License Management System

Comprehensive Knowledge Transfer Document

> *Document Version: 2.0*
> *Last Updated: June 5, 2025*
> *Prepared by: [Your Name]*

Table of Contents

> **Quick Navigation**
> System Overview • Architecture • Admin Guide • User Guide • Troubleshooting

---

System Overview

Introduction

The **License Management System (LMS)** is an enterprise-grade solution designed to streamline software license management, user access control, and product activation processes. This system provides organizations with complete control over their software distribution, ensuring compliance with licensing agreements while offering a seamless experience for both administrators and end-users.

Business Value

- **Revenue Protection**: Prevent unauthorized software usage and license sharing
- **Operational Efficiency**: Automate license provisioning and management
- **Compliance**: Maintain audit trails and ensure software license compliance
- **User Experience**: Provide self-service options for users while maintaining security
- **Business Intelligence**: Gain insights into software usage patterns and license utilization

Key Features

Feature	Description	Business Benefit
User Authentication	Secure login with role-based access control	Ensures only authorized personnel can manage licenses
License Generation	Automated key generation with customizable templates	Reduces manual work and errors
Dashboard	Real-time insights and analytics	Enables data-driven decision making
Activation Tracking	Monitor license activations and deactivations	Prevents unauthorized usage
User Management	Centralized user and role administration	Simplifies access control
Self-Service Portal	Users can manage their own licenses	Reduces IT support load
Reporting	Comprehensive reporting and audit trails	Ensures compliance and provides insights

System Requirements

- **Server**: Node.js 14+ with MongoDB 4.4+
- **Client**: Modern web browser (Chrome, Firefox, Edge, Safari)
- **Network**: Internet connection for license validation
- **Storage**: Minimum 1GB free disk space for database

System Architecture

High-Level Architecture

```
graph TD
    A[Client Applications] <--> B[Frontend - React.js]
    B <--> C[Backend API - Node.js/Express]
    C <--> D[(MongoDB Database)]

    E[Admin Users] --> B
    F[End Users] --> B
    G[System Integrations] --> C

    H[License Validation] --> C

    subgraph "Client Environment"
    I[License Manager Library] <--> H
    end

    style A fill:#e1f5fe,stroke:#0288d1
    style B fill:#e8f5e9,stroke#2e7d32
    style C fill#e8eaf6,stroke#3949ab
    style D fill#f3e5f5,stroke#8e24aa
    style E fill#fff3e0,stroke#f57c00
    style F fill#fff3e0,stroke#f57c00
    style G fill#f3e5f5,stroke#8e24aa
    style H fill#e8f5e9,stroke#2e7d32
    style I fill#e1f5fe,stroke#0288d1
```

Component Architecture

1. Frontend Layer (React.js)

Key Components:

- **Dashboard Module**: Real-time metrics and quick actions
- **User Management**: CRUD operations for user accounts
- **License Administration**: Generate, view, and manage licenses

- **Product Catalog**: Manage software products and versions
- **Reporting Engine**: Generate usage and compliance reports

Key Features:

- Responsive design for all devices
- Real-time updates using WebSockets
- Role-based access control (RBAC)
- Audit logging for all actions

2. Backend Layer (Node.js/Express)

Core Services:

Service	Description	Key Endpoints
Auth Service	Handles authentication & authorization	`/api/auth/*`
User Service	Manages user accounts and roles	`/api/users/*`
License Service	License generation and validation	`/api/licenses/*`
Product Service	Product catalog management	`/api/products/*`
Activation Service	Handles license activations	`/api/activations/*`

Key Features:

- RESTful API design
- JWT-based authentication
- Request validation and sanitization
- Rate limiting and security headers
- Comprehensive error handling

3. Data Layer (MongoDB)

Collections Schema:

```
// Users Collection
{
  _id: ObjectId,
  email: String,
  password: String,
  role: { type: String, enum: ['admin', 'user'] },
  createdAt: Date,
  lastLogin: Date
}

// Products Collection
{
  _id: ObjectId,
  name: String,
  version: String,
  description: String,
  features: [String],
  pricing: {
    type: { type: String, enum: ['one-time', 'subscription'] },
    amount: Number,
    currency: String
  },
  activationLimit: Number
}

// Licenses Collection
{
  _id: ObjectId,
  key: String,
  productId: ObjectId,
  userId: ObjectId,
  status: { type: String, enum: ['active', 'suspended', 'revoked'] },
  activations: [{
    deviceId: String,
    activatedAt: Date,
    lastValidated: Date,
    ipAddress: String
  }],
  validFrom: Date,
  validUntil: Date
}
```

4. Client Library

Key Features:

- Machine fingerprinting
- Offline validation

- Tamper detection
- Graceful degradation
- Automatic reactivation

Integration Example:

```
 const license = new LicenseManager({
  apiUrl: 'https://api.yourdomain.com',
  productId: 'your-product-id',
  storage: 'localStorage', // or 'file' for Node.js
  checkInterval: 3600 // Check every hour
});

// Check license status
const { valid, message } = await license.validate();
if (!valid) {
  console.error('License validation failed:', message);
  // Handle invalid license
}
```

Data Flow

1. **License Activation**

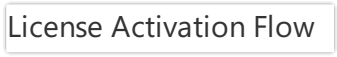

Figure 5: License Activation Sequence

Process Flow:

1. Client app sends activation request to frontend
2. Frontend forwards request to backend API
3. Backend validates license in database
4. Database returns license data
5. Backend records activation
6. Success/failure response is returned to client

2. **License Validation**

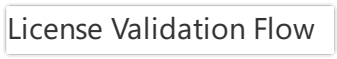

Figure 6: License Validation Sequence

Process Flow:

1. Client sends validation request with machine ID
2. Backend checks activation status in database
3. If valid: Returns success with expiration date
4. If invalid: Returns error with reason

Security Considerations

- **Data Encryption**: All sensitive data encrypted at rest and in transit
- **Rate Limiting**: Protection against brute force attacks
- **Input Validation**: Protection against injection attacks
- **CORS**: Properly configured CORS policies
- **Audit Logging**: All sensitive operations are logged
- **Secure Storage**: License data stored securely on client devices

Technical Stack

Frontend

- **Framework**: React.js
- **State Management**: Redux/Context API
- **UI Components**: Material-UI
- **Routing**: React Router
- **HTTP Client**: Axios

Backend

- **Runtime**: Node.js
- **Framework**: Express.js
- **Authentication**: JWT (JSON Web Tokens)
- **Database**: MongoDB with Mongoose ODM

- **Validation**: Joi
- **Logging**: Winston

Client Library

- **Language**: JavaScript/Node.js
- **Cryptography**: crypto-js
- **HTTP Client**: Axios
- **Persistence**: File system for local storage

User Workflows

Admin Workflow

1. Authentication

Login Process

Figure 2: User Login Flow

Process Steps:

1. User accesses the login page
2. User enters credentials (email/password)
3. System validates credentials
4. On success: Redirect to dashboard
5. On failure: Show error message and allow retry

Key Features:

- Secure login with JWT authentication
- Role-based access control (RBAC)
- Session management
- Failed login attempt tracking

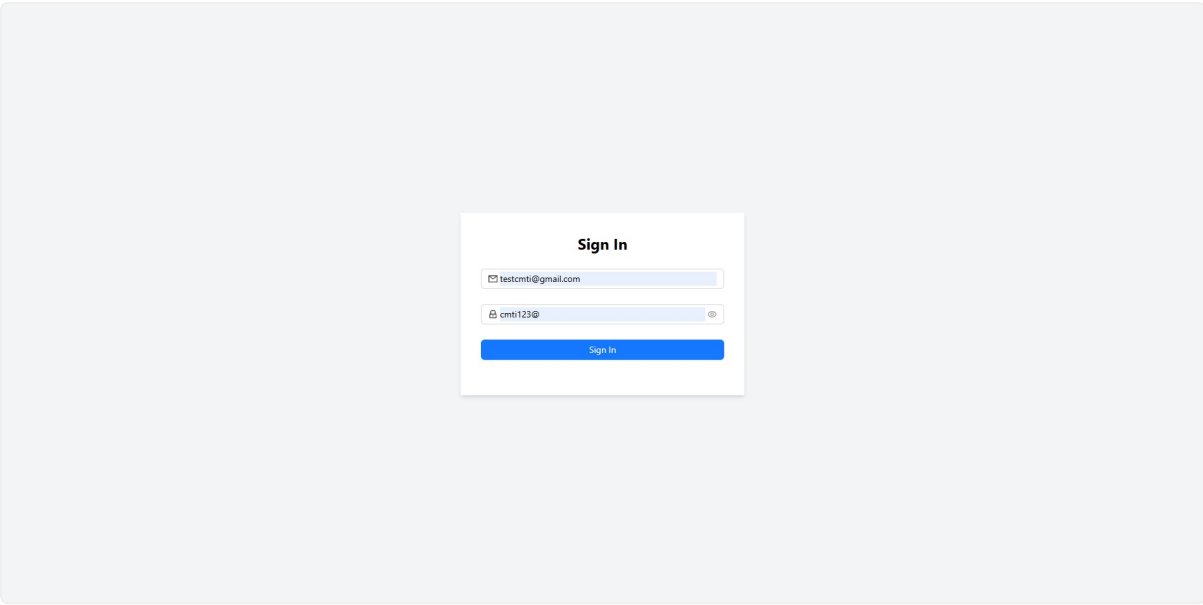

Figure 1: Admin Login Interface

2. Dashboard

Dashboard Components:

- System health metrics
- License usage statistics
- Recent activities
- Quick action buttons

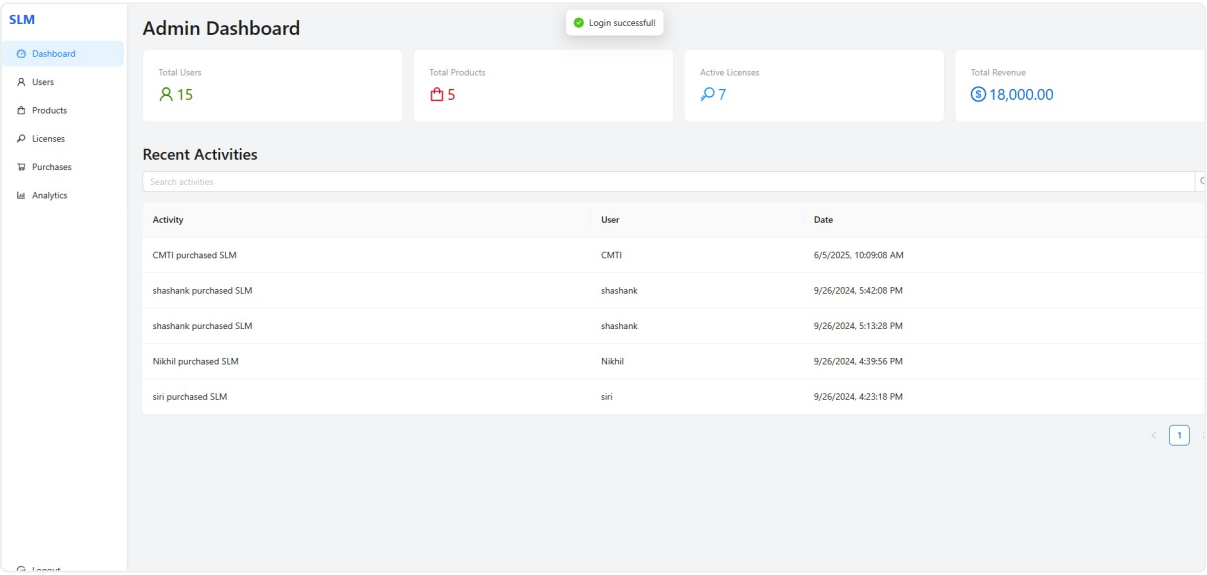

Figure 2: Admin Dashboard Overview

3. User Management

User Management Workflow

User Management Workflow

Figure 3: User Management Flow

Process Steps:

1. View list of users with search/filter options
2. Add new user or edit existing user
3. Set appropriate permissions and roles
4. Save changes to update user record
5. Export user data as needed

Features:

- User CRUD operations
- Role assignment
- Bulk operations
- Activity logging

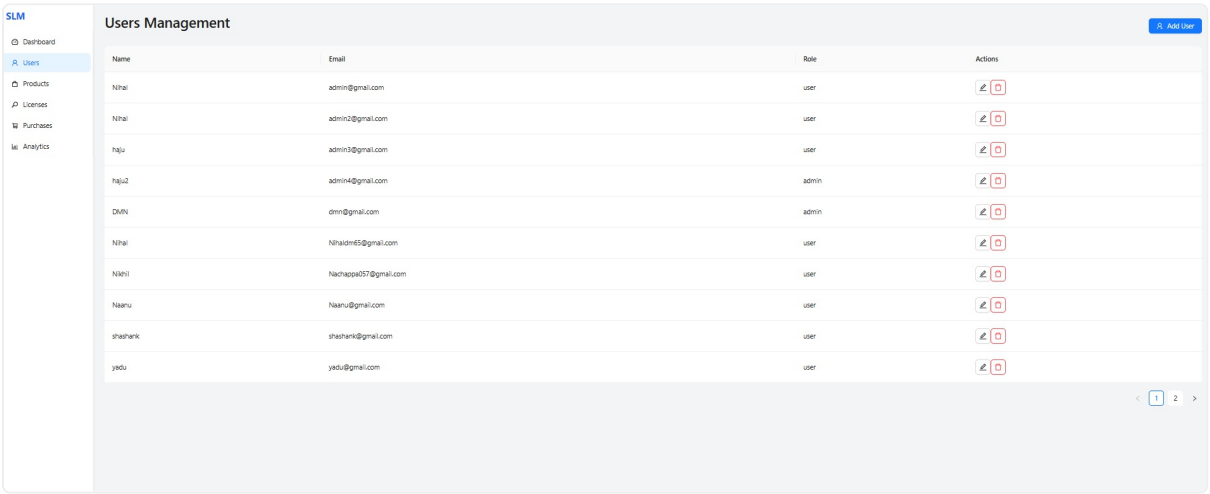

Figure 3: User Management Interface

4. Product Management

Product Lifecycle

1. **Create Product**
    - Basic information
    - Version control
    - Feature sets
    - Pricing models
2. **Update Product**
    - Version updates
    - Feature modifications
    - Deprecation handling

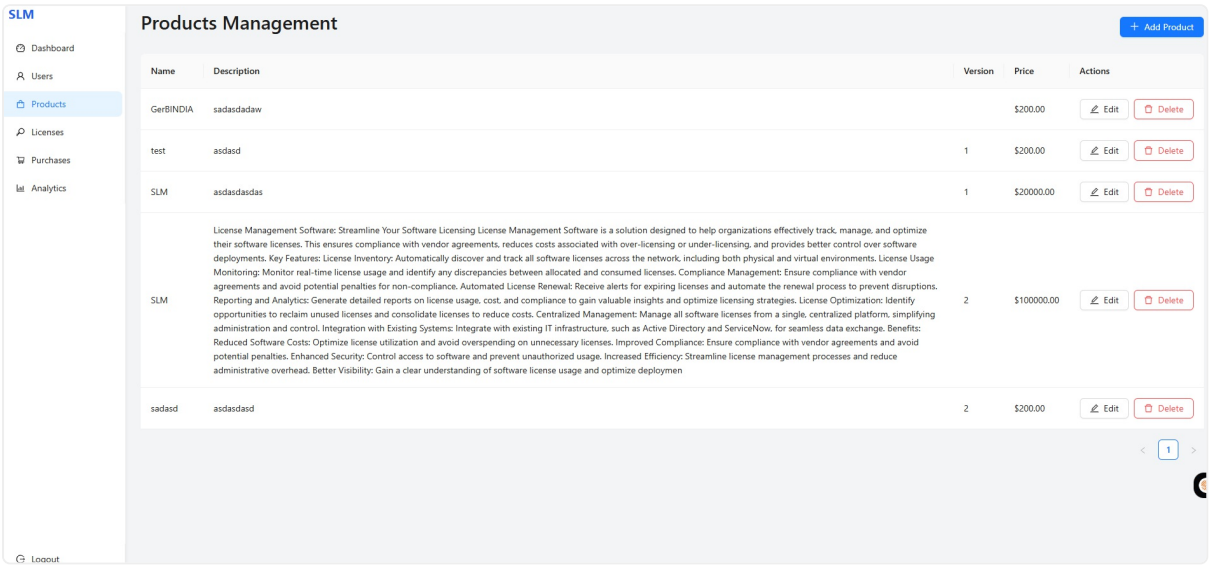

Figure 4: Product Catalog View

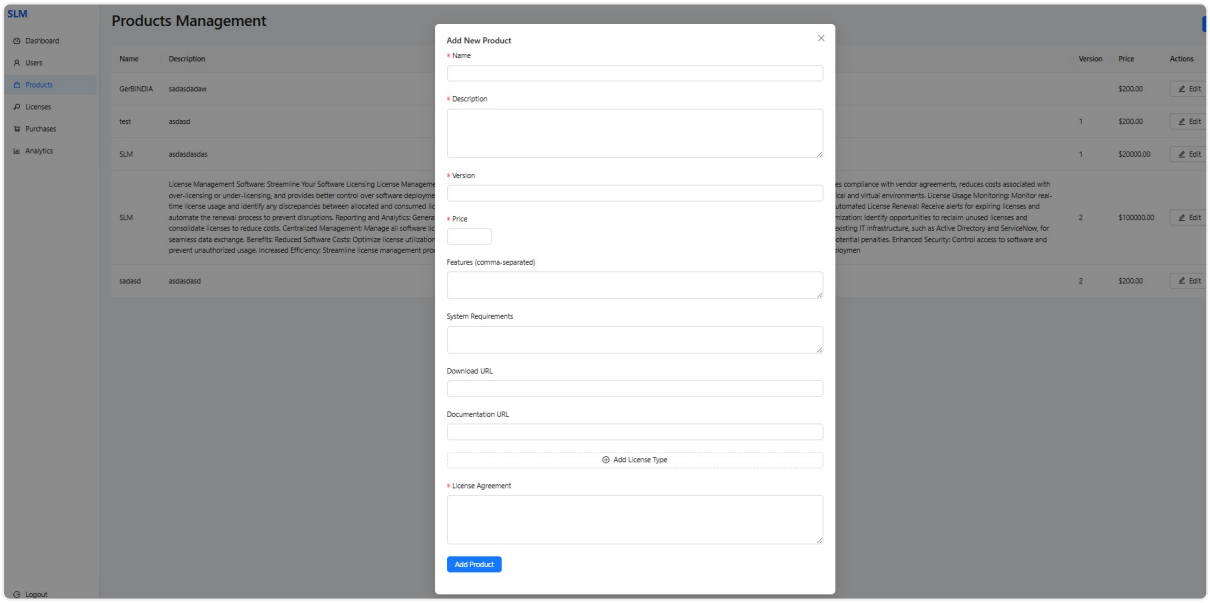

Figure 5: Adding a New Product

5. License Management

License Generation Process

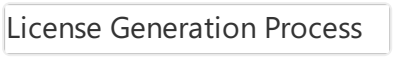

Figure 4: License Generation Flow

Process Steps:

1. Admin requests new license generation
2. Backend validates request and creates license record
3. Database stores the new license information
4. System returns the generated license key
5. Admin distributes key to end user
6. User activates license in their system

Key Features:

- Batch generation
- Custom validation rules
- Expiration settings
- Activation limits

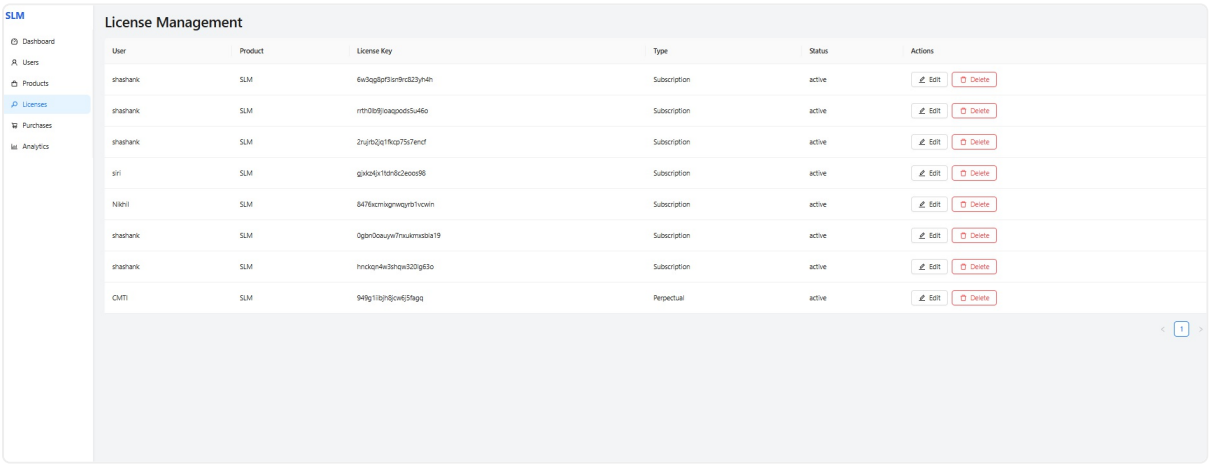

Figure 6: License Management Console

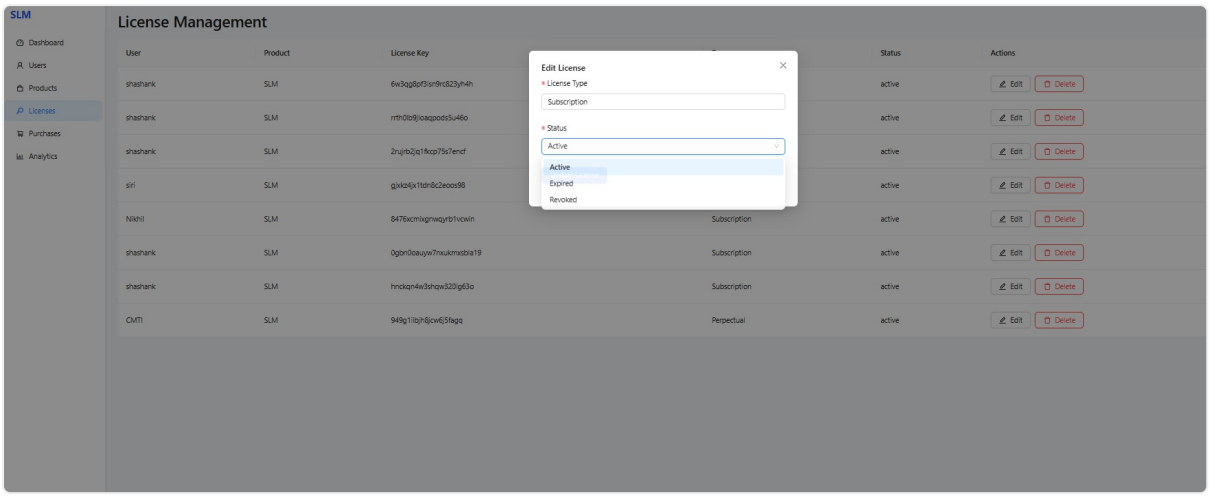

Figure 7: Editing License Details

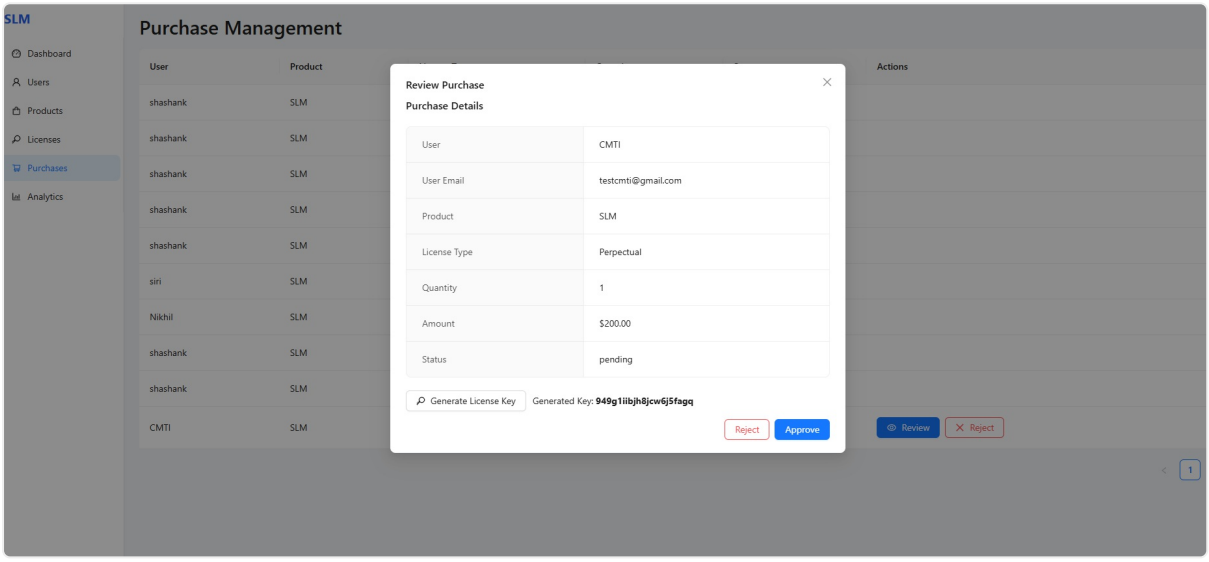

Figure 8: License Key Generation

6. Purchase Management

Purchase Workflow

1. Order Processing
2. Payment Verification
3. License Assignment
4. Notification

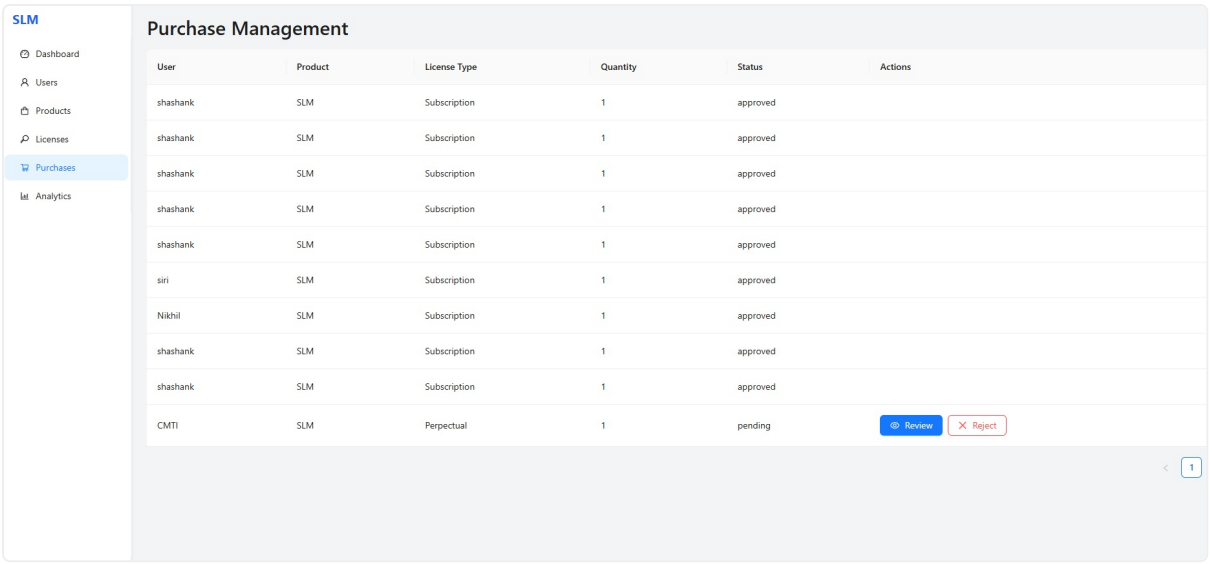

Figure 9: Purchase Management View

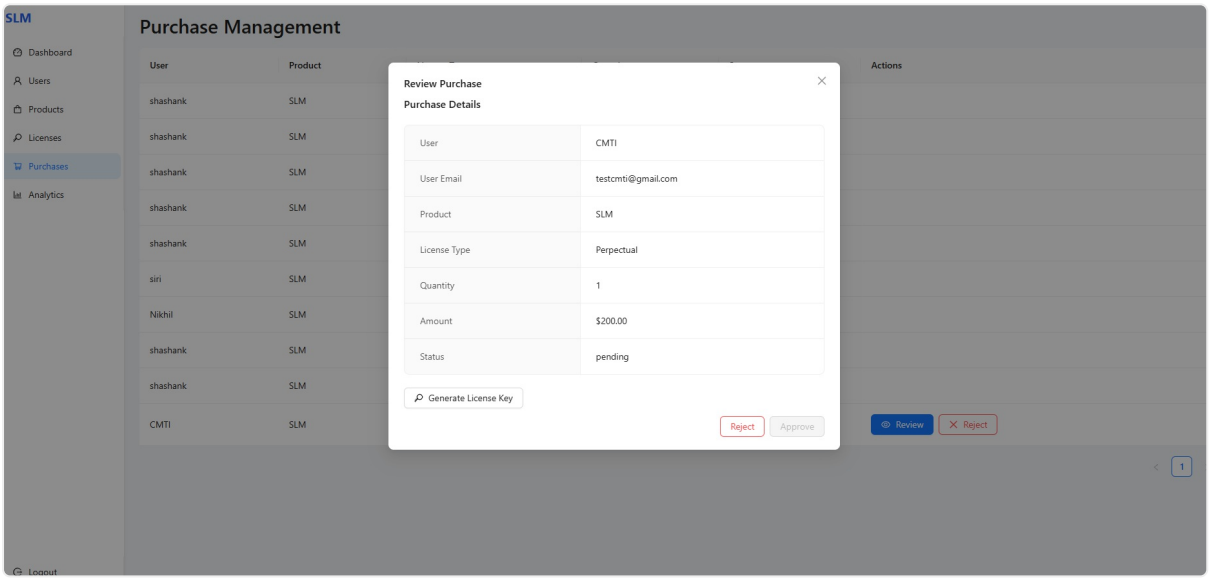

Figure 10: Purchase Review Process

End User Workflow

1. Account Management

Registration Process

Registration Flow

Figure 7: User Registration Process

Process Steps:

1. User starts registration with basic details
2. System sends verification email
3. User verifies email address
4. User sets up password
5. User completes profile information
6. Access granted to user dashboard

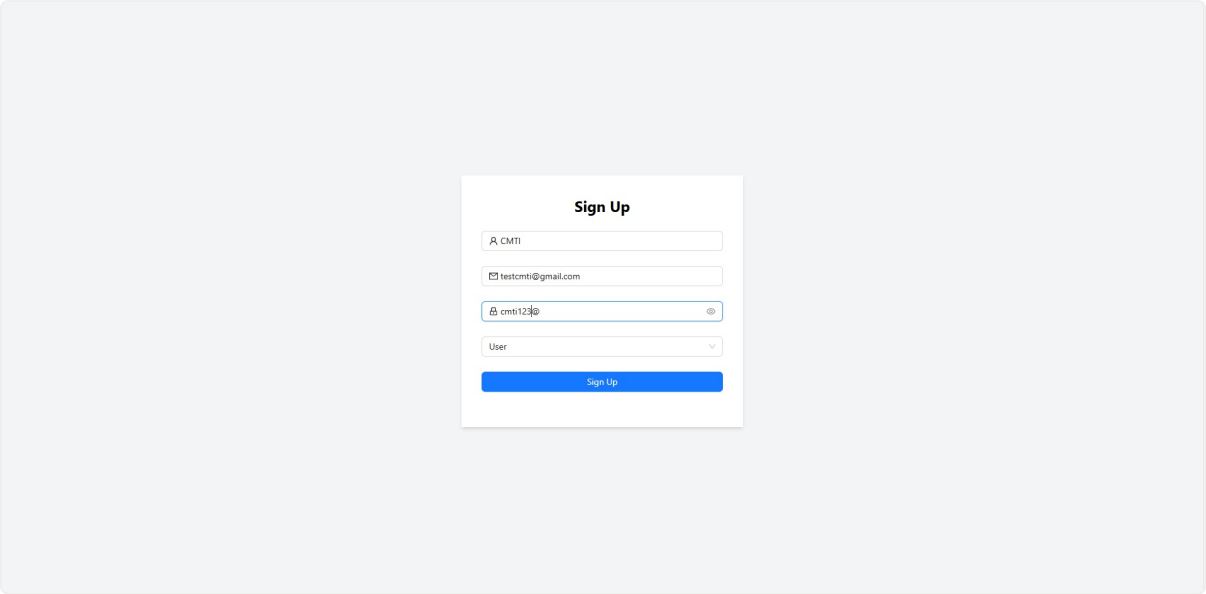

Figure 11: User Registration Form

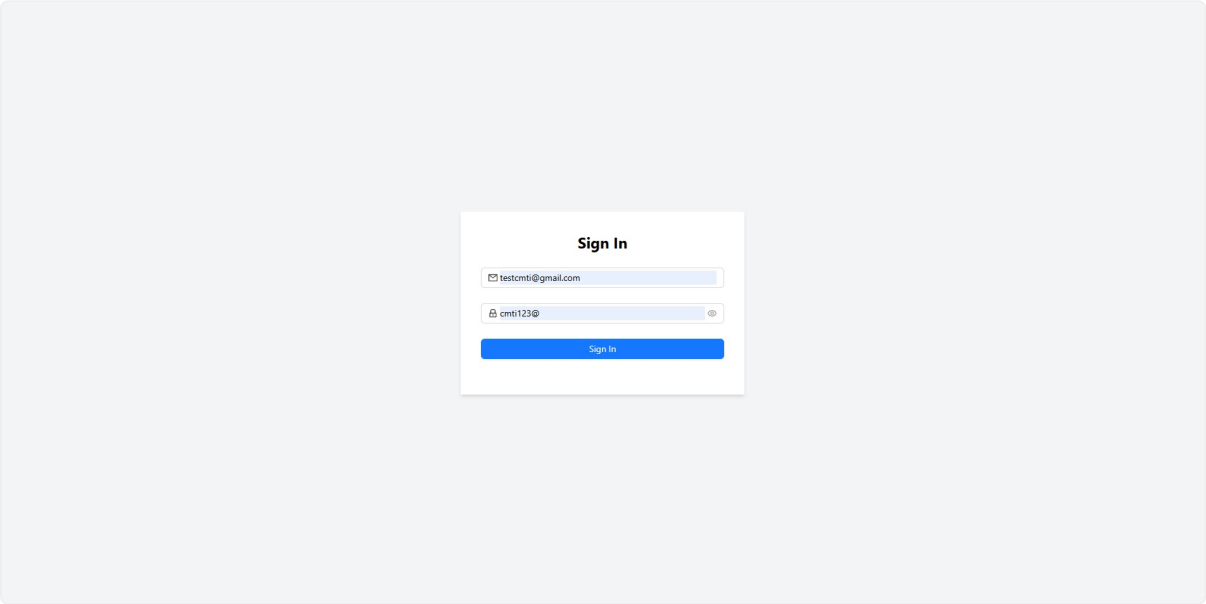

Figure 12: User Login Screen

2. User Dashboard

Dashboard Features:

- License overview
- Quick actions
- Notifications
- Recent activities

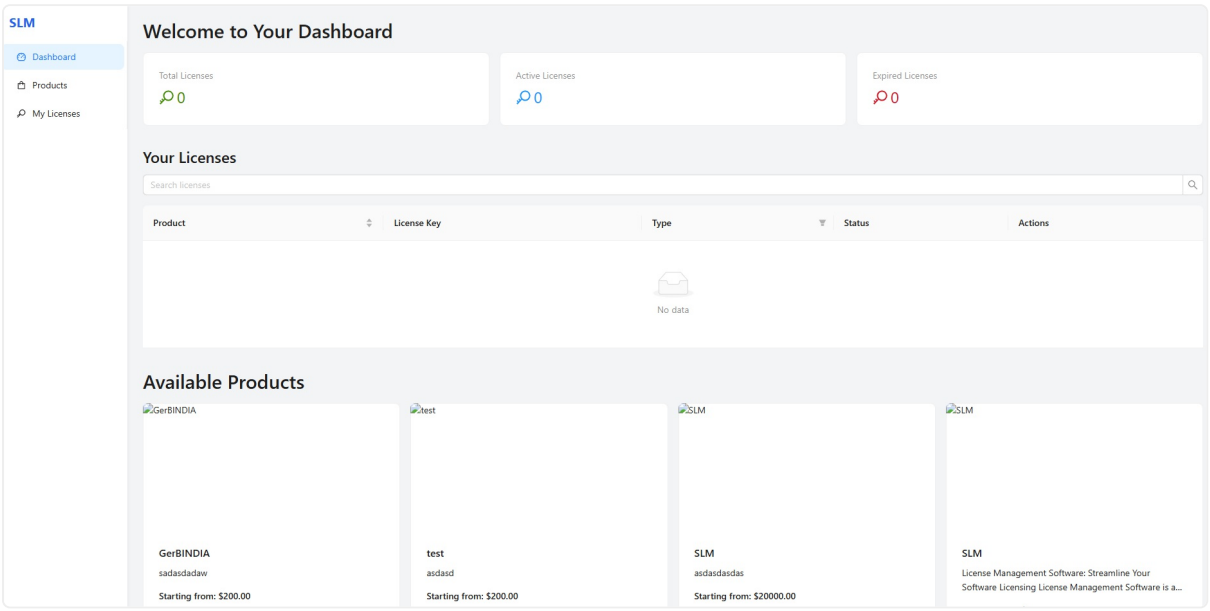

Figure 13: User Dashboard Overview

3. License Management

License Operations:

- View active licenses
- Check activation status
- Request additional licenses
- Download license files

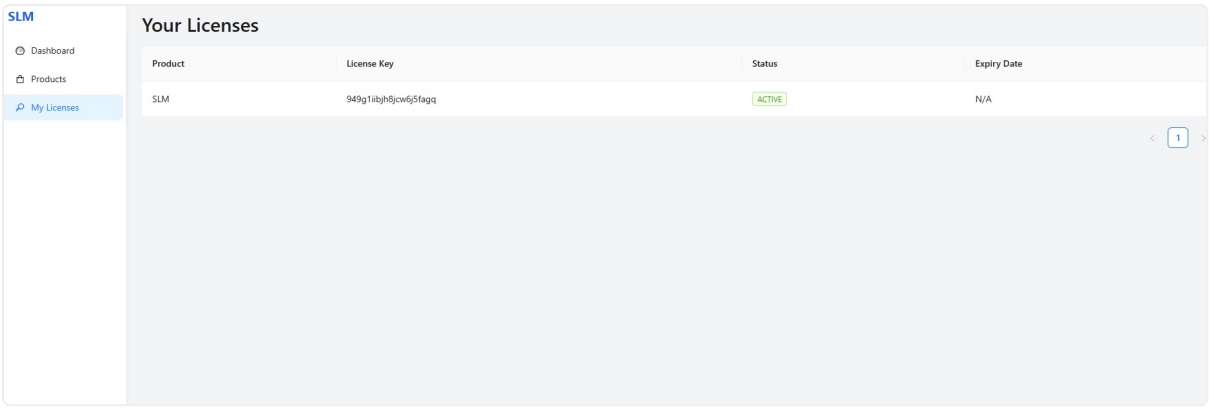

Figure 14: User License Management

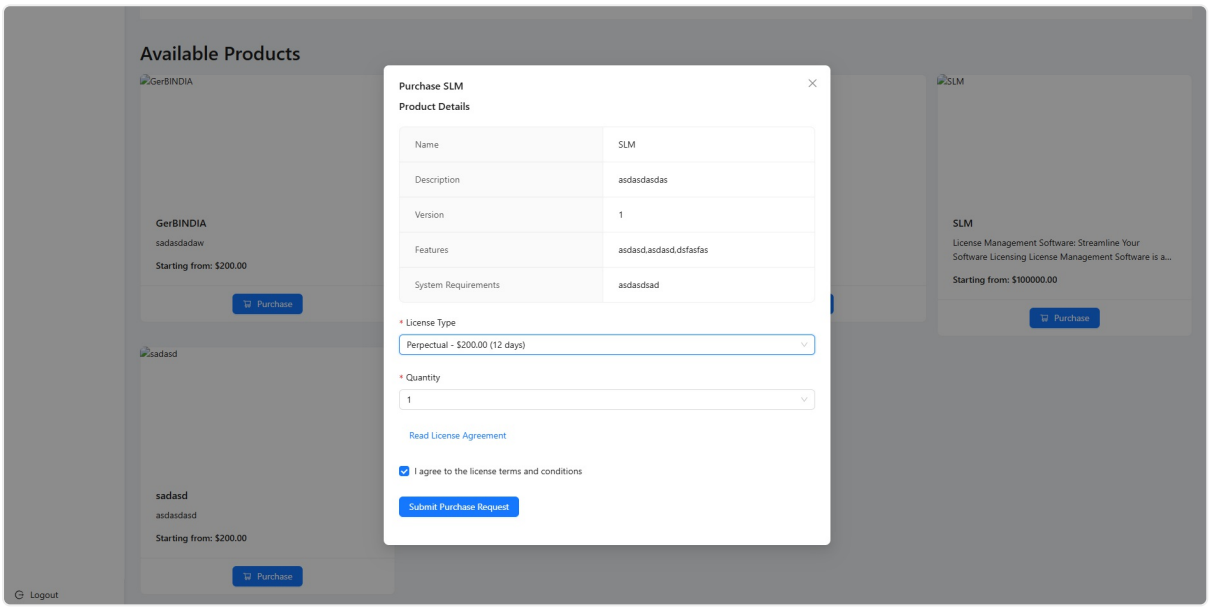

Figure 15: Purchase History

4. License Activation

Activation Flow

1. Enter License Key
2. System Validation
3. Machine Binding
4. Confirmation

Offline Activation

1. Generate Activation Request
2. Submit Request File

3. Receive Activation File
4. Complete Activation

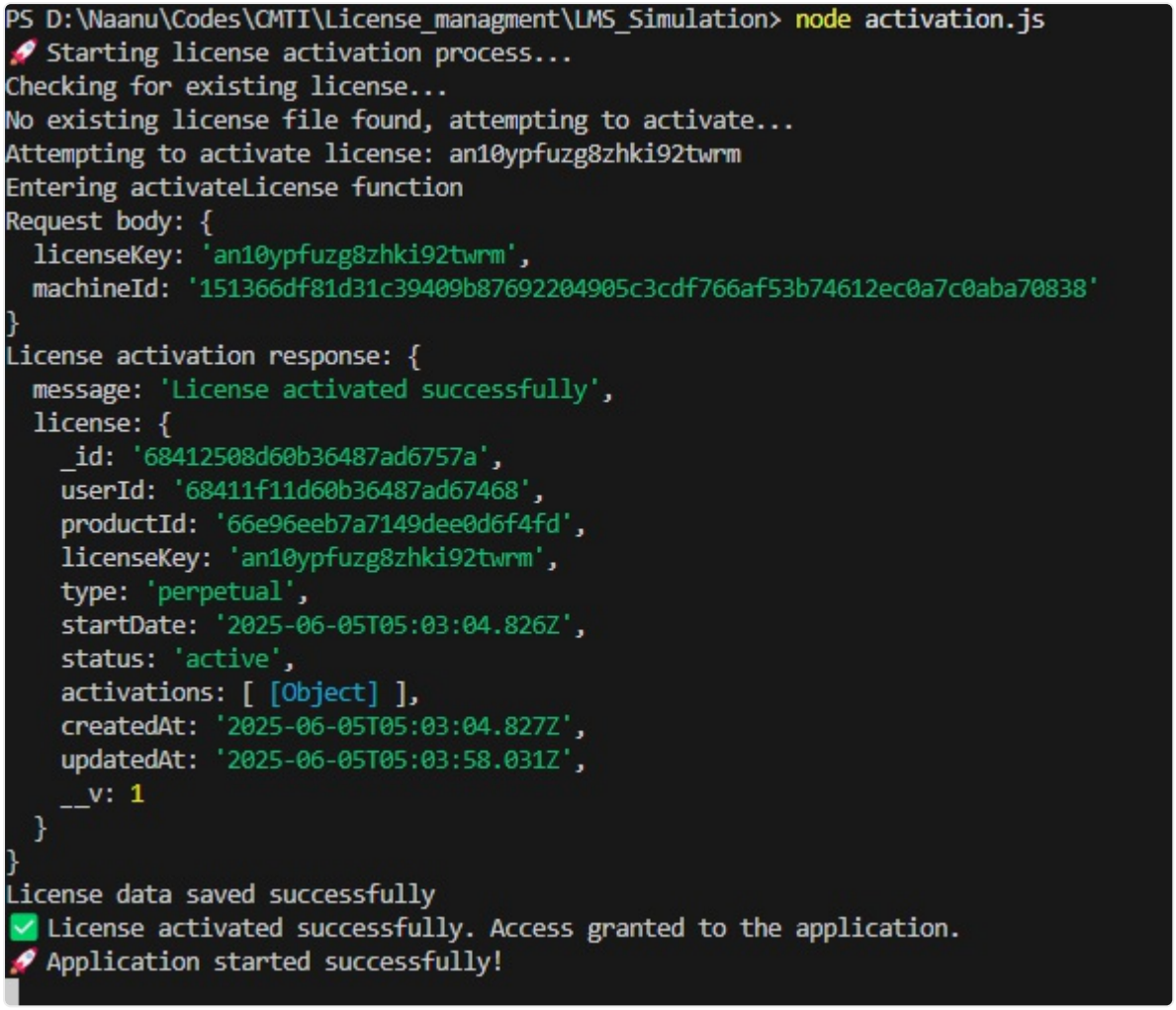

Figure 16: License Activation Process

Support Workflow

Ticket Lifecycle

Support Ticket Flow

Figure 8: Support Ticket Lifecycle

Process Flow:

1. User submits support ticket
2. System performs initial triage

3. Based on complexity:
    - Simple: Resolve directly
    - Complex: Escalate to appropriate team
4. Resolution is implemented
5. Ticket is closed after confirmation

Support Channels:

- In-app messaging
- Email support
- Knowledge base
- Community forums

Integration Points

API Reference

Base URL

```
https://api.yourdomain.com/v1
```

Authentication

All API requests require an API key or JWT token in the header:

```
Authorization: Bearer YOUR_API_KEY
```

Core Endpoints

1. Authentication

Login

```
POST /api/auth/login
```

Request:

```
{
  "email": "user@example.com",
  "password": "your_password"
}
```

Response:

```
{
  "token": "jwt_token_here",
  "user": {
    "id": "user_id",
    "email": "user@example.com",
    "role": "admin"
  }
}
```

2. License Management

Generate License

```
POST /api/licenses
```

Validate License

```
GET /api/licenses/validate/{key}
```

Get License Details

```
GET /api/licenses/{id}
```

Webhooks

Webhooks allow you to receive real-time updates about system events.

Available Events:

- `license.created`
- `license.activated`
- `license.expired`
- `purchase.completed`

- `user.registered`

Webhook Payload Example:

```
{
  "event": "license.activated",
  "data": {
    "licenseKey": "LIC-123-456-789",
    "productId": "prod_123",
    "activatedAt": "2025-06-05T11:30:00Z",
    "machineId": "machine_fingerprint"
  },
  "timestamp": "2025-06-05T11:30:01Z"
}
```

Client Integration

Installation

```
npm install @your-org/license-manager
# or
yarn add @your-org/license-manager
```

Basic Usage

```
const { LicenseManager } = require('@your-org/license-manager');

// Initialize with your API key
const license = new LicenseManager({
  apiKey: 'your_api_key',
  productId: 'your_product_id',
  storage: 'localStorage' // or 'file' for Node.js
});

// Activate license
async function activateLicense(key) {
  try {
    const result = await license.activate(key);
    console.log('Activation successful:', result);
  } catch (error) {
    console.error('Activation failed:', error.message);
  }
}

// Check license status
async function checkLicense() {
  const status = await license.validate();
  if (status.valid) {
    console.log('License is valid until:', status.expiresAt);
  } else {
    console.error('License is invalid:', status.reason);
  }
}
```

Troubleshooting Guide

Common Issues and Solutions

1. Activation Issues

Problem: License activation fails

Solution:

1. Check internet connection
2. Verify license key format
3. Ensure machine ID hasn't changed
4. Check server logs for errors

Error Reference:

```
Error: Invalid license key
  - Verify the key was entered correctly
  - Check if the key has been revoked
  - Contact support if issue persists
```

2. Validation Failures

Problem: License validation fails

Solution:

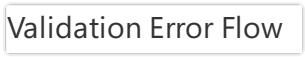

Figure 9: Validation Error Resolution Flow

Resolution Steps:

1. Identify the specific validation error
2. For expired licenses: Contact support for renewal
3. For invalid keys: Verify the key was entered correctly
4. For max activations: Deactivate old devices or request additional activations
5. Retry the activation after resolving the issue

3. API Connectivity

Problem: Cannot connect to license server

Troubleshooting Steps:

1. Check network connectivity
2. Verify API endpoint URL
3. Check CORS configuration
4. Test with Postman/cURL

Test Command:

```
curl -X GET "https://api.yourdomain.com/health" \
  -H "Authorization: Bearer YOUR_API_KEY"
```

4. Performance Issues

Problem: Slow response times

Optimization Tips:

- Enable caching where possible
- Use batch operations for multiple licenses
- Implement retry logic with exponential backoff

5. Security Concerns

Best Practices:

- Always use HTTPS
- Rotate API keys regularly
- Implement rate limiting
- Use secure storage for sensitive data

Getting Help

If you're still experiencing issues, please provide:

1. Error message
2. Steps to reproduce
3. Environment details
4. Relevant logs (with sensitive information redacted)

Contact support at: support@yourdomain.com

FAQs

Technical

Q: How do I integrate the license check in my application?
A: Include the client library and follow the integration guide. Detailed integration examples are available in the Integration Guide.

Q: What data is collected during activation?
A: Only the following data is collected:

- Machine ID (hardware fingerprint)
- License key
- Timestamp

- IP address (for security purposes)
- Product/version information

Conclusion

This document has provided a comprehensive overview of the License Management System, covering its architecture, features, and usage guidelines. The system is designed to be robust, secure, and scalable, meeting the needs of both administrators and end-users.

Key Takeaways

1. **For Administrators**
    - Centralized license management
    - Real-time monitoring and reporting
    - Granular access control
    - Automated workflows
2. **For Developers**
    - Well-documented RESTful API
    - Client libraries for easy integration
    - Webhook support for real-time updates
    - Comprehensive error handling
3. **For End Users**
    - Simple license activation
    - Self-service portal
    - Clear license status
    - Easy device management

Appendix

A. Glossary

Term	Definition
Activation	Process of binding a license to a specific device
API Key	Unique identifier for API authentication
JWT	JSON Web Token for secure authentication
Machine ID	Unique identifier generated from hardware components
Webhook	Method for real-time notifications

B. Error Codes

Code	Description	Resolution
400	Bad Request	Verify request parameters
401	Unauthorized	Check API key/token
403	Forbidden	Verify permissions
404	Not Found	Check endpoint URL
429	Too Many Requests	Implement rate limiting
500	Server Error	Contact support

C. Version History

Version	Date	Changes
1.0	2024-01-15	Initial Release
1.1	2024-03-22	Added webhook support
2.0	2025-06-05	Major UI overhaul, New APIs

D. Additional Resources

- API Reference Documentation (https://docs.yourdomain.com/api)

- Client SDK GitHub Repository (https://github.com/your-org/license-manager-sdk)
- Support Portal (https://support.yourdomain.com)
- Knowledge Base (https://help.yourdomain.com)

Document Version: 2.0
Last Updated: June 5, 2025
Prepared by: [Your Name/Team]
Confidential & Proprietary

Document Control

Version	Date	Author	Changes
1.0	2024-01-15	[Author]	Initial version
1.1	2024-03-22	[Author]	Added webhook section
2.0	2025-06-05	[Author]	Major update, new features

Reviewers

Name	Role	Approval Date
[Name]	[Role]	[Date]
[Name]	[Role]	[Date]

Distribution List

- [Stakeholder 1]
- [Stakeholder 2]
- [Stakeholder 3]

Diagram Maintenance Guide

How to Update Diagrams

1. **Using Draw.io (Recommended)**
    - Open draw.io (https://app.diagrams.net/)
    - Create a new diagram or import an existing one
    - Edit as needed
    - Export as PNG with the following settings:
        - Transparent background
        - Border: 10px
        - Zoom: 100%
    - Save to the `Documentation/Screenshots/` folder with the appropriate filename
2. **Using Mermaid (Alternative)**
    - Open Mermaid Live Editor (https://mermaid.live/)
    - Write or edit your diagram code
    - Click "Download PNG"
    - Save to the `Documentation/Screenshots/` folder

Diagram List

Figure	Filename	Description	Last Updated
1	system-architecture.png	System Architecture	YYYY-MM-DD
2	login-flow.png	User Login Flow	YYYY-MM-DD
3	user-management-flow.png	User Management	YYYY-MM-DD
4	license-generation-flow.png	License Generation	YYYY-MM-DD
5	license-activation-flow.png	License Activation	YYYY-MM-DD
6	license-validation-flow.png	License Validation	YYYY-MM-DD
7	registration-flow.png	User Registration	YYYY-MM-DD
8	support-ticket-flow.png	Support Ticket	YYYY-MM-DD
9	validation-error-flow.png	Error Resolution	YYYY-MM-DD

Best Practices

1. **Naming Conventions**
   - Use kebab-case for filenames
   - Be descriptive but concise
   - Include version numbers if needed (e.g., `system-architecture-v2.png`)
2. **Image Quality**
   - Minimum resolution: 1200x800px
   - Use PNG format for diagrams
   - Optimize file size without losing quality
3. **Version Control**
   - Add all diagram source files to version control
   - Include a README in the Screenshots folder
   - Document any external tools/versions used

This document and the information contained herein is the confidential and proprietary property of [Your Company Name]. Unauthorized copying or distribution is strictly prohibited.