Software Requirements Specification

for

Rikugan

A Bounty-Based Project Management System

Version 1.0

Prepared by CSCI3100 Development Team

December 20, 2025

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document specifies the requirements for Rikugan, version 1.0. The system is designed as a bounty-based project management application that combines traditional Kanban board functionality with gamified task assignment and reward mechanisms. This SRS covers the complete web-based application including user authentication, task management, role-based access control, and reward tracking systems.

1.2 Document Conventions

This document follows the IEEE 830-1998 standard for software requirements specifications. The following typographical conventions are used:

- **Bold text** indicates system components, user interface elements, and important terms
- *Italic text* represents user actions and system responses
- `Monospace text` denotes code elements, database fields, and technical specifications
- Requirements are numbered using the format REQ-XX for functional requirements and NFR-XX for non-functional requirements
- Priority levels are classified as High (essential for system operation), Medium (important but not critical), and Low (nice-to-have features)

1.3 Intended Audience and Reading Suggestions

This SRS is intended for the following audiences:

- **Developers and Software Engineers**: Should focus on Section 4 (System Features) and Section 3 (External Interface Requirements) for implementation details
- **Project Managers**: Should review Section 2 (Overall Description) and Section 6 (Other Requirements) for project scope and constraints
- **Quality Assurance Testers**: Should concentrate on Section 4 for functional requirements and Section 5 for non-functional requirements
- **End Users**: Should review Section 2 for system overview and user role descriptions
- **Course Instructors**: Should examine all sections to evaluate project completeness and feasibility

The document is organized to provide a high-level overview first, followed by detailed technical specifications. Readers are recommended to start with Section 2 (Overall Description) before proceeding to their area-specific sections.

1.4 Product Scope

Rikugan is a web-based application designed to revolutionize project management through gamification and role-based task assignment. The system addresses the need for better organization and resource allocation in software development projects by implementing a bounty-based reward system.

Key Benefits and Objectives:

- Improved task visibility and progress tracking through Kanban-style boards
- Increased developer motivation through gamified bounty rewards
- Clear role-based hierarchies with appropriate access controls
- Streamlined project communication and collaboration
- Efficient resource allocation and deadline management

The system supports three distinct user roles (Goons, Hashira, and Oyakatasama) with escalating privileges and responsibilities, creating a structured environment that promotes both individual accountability and team collaboration.

1.5 References

1. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
2. React Documentation, `https://reactjs.org/docs/`
3. Vite Build Tool Documentation, `https://vitejs.dev/`
4. HeroUI Component Library, `https://heroui.com/`
5. MySQL Database Documentation, `https://dev.mysql.com/doc/`
6. Docker Documentation, `https://docs.docker.com/`
7. Kanban Methodology Guide, Agile Alliance, 2023

2 Overall Description

2.1 Product Perspective

Rikugan is a new, self-contained web-based application designed specifically for project management in software development environments. The system draws inspiration from popular task management tools like Trello and Jira but incorporates unique gamification elements through a bounty-based reward system.

2.1.1 System Context Diagram

The system architecture consists of the following components:

- **Web Frontend** (React/Vite): User interface components
- **Backend API** (Node.js): RESTful API services
- **MySQL Database**: Data persistence layer
- **Docker Runtime Environment**: Container orchestration

2.2 Product Functions

Rikugan provides the following major functions:

2.2.1 Authentication & Authorization

- Secure user registration and login
- Role-based access control (Goons, Hashira, Oyakatasama)
- License-based system access management

2.2.2 Task Management

- Create and manage task bounties with monetary rewards
- Kanban-style task boards with status tracking
- Task assignment and progression monitoring
- Deadline management with penalty systems

2.2.3 User Management

- Profile management with earnings tracking
- Performance monitoring and reward history
- Hierarchical user role administration

2.2.4 Notification System

- Real-time notifications for task assignments
- Deadline reminders and status updates
- Achievement and reward notifications

2.2.5 Reporting & Analytics

- Task completion statistics
- User performance metrics
- Project progress visualization

2.3 User Classes and Characteristics

2.3.1 Goons (Junior Programmers)

- *Frequency of Use:* Daily active users
- *Technical Expertise:* Basic to intermediate programming skills
- *Primary Functions:* Task selection, status updates, profile viewing
- *Characteristics:* Entry-level developers seeking skill improvement and monetary rewards
- *Security Level:* Basic user permissions

2.3.2 Hashira (Senior Programmers)

- *Frequency of Use:* Daily active users
- *Technical Expertise:* Advanced programming and project management skills
- *Primary Functions:* Task creation, team monitoring, all Goon functions
- *Characteristics:* Experienced developers with leadership responsibilities
- *Security Level:* Elevated permissions for task and team management

2.3.3 Oyakatasama (Administrators)

- *Frequency of Use:* Regular but less frequent than other roles
- *Technical Expertise:* System administration and project oversight
- *Primary Functions:* System administration, license management, user account control
- *Characteristics:* Project owners and system administrators
- *Security Level:* Full system administrative privileges

2.4 Operating Environment

2.4.1 Client-Side Requirements

- Modern web browsers (Chrome 90+, Firefox 88+, Safari 14+, Edge 90+)
- JavaScript enabled
- Minimum screen resolution: 1024x768
- Stable internet connection

2.4.2 Server-Side Environment

- Node.js runtime environment (v16+ recommended)
- MySQL database server (v8.0+)
- Docker containerization platform
- Linux/Unix-based hosting environment (Ubuntu 20.04 LTS recommended)
- Minimum 4GB RAM, 20GB storage space

2.4.3 Development Environment

- React 18+ with Vite build system
- HeroUI component library
- Modern JavaScript (ES2020+)

2.5 Design and Implementation Constraints

2.5.1 Technical Constraints

- Must be implemented as a web application using React and Vite
- Database must use MySQL for data persistence
- Must utilize Docker for development environment consistency
- Frontend must use HeroUI component library for consistent styling

2.5.2 Regulatory and Policy Constraints

- Must implement secure authentication and authorization
- User data must be protected according to basic privacy standards

2.5.3 Resource Constraints

- Development timeline: One month maximum
- Team size: Undergraduate student project (5 developers)
- Budget: Educational project with no commercial constraints
- Hosting: Development environment only (no production deployment required)

2.5.4 Interface Constraints

- Must be responsive for desktop and tablet devices
- Must follow accessibility guidelines for basic usability

2.6 User Documentation

The following documentation will be delivered with the system:

2.6.1 End-User Documentation

- Demo Video of Regular usage

2.6.2 Technical Documentation

- API documentation for backend services
- Database schema documentation
- Installation and deployment guide
- Developer setup instructions

2.6.3 Administrative Documentation

- System administration guide
- User account management procedures
- Backup and recovery procedures

2.7 Assumptions and Dependencies

2.7.1 Assumptions

- Users have basic familiarity with web applications and project management concepts
- Development team has access to modern development tools and environments
- MySQL database will be available and properly configured
- Docker environment can be successfully set up for development

2.7.2 Dependencies

- React framework and associated npm packages
- HeroUI component library availability and compatibility
- MySQL database server functionality
- Docker container runtime environment
- Node.js runtime for backend development
- Vite build tool for frontend compilation

2.7.3 Risk Factors

- Potential learning curve for team members unfamiliar with React or MySQL
- Docker environment setup complexity on different operating systems
- Time constraints may limit advanced feature implementation
- Third-party component library updates could affect compatibility

3 External Interface Requirements

3.1 User Interfaces

Rikugan shall provide a responsive web-based user interface built with React and HeroUI components.

3.1.1 General UI Requirements

- Clean, modern design following Material Design principles
- Responsive layout supporting desktop (1024px+) and tablet (768px+) viewports
- Consistent navigation with role-based menu options
- Loading states and error handling for all user actions
- Accessibility compliance with WCAG 2.1 Level A standards

3.1.2 Login/Authentication Interface

- Login form with username/email and password fields
- ”Remember me” checkbox for session persistence
- Password reset functionality
- Role-based redirection after successful login

3.1.3 Dashboard Interface (Role-specific)

- **Goon Dashboard**: Task selection grid, personal statistics, current tasks
- **Hashira Dashboard**: Task management panel, team overview
- **Oyakatasama Dashboard**: System administration, user management

3.1.4 Task Management Interface

- Kanban board with drag-and-drop functionality
- Task cards displaying title, bounty amount, deadline, and status
- Task detail modal with description, comments, and status updates
- Task creation form (Hashira only) with all required fields

3.1.5 Profile Interface

- User profile page with editable personal information
- Earnings history and statistics
- Achievement badges and milestones
- Task completion history with filtering options

3.2 Hardware Interfaces

As a web-based application, Rikugan has minimal direct hardware interface requirements:

3.2.1 Client-Side Hardware

- Standard web browser capable of running modern JavaScript (ES2020+)
- Minimum 4GB RAM for optimal browser performance
- Network interface for internet connectivity
- Display resolution of at least 1024x768 pixels
- Input devices: keyboard and mouse/touchpad (touch support for tablets)

3.2.2 Server-Side Hardware

- x86_64 processor architecture (Intel/AMD)
- Minimum 4GB RAM for development environment
- 20GB available storage space
- Network interface for client connections
- Compatible with standard virtualization platforms (Docker)

3.2.3 No Special Hardware Requirements

- No specialized input devices required
- No direct hardware control interfaces
- No real-time hardware monitoring needs
- Standard commodity hardware is sufficient

3.3 Software Interfaces

3.3.1 Frontend Framework Integration

- **React 18+**: Core frontend framework for component-based UI development
- **Vite**: Build tool and development server for fast compilation and hot reload
- **HeroUI**: Component library providing pre-built UI components and theming

3.3.2 Backend Integration

- **Node.js Runtime**: JavaScript runtime environment for backend services
- **Express.js**: Web application framework for RESTful API development
- **JWT (JSON Web Tokens)**: Authentication and authorization token management

3.3.3 Database Interface

- **MySQL 8.0+**: Primary database for persistent data storage
- **Database Schema**: Tables for users, tasks, bounties, notifications, and licenses
- **Connection Pooling**: Efficient database connection management
- **Data Validation**: Server-side validation for all database operations

3.3.4 Development Tools Integration

- **Docker**: Containerization for consistent development environment
- **Docker Compose**: Multi-container application orchestration
- **npm/yarn**: Package management for JavaScript dependencies
- **ESLint**: Code quality and style enforcement

3.3.5 API Specifications

- RESTful API design with JSON data format
- Standard HTTP methods (GET, POST, PUT, DELETE)
- API versioning through URL path (/api/v1/)
- Consistent error response format with HTTP status codes

3.3.6 External Libraries

- Password hashing: bcrypt for secure password storage
- Date manipulation: date-fns or moment.js for deadline calculations
- Validation: Joi or Yup for input validation schemas
- Logging: Winston for structured application logging

3.4 Communications Interfaces

3.4.1 HTTP/HTTPS Protocol

- Primary communication via HTTP/HTTPS for web browser to server communication
- HTTPS required for production deployment (TLS 1.2 minimum)
- Standard HTTP methods with appropriate status codes
- Content-Type: application/json for API responses

3.4.2 RESTful API Communication

- Base URL structure: `https://domain.com/api/v1/`
- Authentication via Bearer tokens in Authorization header
- Request/Response format: JSON with UTF-8 encoding
- Rate limiting: 1000 requests per hour per user

3.4.3 API Endpoints Structure

```
GET    /api/v1/auth/login            - User authentication
POST   /api/v1/auth/logout           - User logout
GET    /api/v1/users/profile         - Get user profile
PUT    /api/v1/users/profile         - Update user profile
GET    /api/v1/tasks                 - Get tasks list
POST   /api/v1/tasks                 - Create new task
GET    /api/v1/tasks/{id}            - Get specific task
PUT    /api/v1/tasks/{id}/status     - Update task status
GET    /api/v1/notifications         - Get user notifications
```

3.4.4 Real-time Communication (Future Enhancement)

- WebSocket support for real-time notifications
- Server-Sent Events for live dashboard updates
- Push notification capability for mobile browsers

3.4.5 Security Protocols

- JWT tokens for stateless authentication
- CORS (Cross-Origin Resource Sharing) configuration for frontend-backend communication
- Input sanitization to prevent XSS and SQL injection
- Request validation and error handling

4 System Features

4.1 User Authentication and Authorization

4.1.1 Description and Priority

Priority: High - Essential for system security and role-based access control

This feature provides secure user registration, login, and role-based access control for the three user types: Goons, Hashira, and Oyakatasama. The system implements license-based access management to control system access.

4.1.2 Stimulus/Response Sequences

1. User attempts to access the system → System redirects to login page if not authenticated
2. User enters credentials → System validates and grants role-appropriate access
3. Admin creates new user account → System generates credentials and assigns role
4. User attempts unauthorized action → System denies access and logs attempt

4.1.3 Functional Requirements

REQ-1: User Registration The system shall allow administrators (Oyakatasama) to create new user accounts with the following information:

- Username (unique, 3-50 characters)
- Email address (valid format)
- Initial password (minimum 8 characters)
- Assigned role (Goon, Hashira, or Oyakatasama)
- License allocation for system access

REQ-2: User Login The system shall authenticate users using username/email and password combination and maintain session state for 8 hours of inactivity.

REQ-3: Role-Based Access Control The system shall enforce the following access levels:

- **Goons**: Task selection, status updates, profile viewing
- **Hashira**: All Goon functions plus task creation and team monitoring
- **Oyakatasama**: All functions plus user management and system administration

REQ-4: License Management The system shall implement a license-based access control where only users with valid licenses can access the system. Without a valid license, the entire project shall be limited to creating no more than 3 tasks total.

4.2 Task Management System

4.2.1 Description and Priority

Priority: High - Core functionality for project management

This feature implements the bounty-based task management system with Kanban-style boards, task assignment, and progress tracking.

4.2.2 Stimulus/Response Sequences

1. Hashira creates new task with bounty → System adds task to available tasks pool
2. Goon selects task → System assigns task and moves to "In Progress" status
3. User updates task status → System updates Kanban board and notifies stakeholders
4. Task reaches deadline → System applies penalties if incomplete

4.2.3 Functional Requirements

REQ-5: Task Creation The system shall allow Hashira to create tasks with the following attributes:

- Task name (required, max 100 characters)
- Description (required, max 1000 characters)
- Bounty amount (required, positive number)
- Deadline (required, future date)
- Priority level (High, Medium, Low)
- Required skills/tags

REQ-6: Task Assignment The system shall allow Goons to select and join available tasks, with automatic assignment to the first qualified user who selects the task.

REQ-7: Kanban Board The system shall display tasks in a Kanban board with the following columns:

- Available
- In Progress
- Review
- Completed

REQ-8: Task Status Updates The system shall allow assigned users to update task status and add progress comments.

REQ-9: Deadline Management The system shall track task deadlines and apply monetary penalties for missed deadlines as defined by system configuration.

REQ-10: Task Deletion The system shall allow Hashira and Oyakatasama to delete tasks from the board. Tasks can only be deleted if they are in "Available" status or if no user is currently assigned to them.

4.3 User Profile and Reward System

4.3.1 Description and Priority

Priority: Medium - Important for gamification and user engagement

This feature manages user profiles, tracks earnings, and implements the reward/penalty system.

4.3.2 Stimulus/Response Sequences

1. User completes task → System awards bounty to user's account
2. User misses deadline → System applies penalty from user's account
3. User views profile → System displays current balance and task history
4. User achieves milestone → System displays achievement notification

4.3.3 Functional Requirements

REQ-11: User Profile Management The system shall maintain user profiles containing:

- Personal information (username, email, role)
- Current monetary balance
- Task completion history
- Performance statistics

REQ-12: Bounty Reward System The system shall automatically credit user accounts with bounty amounts upon successful task completion.

REQ-13: Penalty System The system shall apply configurable monetary penalties for missed deadlines or unsatisfactory task completion.

REQ-14: Performance Tracking The system shall track and display user performance metrics including:

- Tasks completed
- Average completion time
- Success rate
- Total earnings

4.4 Notification System

4.4.1 Description and Priority

Priority: Medium - Important for user engagement and communication

This feature provides real-time notifications for task assignments, deadlines, and system events.

4.4.2 Stimulus/Response Sequences

1. Task assigned to user → System sends notification to user
2. Deadline approaching → System sends reminder notification
3. Task status changed → System notifies relevant stakeholders
4. User achieves milestone → System sends congratulatory notification

4.4.3 Functional Requirements

REQ-15: Task Notifications The system shall notify users when:

- New tasks are assigned to them
- Task deadlines are approaching (24 hours before)
- Task status is updated by team members
- Tasks are completed or cancelled

REQ-16: Achievement Notifications The system shall notify users of achievements such as:

- First task completion
- Earning milestones
- Performance improvements
- Role promotions

5 Other Non-Functional Requirements

5.1 Performance Requirements

5.1.1 Response Time Requirements

- Page loading: All pages must load within 3 seconds under normal network conditions
- API response time: 95% of API calls must respond within 500ms
- Database queries: Complex queries must execute within 2 seconds
- User authentication: Login process must complete within 1 second

5.1.2 Throughput Requirements

- Concurrent users: System must support up to 50 concurrent users (suitable for undergraduate project scope)
- Task operations: System must handle 100 task status updates per minute
- Notification delivery: Real-time notifications must be delivered within 5 seconds

5.1.3 Scalability Requirements

- User capacity: System designed to handle up to 200 registered users
- Task capacity: Support for up to 1000 active tasks simultaneously

5.1.4 Resource Usage

- Memory usage: Frontend application should use less than 100MB RAM per browser tab
- Storage efficiency: Database size should not exceed 1GB for typical usage
- CPU utilization: Server CPU usage should remain below 70% under normal load

5.2 Safety Requirements

5.2.1 User Safety Measures

- Prevent unauthorized access to sensitive user information
- Implement session timeout to protect unattended sessions
- Provide clear confirmation dialogs for destructive actions (task deletion, account removal)
- Sanitize all user inputs to prevent malicious code execution

5.3 Security Requirements

5.3.1 Authentication & Authorization

- Implement secure password policies (minimum 8 characters, complexity requirements)
- Use industry-standard password hashing (bcrypt with salt)
- Implement JWT-based authentication with token expiration
- Enforce role-based access control for all system functions
- Provide secure logout functionality that invalidates tokens

5.3.2 Data Security

- Encrypt sensitive data at rest (user credentials, personal information)
- Use HTTPS for all client-server communications
- Implement SQL injection prevention through parameterized queries
- Sanitize all user inputs to prevent XSS attacks
- Validate and escape output data before rendering

5.3.3 System Security

- Regular security updates for all dependencies
- Implement rate limiting to prevent brute force attacks
- Monitor and log security-relevant events
- Implement proper error handling that doesn't leak sensitive information

5.4 Software Quality Attributes

5.4.1 Usability

- Intuitive interface requiring minimal training for new users
- Consistent navigation and interaction patterns across all pages
- Clear error messages with actionable guidance
- Responsive design supporting multiple screen sizes
- Accessibility compliance with WCAG 2.1 Level A standards

5.4.2 Reliability

- System uptime of 95% during active development and testing phases
- Graceful degradation when external services are unavailable
- Automatic error recovery for transient failures
- Data consistency maintained across all operations

5.4.3 Maintainability

- Modular code structure following React best practices
- Comprehensive API documentation
- Clear separation of concerns between frontend and backend
- Standardized coding conventions with ESLint enforcement
- Unit test coverage of at least 70% for critical functions

5.4.4 Portability

- Cross-browser compatibility (Chrome, Firefox, Safari, Edge)
- Operating system independence through web-based deployment
- Docker containerization for consistent deployment environments
- Database abstraction to support potential MySQL alternatives

5.4.5 Scalability

- Modular architecture supporting incremental feature additions
- Efficient database schema design for future growth
- Component-based frontend architecture for easy extension
- RESTful API design supporting future mobile applications

5.4.6 Testability

- Unit testing capabilities for all major functions
- API endpoints designed for automated testing
- Test data generation and cleanup procedures
- Clear separation between business logic and UI components

5.5 Business Rules

5.5.1 User Role Hierarchy

- Only Oyakatasama can create, modify, or delete user accounts
- Only Hashira and Oyakatasama can create new task bounties
- Only Goons and Hashira can select and work on tasks
- Users can only update status of tasks assigned to them
- Users cannot exceed their available balance when penalties are applied

5.5.2 Task Management Rules

- Each task can only be assigned to one user at a time
- Task bounties must be positive monetary values
- Task deadlines must be future dates when created
- Completed tasks cannot be modified or reassigned
- Task status progression follows: Available $\rightarrow$ In Progress $\rightarrow$ Review $\rightarrow$ Completed

5.5.3 Financial Rules

- Bounty payments are processed automatically upon task completion
- Deadline penalties are applied automatically when deadlines are missed
- Users cannot have negative account balances below -$100
- All financial transactions must be logged with timestamps
- Only administrators can manually adjust user account balances

5.5.4 License and Access Rules

- All users must have valid licenses to access the system
- License validation occurs at every login attempt
- Expired licenses prevent system access until renewed
- Only Oyakatasama can issue, renew, or revoke licenses

5.5.5 Notification Rules

- Task deadline reminders sent 24 hours before due date
- Assignment notifications sent immediately upon task selection
- Achievement notifications sent upon reaching milestones
- Administrative notifications have higher priority than user notifications

6 Other Requirements

6.1 Database Requirements

6.1.1 Database Management System

- MySQL 8.0 or higher for primary data storage
- InnoDB storage engine for transaction support and data integrity
- UTF-8 character encoding for international character support
- Automated backup scheduling (daily for development environment)

6.1.2 Database Schema Requirements

- User table: id, username, email, password_hash, role, balance, created_at, updated_at
- Tasks table: id, title, description, bounty_amount, deadline, status, created_by, assigned_to, created_at, updated_at
- Notifications table: id, user_id, type, message, read_status, created_at
- Licenses table: id, user_id, license_key, expiry_date, status, created_at

6.1.3 Data Integrity Requirements

- Foreign key constraints for data relationships
- Unique constraints for usernames and email addresses
- Check constraints for positive bounty amounts and future deadlines
- Triggers for automatic timestamp updates

6.1.4 Database Performance Requirements

- Index on frequently queried columns (user_id, task_status, created_at)
- Query execution time optimization for complex joins
- Connection pooling for efficient resource utilization

6.2 Development Environment Requirements

6.2.1 Container Requirements

- Docker Engine 20.0+ for containerization
- Docker Compose for multi-service orchestration
- Separate containers for frontend, backend, and database services
- Volume mounting for persistent data storage during development

6.2.2 Development Tools

- Node.js 16+ runtime environment
- npm or yarn package manager
- Git version control with branching strategy
- Code editor with ESLint and Prettier integration

6.2.3 Build and Deployment

- Vite build system for frontend compilation
- Hot module replacement for development efficiency
- Environment-specific configuration files (.env)
- Automated testing pipeline with npm scripts

6.3 Legal and Compliance Requirements

6.3.1 Educational Use

- System designed for educational purposes only
- No commercial use or real monetary transactions
- Compliance with university academic integrity policies
- Open source libraries with compatible licenses

6.3.2 Data Privacy

- No collection of sensitive personal information beyond username and email
- Local development environment only (no cloud deployment for student project)
- User consent for data collection through registration process
- Right to data deletion upon account termination

6.3.3 Intellectual Property

- Original code development by student team
- Proper attribution of third-party libraries and components
- MIT or similar permissive license for project code
- No use of proprietary or copyrighted assets

6.4 Documentation Requirements

6.4.1 Technical Documentation

- API documentation with request/response examples
- Database schema documentation with entity relationships
- Setup and installation guide for development environment
- Code documentation with JSDoc comments for complex functions

6.4.2 User Documentation

- User manual with role-specific feature descriptions
- Quick start guide for new users
- FAQ section addressing common issues
- Screenshot-based tutorials for key workflows

6.4.3 Project Documentation

- Software Requirements Specification (this document)
- Project timeline and milestone tracking
- Team member responsibilities and contact information
- Version control and branching strategy documentation

6.5 Internationalization Requirements

6.5.1 Language Support

- English as primary language for user interface
- UTF-8 character encoding for future multi-language support
- Externalized text strings for easy translation
- Date and time formatting using locale-aware libraries

6.5.2 Future Considerations

- Modular text resources for potential localization
- Currency formatting flexibility for international use
- Time zone handling for distributed teams

6.6 Testing Requirements

6.6.1 Unit Testing

- Jest framework for JavaScript unit tests
- Test coverage of at least 70% for critical business logic
- Mock dependencies for isolated testing
- Automated test execution in CI/CD pipeline

6.6.2 Integration Testing

- API endpoint testing with appropriate test data
- Database integration testing with test database
- Component integration testing for React components
- End-to-end testing for critical user workflows

6.6.3 Test Data Management

- Seed data for consistent testing environments
- Test user accounts for different role types
- Sample tasks and bounties for testing scenarios
- Database cleanup procedures between test runs