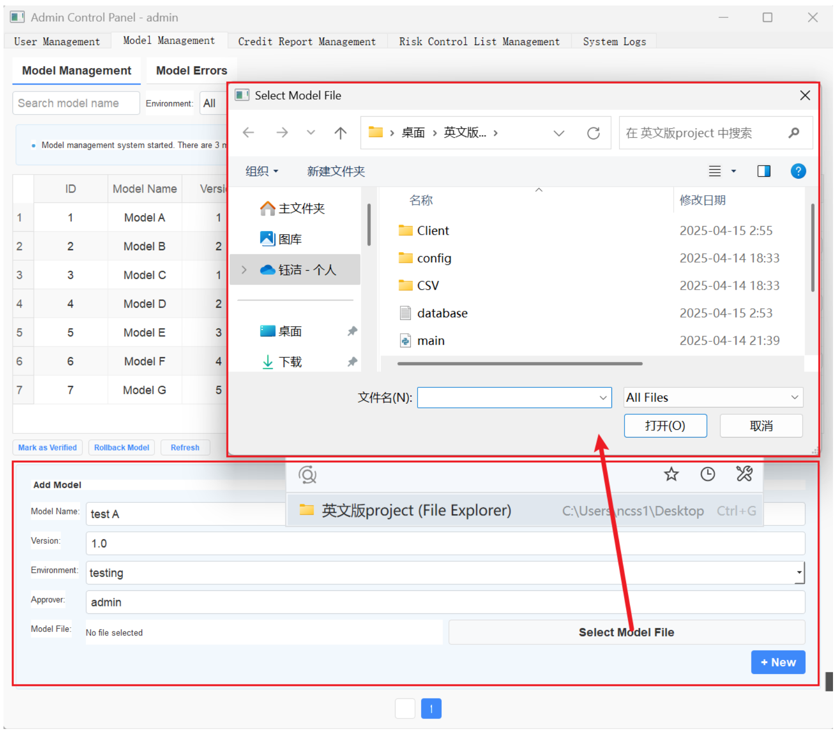
While file import is currently supported, model type recognition and automatic integration of machine learning models are not yet implemented. However, the interface is designed with extensibility in mind, and future versions will support model registration, runtime evaluation, and scoring.



• Model management system started. There are 3 models.

	ID	Model Name	Version	Environment	Caller	Approver	Verified	Rolled Back	Created At	Actions
1	1	Model A	1	production	admin	manager	Yes	Yes	2025/4/12	Unverify Cancel Rollback
2	2	Model B	2	development	user1	manager	No	Yes	2025/4/12	Verify Cancel Rollback
3	3	Model C	1	production	user2	manager	Yes	No	2025/4/12	Unverify Rollback
4	4	Model D	2	testing	admin	manager	No	Yes	2025/4/12	Verify Cancel Rollback
5	5	Model E	3	production	user3	manager	Yes	No	2025/4/12	Unverify Rollback
6	6	Model F	4	development	admin	manager	No	Yes	2025/4/12	Verify Cancel Rollback
7	7	Model G	5	production	user4	manager	Yes	No	2025/4/12	Unverify Rollback
8	8	test A	1.0	testing	admin	admin	No	No	2025-04-15	Verify Rollback

Model Name:
Version: e.g.: 1.0.0
Environment: development
Approver:
Model File: No file selected



• Model management system started. There are 3 m

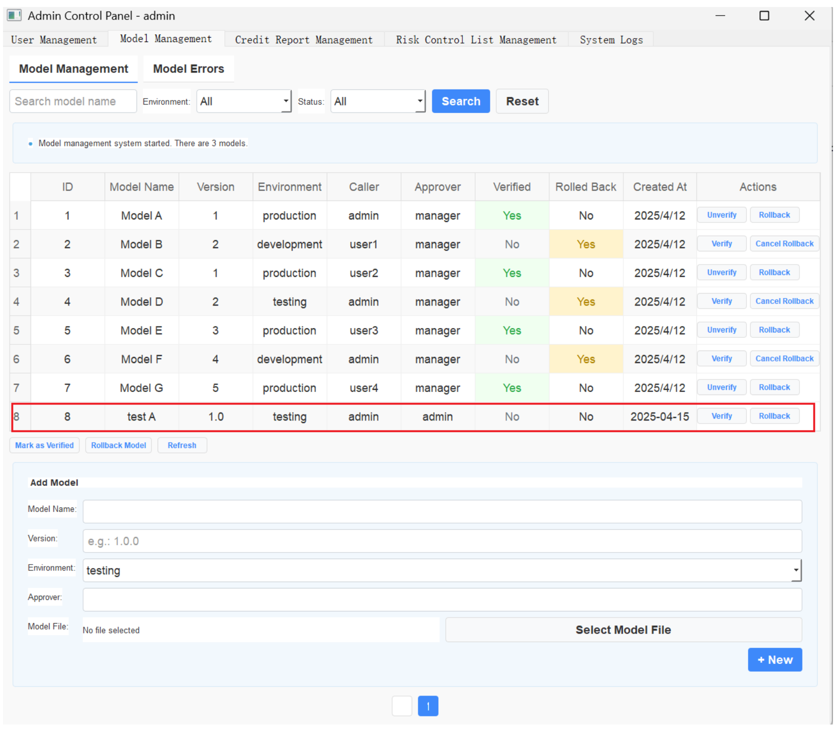
	ID	Model Name	Versi
1	1	Model A	1
2	2	Model B	2
3	3	Model C	1
4	4	Model D	2
5	5	Model E	3
6	6	Model F	4
7	7	Model G	5

名称	修改日期
Client	2025-04-15 2:55
config	2025-04-14 18:33
CSV	2025-04-14 18:33
database	2025-04-15 2:53
main	2025-04-14 21:39

C:\Users\ncss1\Desktop

Add Model

Model Name:	test A
Version:	1.0
Environment:	testing
Approver:	admin
Model File:	No file selected



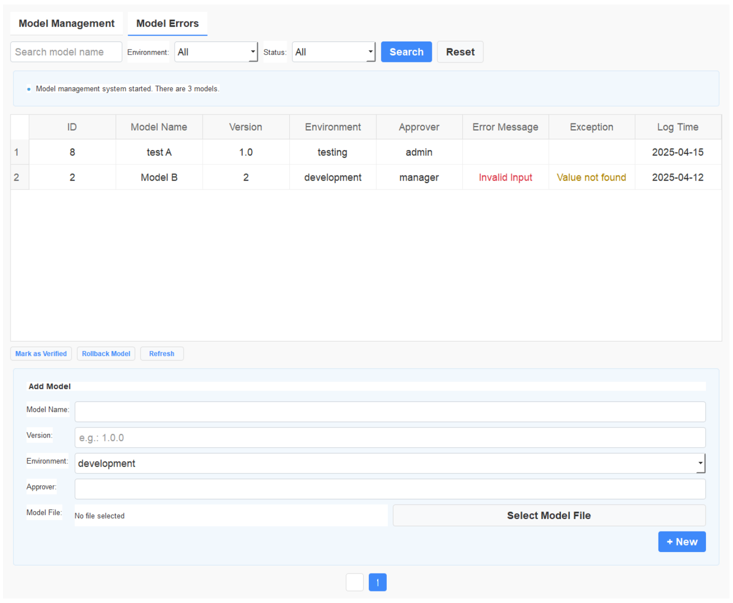
• Model management system started. There are 3 models.

	ID	Model Name	Version	Environment	Caller	Approver	Verified	Rolled Back	Created At	Actions
1	1	Model A	1	production	admin	manager	Yes	No	2025/4/12	Unverify Rollback
2	2	Model B	2	development	user1	manager	No	Yes	2025/4/12	Verify Cancel Rollback
3	3	Model C	1	production	user2	manager	Yes	No	2025/4/12	Unverify Rollback
4	4	Model D	2	testing	admin	manager	No	Yes	2025/4/12	Verify Cancel Rollback
5	5	Model E	3	production	user3	manager	Yes	No	2025/4/12	Unverify Rollback
6	6	Model F	4	development	admin	manager	No	Yes	2025/4/12	Verify Cancel Rollback
7	7	Model G	5	production	user4	manager	Yes	No	2025/4/12	Unverify Rollback
8	8	test A	1.0	testing	admin	admin	No	No	2025-04-15	Verify Rollback

Add Model

Model Name:	
Version:	e.g.: 1.0.0
Environment:	testing
Approver:	
Model File:	No file selected

As shown above, the interface presents both the main model table and the input form for adding new models, enabling intuitive model management and submission. Additionally, the system includes a model error management submodule that displays error messages and logs from model execution. Administrators can switch to the “Model Errors” tab to view detailed failure records, including model name, version, environment, error description, and timestamp. Errors are grouped by type for quick diagnosis.



Environment: All Status: All

- Model management system started. There are 3 models.

	ID	Model Name	Version	Environment	Approver	Error Message	Exception	Log Time
1	8	test A	1.0	testing	admin			2025-04-15
2	2	Model B	2	development	manager	Invalid Input	Value not found	2025-04-12

Add Model

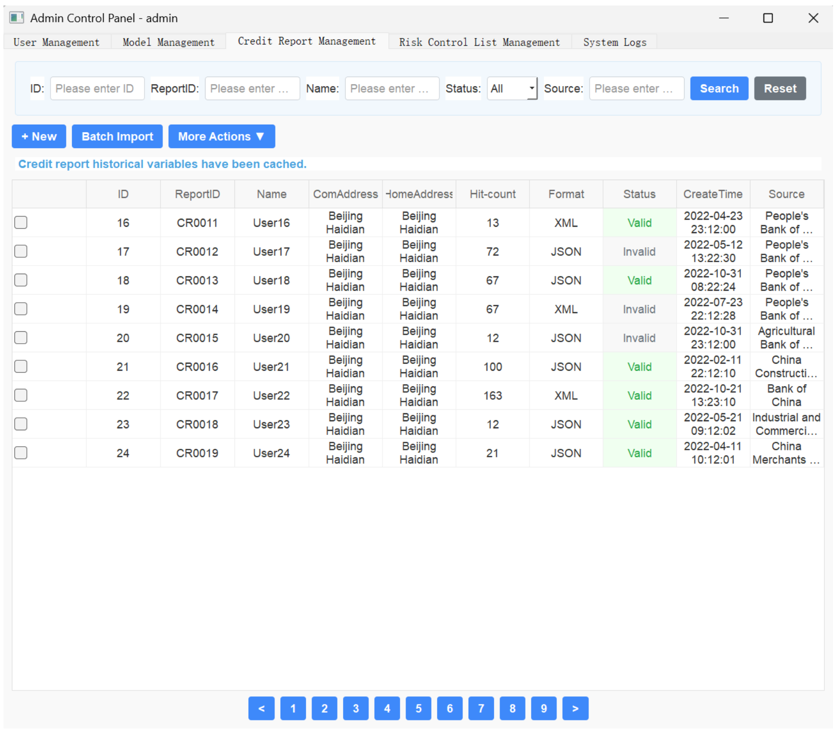
Model Name:	
Version:	e.g.: 1.0.0
Environment:	development
Approver:	
Model File:	No file selected

6.4 Credit Report Management Process

The credit report management module handles centralized management of user credit data, serving as a vital data source for assessing users’ creditworthiness and behavioral characteristics within the risk control system. It supports batch import of historical credit reports, format recognition, report status tagging, and multi-condition filtering, enabling efficient data cleansing and match analysis.

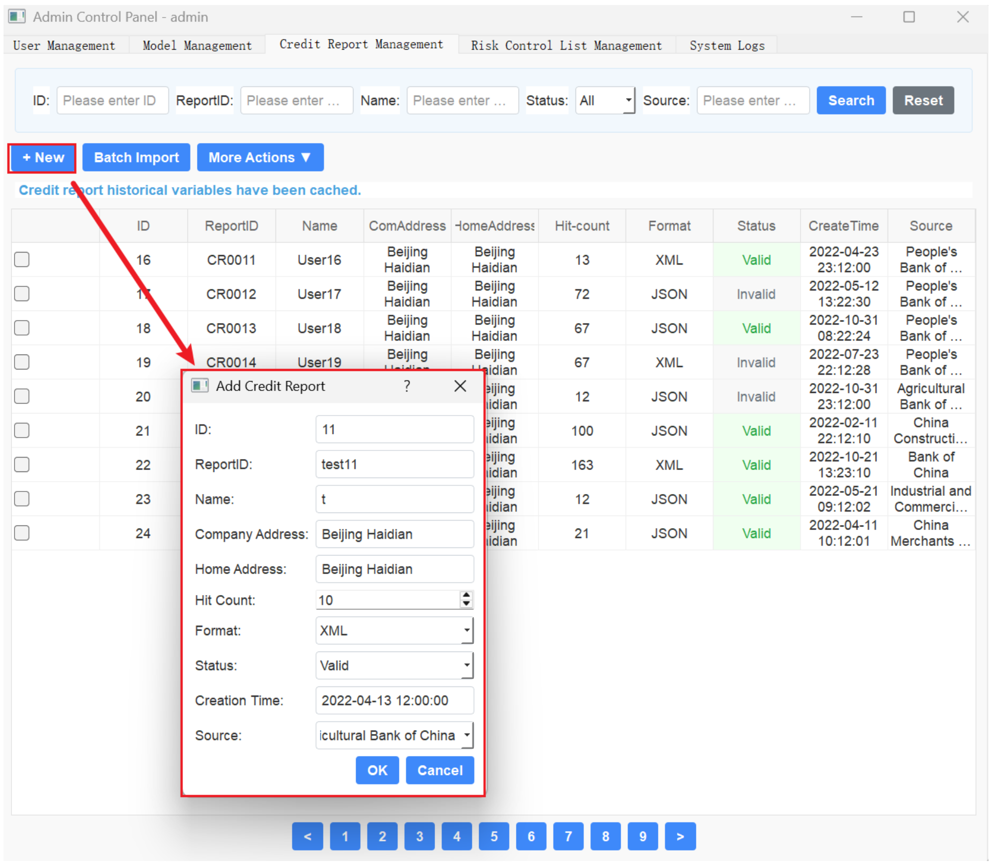
In the “Credit Report Management” interface, administrators can use the top filter section to input conditions such as report ID, username, status, or data source to quickly locate target records. The interface supports paginated browsing of credit reports, displaying key user identity information in a tabular format. Columns include Report ID, username, address, number of matches, data format (XML/JSON), status label (Valid/Invalid), creation time, and source organization.

Table headers are clickable for column-based sorting, and batch selection is supported for further processing. Administrators can click “+New” to add a new credit report or use the “Batch Import” button to upload multiple entries. The “More Actions” menu provides extended batch operations such as format conversion and status resetting. Expansion interfaces for these features are reserved for future functionality.



Credit report historical variables have been cached.

	ID	ReportID	Name	ComAddress	HomeAddress	Hit-count	Format	Status	CreateTime	Source
☐	16	CR0011	User16	Beijing Haidian	Beijing Haidian	13	XML	Valid	2022-04-23 23:12:00	People's Bank of ...
☐	17	CR0012	User17	Beijing Haidian	Beijing Haidian	72	JSON	Invalid	2022-05-12 13:22:30	People's Bank of ...
☐	18	CR0013	User18	Beijing Haidian	Beijing Haidian	67	JSON	Valid	2022-10-31 08:22:24	People's Bank of ...
☐	19	CR0014	User19	Beijing Haidian	Beijing Haidian	67	XML	Invalid	2022-07-23 22:12:28	People's Bank of ...
☐	20	CR0015	User20	Beijing Haidian	Beijing Haidian	12	JSON	Invalid	2022-10-31 23:12:00	Agricultural Bank of ...
☐	21	CR0016	User21	Beijing Haidian	Beijing Haidian	100	JSON	Valid	2022-02-11 22:12:10	China Constructi...
☐	22	CR0017	User22	Beijing Haidian	Beijing Haidian	163	XML	Valid	2022-10-21 13:23:10	Bank of China
☐	23	CR0018	User23	Beijing Haidian	Beijing Haidian	12	JSON	Valid	2022-05-21 09:12:02	Industrial and Commerci...
☐	24	CR0019	User24	Beijing Haidian	Beijing Haidian	21	JSON	Valid	2022-04-11 10:12:01	China Merchants ...



Credit report historical variables have been cached.

	ID	ReportID	Name	ComAddress	HomeAddress	Hit-count	Format	Status	CreateTime	Source
☐	16	CR0011	User16	Beijing Haidian	Beijing Haidian	13	XML	Valid	2022-04-23 23:12:00	People's Bank of ...
☐	17	CR0012	User17	Beijing Haidian	Beijing Haidian	72	JSON	Invalid	2022-05-12 13:22:30	People's Bank of ...
☐	18	CR0013	User18	Beijing Haidian	Beijing Haidian	67	JSON	Valid	2022-10-31 08:22:24	People's Bank of ...
☐	19	CR0014	User19	Beijing	Beijing aidian	67	XML	Invalid	2022-07-23 22:12:28	People's Bank of ...
☐	20				eijing idian	12	JSON	Invalid	2022-10-31 23:12:00	Agricultural Bank of ...
☐	21				eijing idian	100	JSON	Valid	2022-02-11 22:12:10	China Constructi...
☐	22				eijing idian	163	XML	Valid	2022-10-21 13:23:10	Bank of China
☐	23				eijing idian	12	JSON	Valid	2022-05-21 09:12:02	Industrial and Commerci...
☐	24				eijing idian	21	JSON	Valid	2022-04-11 10:12:01	China Merchants ...

Add Credit Report

ID:	11
ReportID:	test11
Name:	t
Company Address:	Beijing Haidian
Home Address:	Beijing Haidian
Hit Count:	10
Format:	XML
Status:	Valid
Creation Time:	2022-04-13 12:00:00
Source:	icultural Bank of China

OK Cancel

As shown in the figure, the interface features a clear structure, supporting flexible filtering and pagination. The report table uses color-coded status indicators (e.g., green for Valid, gray for Invalid) to improve visual differentiation. This design allows users to quickly identify credit records with high match frequency or data anomalies for further processing.

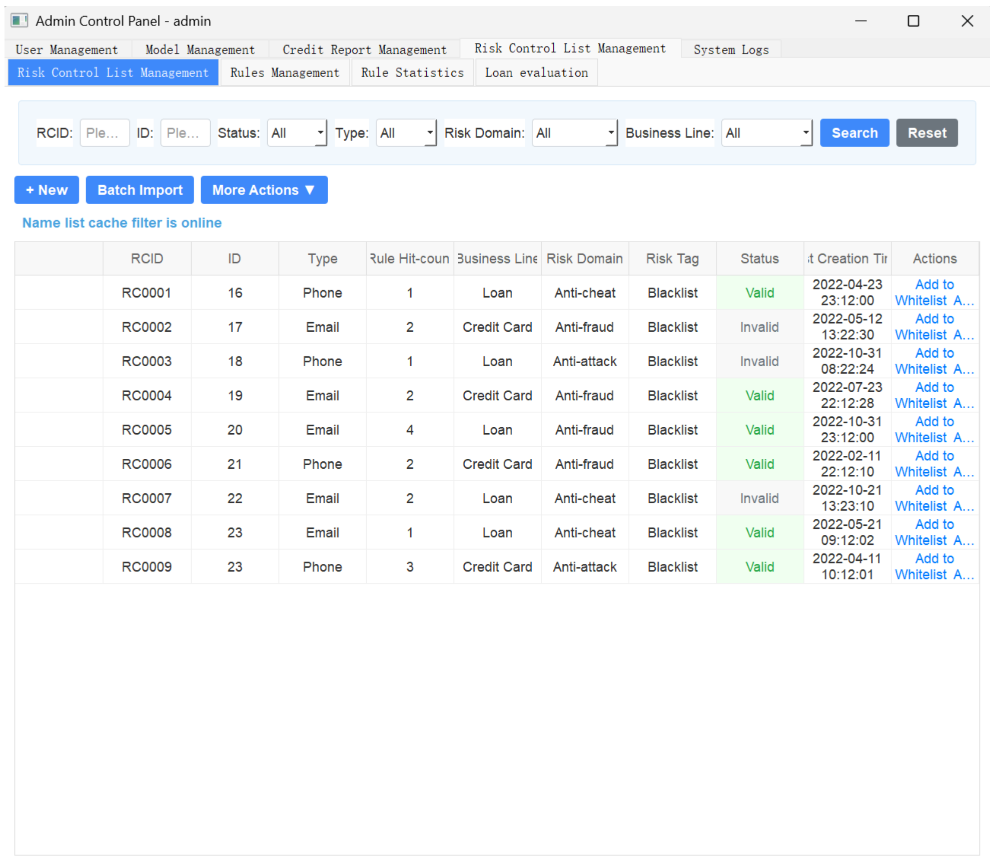
6.5 Risk Control List Management Process

The risk control list management module handles centralized management of high-risk entities such as blacklists, whitelists, and graylists. It serves as a critical support module for the execution of risk control rules and loan evaluation logic. The module enables data entry, query, filtering, status updates, and hit analysis of risk lists, facilitating preemptive risk control.

6.5.1 Risk Control List Management

In the "Risk Control List Management" interface, administrators can perform precise queries using multiple filters, including RCID (rule code), ID, type (e.g., phone, email), status, risk domain, and business line. Filtered results are displayed in a structured table format with fields such as list ID, hit type, hit count, business line, risk domain, tag category, list status, and creation time, helping administrators quickly identify high-risk entities.

On the right side of the table, users can utilize the “Add to Whitelist/Blacklist” action to move specific entries between list categories in one click, improving efficiency in list maintenance. The “+New” button at the top of the interface allows manual addition of list entries, while the “Batch Import” function supports importing multiple records at once. The “More Actions” menu reserves extension interfaces for future features such as list merging, batch updating, and multi-dimensional tagging.



RCID: Ple... ID: Ple... Status: All Type: All Risk Domain: All Business Line: All

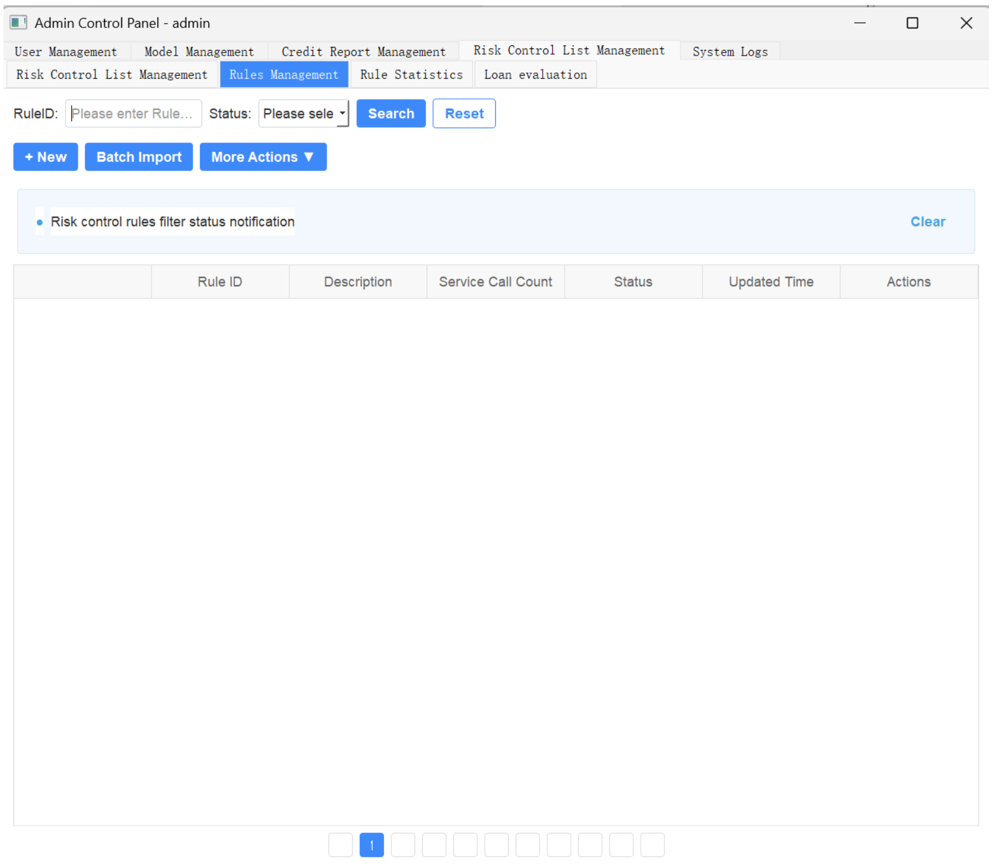
Name list cache filter is online

	RCID	ID	Type	Rule Hit-coun	Business Line	Risk Domain	Risk Tag	Status	t Creation Tir	Actions
	RC0001	16	Phone	1	Loan	Anti-cheat	Blacklist	Valid	2022-04-23 23:12:00	Add to Whitelist A...
	RC0002	17	Email	2	Credit Card	Anti-fraud	Blacklist	Invalid	2022-05-12 13:22:30	Add to Whitelist A...
	RC0003	18	Phone	1	Loan	Anti-attack	Blacklist	Invalid	2022-10-31 08:22:24	Add to Whitelist A...
	RC0004	19	Email	2	Credit Card	Anti-fraud	Blacklist	Valid	2022-07-23 22:12:28	Add to Whitelist A...
	RC0005	20	Email	4	Loan	Anti-fraud	Blacklist	Valid	2022-10-31 23:12:00	Add to Whitelist A...
	RC0006	21	Phone	2	Credit Card	Anti-fraud	Blacklist	Valid	2022-02-11 22:12:10	Add to Whitelist A...
	RC0007	22	Email	2	Loan	Anti-cheat	Blacklist	Invalid	2022-10-21 13:23:10	Add to Whitelist A...
	RC0008	23	Email	1	Loan	Anti-cheat	Blacklist	Valid	2022-05-21 09:12:02	Add to Whitelist A...
	RC0009	23	Phone	3	Credit Card	Anti-attack	Blacklist	Valid	2022-04-11 10:12:01	Add to Whitelist A...

As shown in the figure, the system presents risk list data in a tabular format. A color-coding strategy is used to distinguish list statuses (e.g., green for Valid, gray for Invalid), helping users intuitively assess the validity of each entry. Records with high hit counts can be flagged for closer monitoring, aiding manual judgment and strategy refinement.

6.5.2 Rules Management

The risk control rules management module is responsible for defining and maintaining the core logic of the system's risk control strategy. It serves as the foundation of the entire risk evaluation mechanism. Through the "Rules Management" interface, administrators can perform create, read, update, and delete (CRUD) operations on rules, enabling full lifecycle management of risk control rules.



Risk control rules filter status notification

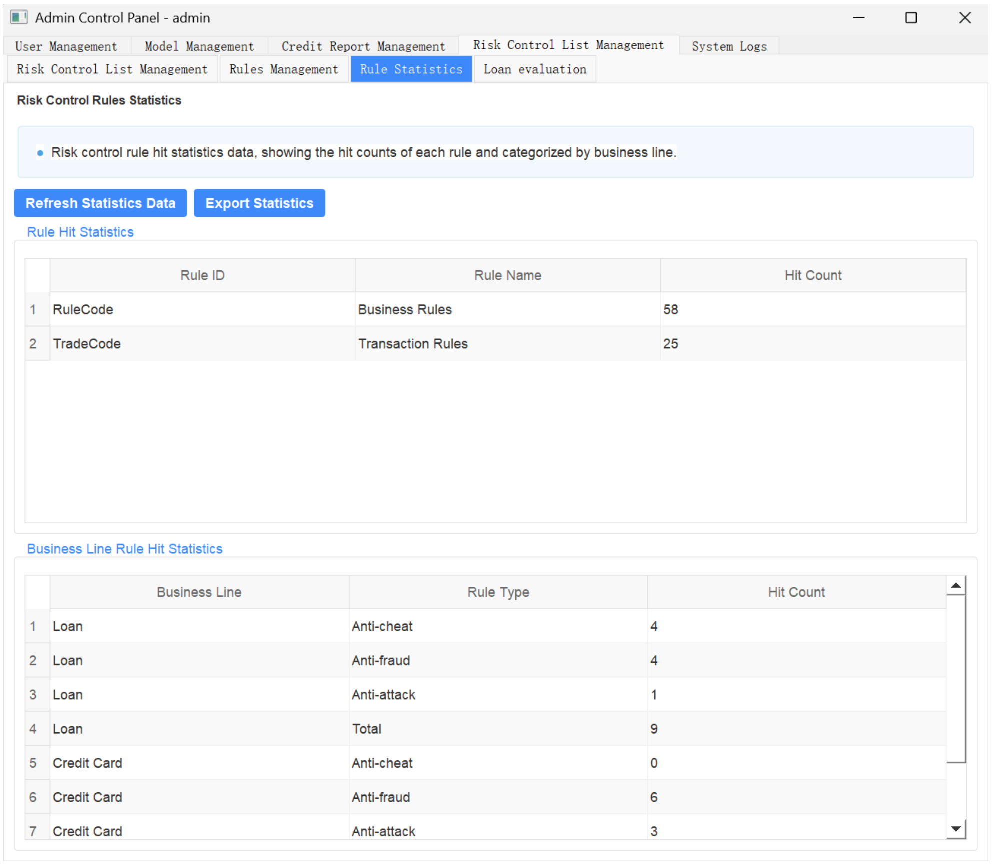
	Rule ID	Description	Service Call Count	Status	Updated Time	Actions

6.5.3 Rule Statistics

The rule statistics module provides quantitative analysis of rule hit frequency, assisting administrators in evaluating rule effectiveness and refining risk control strategies.

In the "Rule Statistics" interface, the system displays hit counts categorized by rule ID (Rule Hit Statistics), with further breakdowns by business line (e.g., loans, credit cards) under Business Line Rule Hit Statistics.

Administrators can click the “Refresh Statistics Data” button to update the displayed data or export the statistics for further analysis and reporting.



Risk Control Rules Statistics

• Risk control rule hit statistics data, showing the hit counts of each rule and categorized by business line.

Rule Hit Statistics

	Rule ID	Rule Name	Hit Count
1	RuleCode	Business Rules	58
2	TradeCode	Transaction Rules	25

Business Line Rule Hit Statistics

	Business Line	Rule Type	Hit Count
1	Loan	Anti-cheat	4
2	Loan	Anti-fraud	4
3	Loan	Anti-attack	1
4	Loan	Total	9
5	Credit Card	Anti-cheat	0
6	Credit Card	Anti-fraud	6
7	Credit Card	Anti-attack	3

The statistical table includes each rule’s hit count and classification information. By periodically reviewing this data, administrators can identify redundant or ineffective rules, thereby optimizing system performance and improving the accuracy of risk identification.

6.5.4 Loan Evaluation

The loan evaluation module represents the final decision-making stage in the financial risk control process. It integrates risk control lists, the rule engine, and user-submitted application data to perform comprehensive scoring and automatically determine approval results.

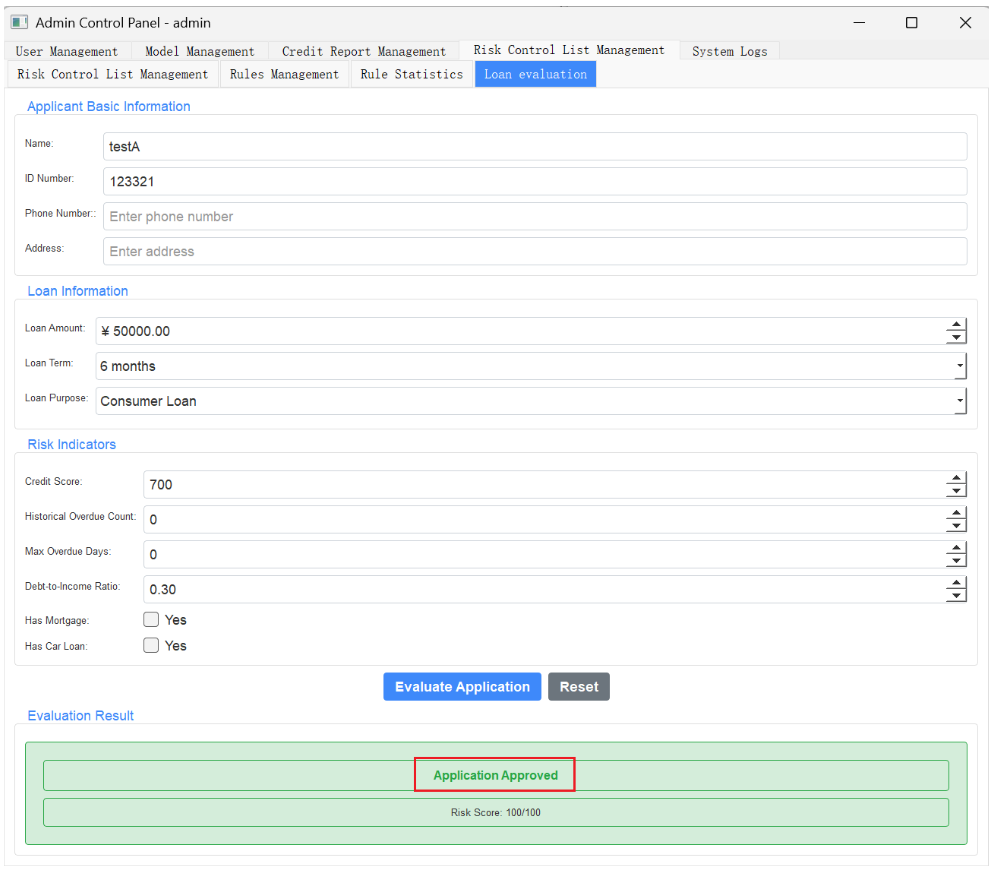
In the “Loan Evaluation” interface, administrators can input key applicant data, including:

- **Basic Information**: Name, ID number, phone number, address, etc.

- **Loan Details**: Amount, term, purpose.
- **Risk Indicators**: Credit score, number of past defaults, longest overdue days, debt ratio, existing mortgage or auto loan status.

After filling in the necessary information, clicking the "Evaluate Application" button initiates the evaluation process. The system automatically invokes active rules and uses the rule engine to determine which rules are triggered. Based on the triggered rules and their corresponding deductions, a final risk score is calculated. The evaluation starts with a base score of 100, and any score below 60 is automatically classified as "Rejected."

The result area displays whether the application is approved and lists all triggered rules along with their penalty points, ensuring full transparency of the evaluation process. If the result is rejection, the applicant's information is automatically added to the risk control list for future reference.


Admin Control Panel - admin

Applicant Basic Information

Name:	testA
ID Number:	123321
Phone Number::	Enter phone number
Address:	Enter address

Loan Information

Loan Amount:	¥ 50000.00
Loan Term:	6 months
Loan Purpose:	Consumer Loan

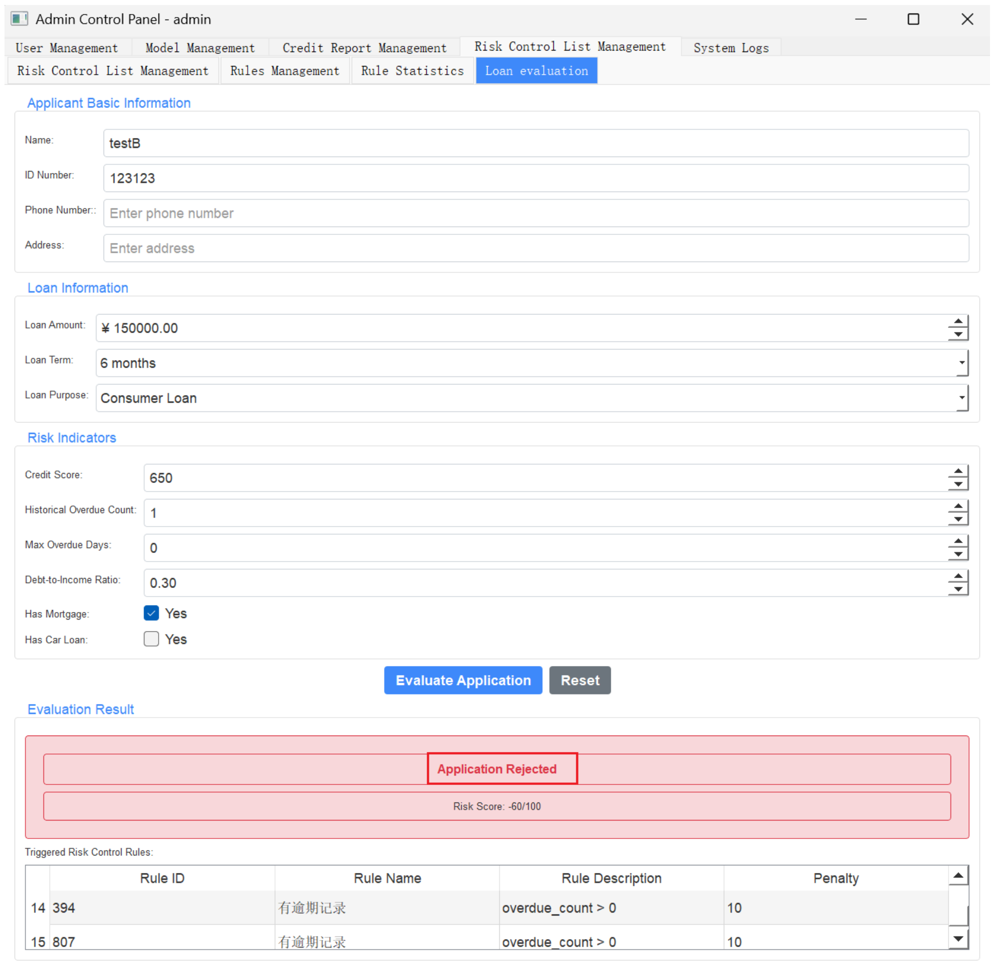
Risk Indicators

Credit Score:	700
Historical Overdue Count:	0
Max Overdue Days:	0
Debt-to-Income Ratio:	0.30
Has Mortgage:	☐ Yes
Has Car Loan:	☐ Yes

Evaluation Result

Application Approved

Risk Score: 100/100


Admin Control Panel - admin

Applicant Basic Information

Name:	testB
ID Number:	123123
Phone Number::	Enter phone number
Address:	Enter address

Loan Information

Loan Amount:	¥ 150000.00
Loan Term:	6 months
Loan Purpose:	Consumer Loan

Risk Indicators

Credit Score:	650
Historical Overdue Count:	1
Max Overdue Days:	0
Debt-to-Income Ratio:	0.30
Has Mortgage:	☑ Yes
Has Car Loan:	☐ Yes

Evaluation Result

Application Rejected

Risk Score: -60/100

Triggered Risk Control Rules:

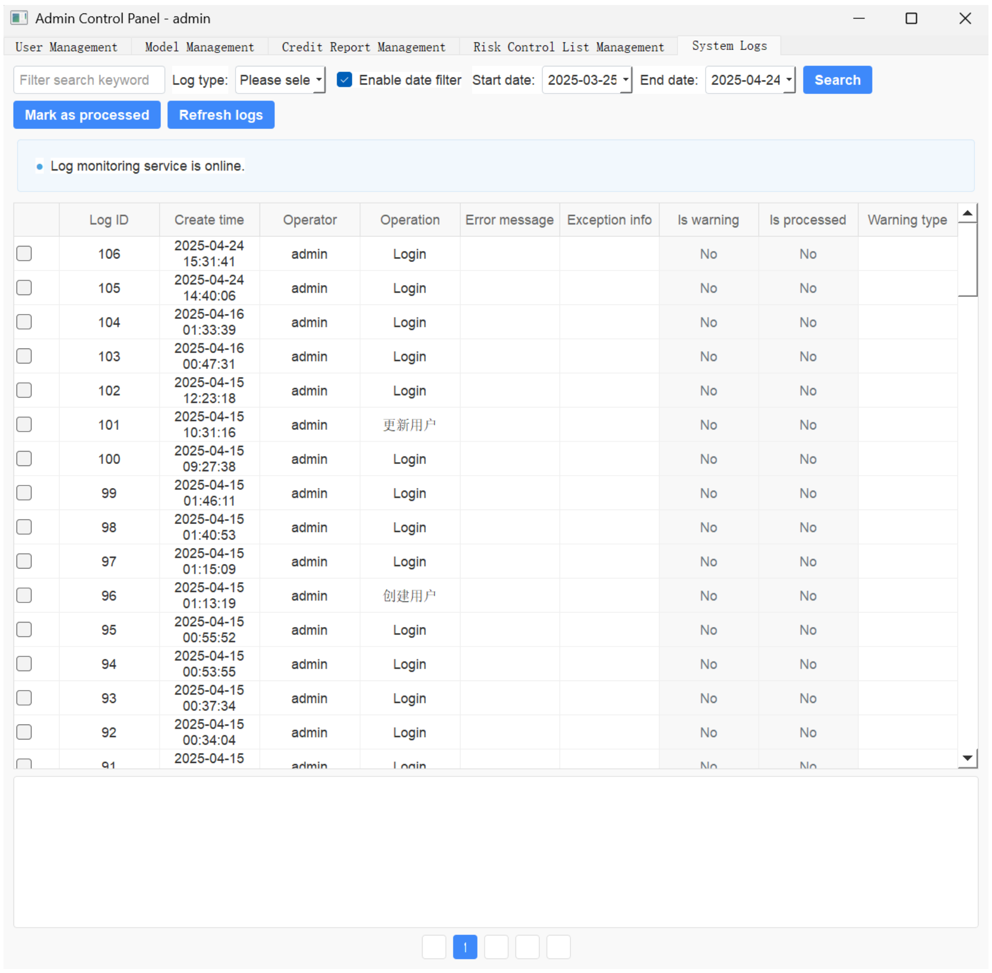
	Rule ID	Rule Name	Rule Description	Penalty
14	394	有逾期记录	overdue_count > 0	10
15	807	有逾期记录	overdue_count > 0	10

6.6 System Log Monitoring Process

The system log module provides essential support for operational recording, exception tracing, and runtime monitoring. It is a key tool for ensuring system traceability and operational stability.

This module includes log querying, categorized filtering, alert handling, and status tagging functionalities. Users can use the filter panel at the top to select log types (operation logs or runtime logs), enter keywords, and apply time filters to accurately locate target entries.

Additionally, by clicking the “Mark as processed” button, users can mark selected alert logs as resolved, supporting clearer follow-up tracking and task assignment.



Log monitoring service is online.

	Log ID	Create time	Operator	Operation	Error message	Exception info	Is warning	Is processed	Warning type
☐	106	2025-04-24 15:31:41	admin	Login			No	No	
☐	105	2025-04-24 14:40:06	admin	Login			No	No	
☐	104	2025-04-16 01:33:39	admin	Login			No	No	
☐	103	2025-04-16 00:47:31	admin	Login			No	No	
☐	102	2025-04-15 12:23:18	admin	Login			No	No	
☐	101	2025-04-15 10:31:16	admin	更新用户			No	No	
☐	100	2025-04-15 09:27:38	admin	Login			No	No	
☐	99	2025-04-15 01:46:11	admin	Login			No	No	
☐	98	2025-04-15 01:40:53	admin	Login			No	No	
☐	97	2025-04-15 01:15:09	admin	Login			No	No	
☐	96	2025-04-15 01:13:19	admin	创建用户			No	No	
☐	95	2025-04-15 00:55:52	admin	Login			No	No	
☐	94	2025-04-15 00:53:55	admin	Login			No	No	
☐	93	2025-04-15 00:37:34	admin	Login			No	No	
☐	92	2025-04-15 00:34:04	admin	Login			No	No	
☐	91	2025-04-15	admin	Login			No	No	

Through this module, administrators gain comprehensive visibility into system activities, enabling timely identification of potential risks, tracking of user behavior, and ensuring secure, compliant, and transparent platform operations.

7 Future Improvements

Building on the current system architecture and functional implementation, several key areas can be further optimized to enhance intelligence, scalability, and operational efficiency.

In terms of functionality, the system can integrate machine learning models for predictive risk assessment based on historical data, enabling proactive risk prevention. While the current version supports model file selection and import, future improvements will focus on unified access and registration of machine learning models, enhancing the management of data-driven models. Additionally, incorporating interactive dashboards, charts, and 3D visualization tools will improve data insight and support intuitive decision-making in risk strategy.

From a technical perspective, future upgrades may include transitioning to a microservices architecture with front-end/back-end decoupling to improve modularity and deployment flexibility. Containerization and cloud-native technologies will further enable efficient resource management and elastic scalability. Integrating stream processing platforms and big data frameworks will significantly strengthen the system's capability to process real-time data and respond to dynamic risks.

8 Conclusion

This report presents a comprehensive overview of the development of an internet finance risk control system based on the MVC architecture, including project requirements, module analysis, software design, operational principles, implementation examples, and future improvement directions.

The system follows the MVC design pattern, which cleanly separates data access, business logic, and UI presentation, thus improving maintainability and extensibility. Key modules include user authentication, user management, risk list management, rule configuration, model management, loan evaluation, and log monitoring, forming a complete risk control framework.

Core functionalities of the system include identity verification, permission control, user data management, rule engine evaluation, model lifecycle management, loan risk scoring, and log-based monitoring and alerting. Through a multi-layered defense approach, the system effectively supports risk identification and decision-making.

The implementation utilizes Python and PyQt5 for an intuitive user interface and interaction, with SQLite and PostgreSQL supporting deployments of various scales. Custom connection pooling and transaction management optimize database performance.

Despite achieving core functions, the system still offers room for enhancement. Security can be strengthened through better password protection, refined permission control, and sensitive data handling. The user interface can be improved for a smoother experience and enhanced visualization. Functionally, the rule engine can be upgraded, machine learning integration expanded, and risk strategies further optimized. Performance-wise, improvements to database access, caching, concurrency handling, and critical code paths can be pursued for greater efficiency.