

**SOFTWARE REQUIREMENT SPECIFICATION**

**License Marketplace System**

– Hanoi, Jan 2024 –

**Table of Contents**

[I. Record of Changes 3](#_dn2gwrbhr7n3)

[II. Software Requirement Specification 4](#_vbd22grmccnj)

[1. Overall Requirements 4](#_q7havltpywnx)

[1.1 Context Diagram 4](#_w7dtfaqt80ds)

[1.2 Main Business Processes 5](#_p1bdl9w2kxm5)

[1.3 User Requirements 5](#_jcyky67yn9b8)

[1.4 System Functionalities 7](#_nz5zf4i6kb6f)

[1.5 Entity Relationship Diagram 7](#_i931jlmxycq)

[2. Use Case Specifications 8](#_1y8fyihinmee)

[2.1 <<Feature Name1>> 8](#_uz8s0iu40n43)

[2.2 Xyz Feature 10](#_xn06jmhqxh2y)

[3. Functional Requirements 11](#_tifntvf5s2mw)

[3.1 Feature Name1 11](#_oduebmq3xunt)

[3.2 User Authentication 11](#_toi4xaxct1dy)

[3.3 System Administration 12](#_bkmr78vku96h)

[4. Non-Functional Requirements 13](#_soe00xcdlxpg)

[3.1 External Interfaces 13](#_pzb43pv2bh02)

[3.2 Quality Attributes 13](#_c6d8eqt0l5yr)

[5. Requirement Appendix 14](#_b3c89zi1d5ds)

[5.1 Business Rules 14](#_yho193gwq5ab)

[5.2 System Messages 14](#_2gr8zrl3rpod)

[5.3 Other Requirements… 15](#_3163hous5ore)

# I. Record of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **A\* M, D** | **In charge** | **Change Description** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

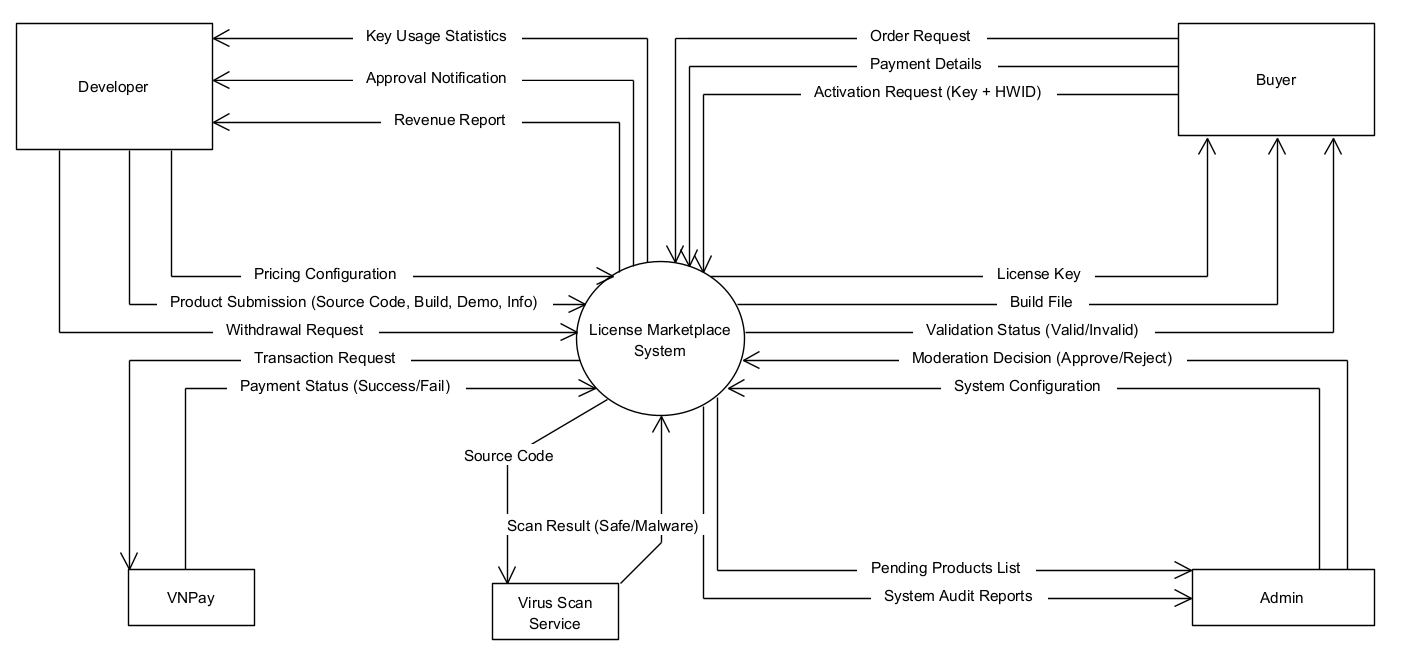
\*A - Added M - Modified D - Deleted

# II. Software Requirement Specification

## 1. Overall Requirements

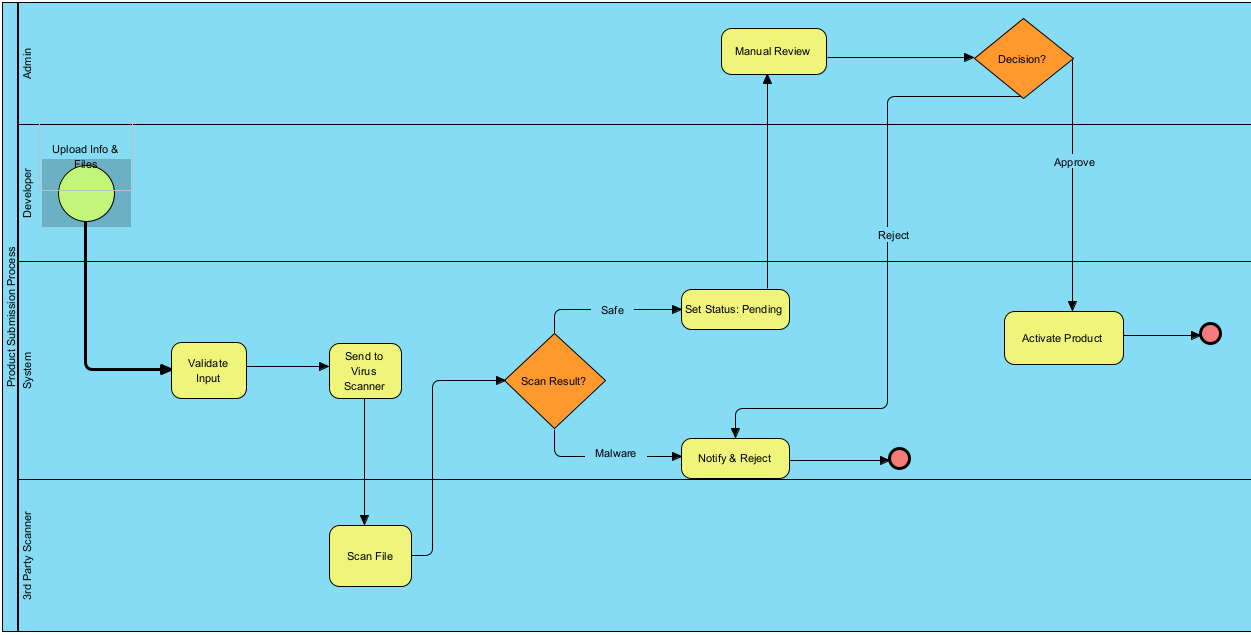
### 1.1 Context Diagram

**License Marketplace System** is a new web-based platform that streamlines the distribution, sale, and copyright protection of software products, replacing the fragmented and insecure manual processes currently used by independent developers. The context diagram below illustrates the external entities and system interfaces for **Release 1.0**, focusing on secure code submission, automated virus scanning via **3rd Party Security Services**, and real-time license key validation. The system is expected to evolve over several releases, ultimately connecting to recurring billing services for subscription-based software and providing advanced analytics APIs for developers to track user engagement.

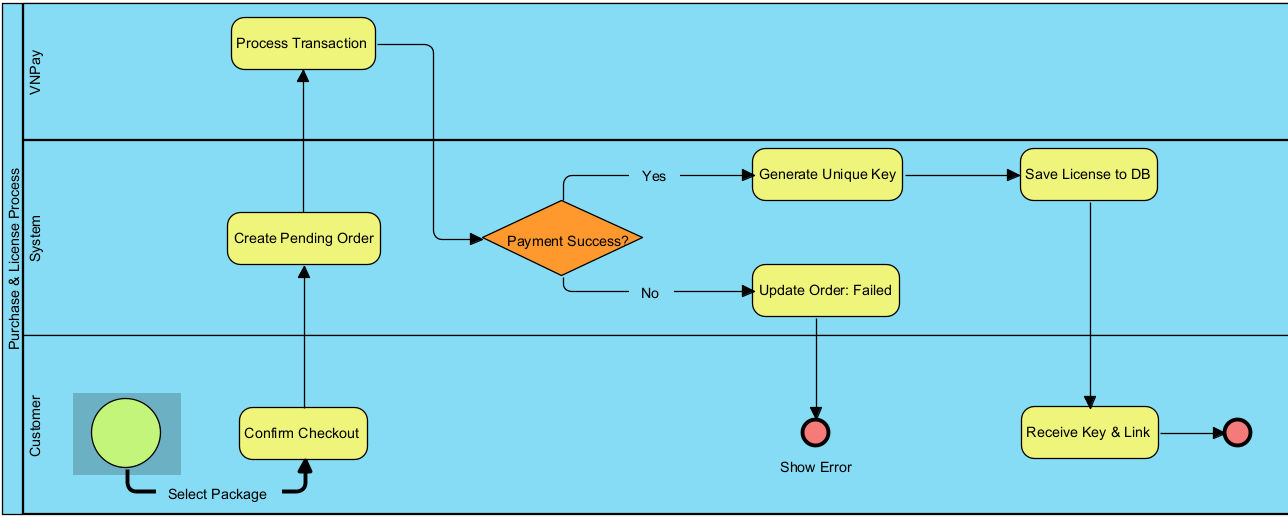


### 1.2 Main Business Processes

#### 1.2.1 Product Submission Processing



#### 1.2.2 Purchase & License Generation Process



### 1.3 User Requirements

#### 1.3.1 Actors

|  |  |  |
| --- | --- | --- |
| **#** | **Actor** | **Description** |
| 1 | Buyer (End User) | An individual who accesses the system to browse, purchase, and use software licenses. |
| 2 | Developer | An individual or entity who creates software and uses the platform to distribute their products. |
| 3 | Admin | A super-user responsible for managing the platform's operation, content moderation, and system configuration. |
| 4 | Payment Gateway | An external system that processes online financial transactions: VNPay |
| 5 | 3rd Party Security Service | An external system used for automated malware and virus scanning: |
| 6 | Client Software | The software installed on the Buyer's machine that interacts with the system for license validation. |

#### 1.3.2 Use Cases (UC)

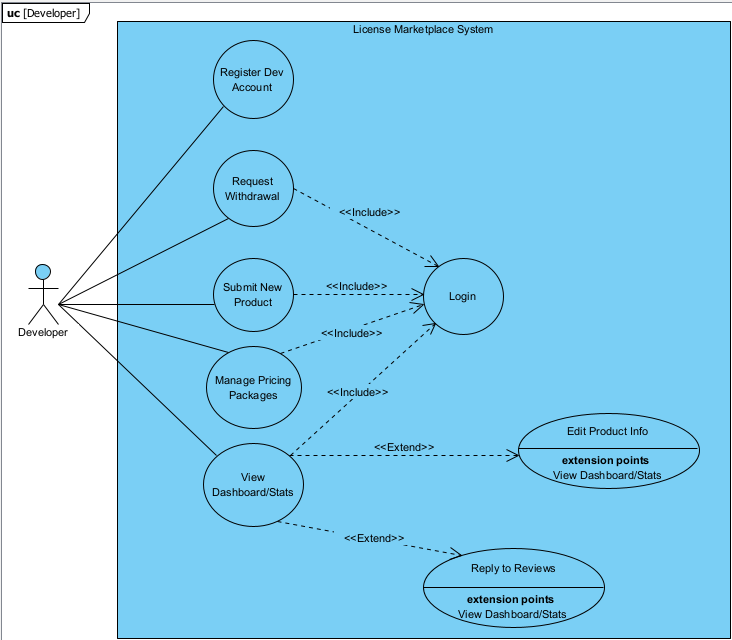
|  |  |  |  |
| --- | --- | --- | --- |
| **UC ID** | **Use Case Name** | **Primary Actor** | **Description (Outcome of Value)** |
| **UC-01** | Register Account | Buyer, Developer | The actor creates a new account on the system to access restricted features (buying or selling). |
| **UC-02** | Log In | All Actors | The actor authenticates their identity to access their specific dashboard and permissions. |
| **UC-03** | Search Products | Buyer (Guest) | The actor searches for software by name, category, or filter to find a product of interest. |
| **UC-04** | View Product Details | Buyer (Guest) | The actor views detailed information (video, description, pricing) of a specific product to make a purchase decision. |
| **UC-05** | Purchase License | Buyer | The actor selects a license package (e.g., monthly, lifetime) and completes the payment process to obtain the right to use the software. |
| **UC-06** | Download Product | Buyer | The actor retrieves the installation file (build) of the purchased software from the system. |
| **UC-07** | Submit New Product | Developer | The actor uploads product details, source code, and build files for administrator review. |
| **UC-08** | Manage Pricing Packages | Developer | The actor defines or updates the cost and duration (e.g., 1 month, 1 year) for their software licenses. |
| **UC-09** | Request Withdrawal | Developer | The actor requests a payout of their accumulated earnings from the system wallet to their bank account. |
| **UC-10** | Approve Product | Administrator | The actor reviews a submitted product (and its virus scan results) and publishes it to the marketplace. |
| **UC-11** | Reject Product | Administrator | The actor declines a submitted product due to policy violations or malware detection, providing a reason to the Developer. |
| **UC-12** | Process Withdrawal | Administrator | The actor reviews a financial withdrawal request and transfers funds to the Developer. |
| **UC-13** | Ban User | Administrator | The actor locks a specific user account to prevent access due to suspicious activity or policy violations. |
| **UC-14** | Activate License | Client App (User) | The Client Application (acting on behalf of the user) validates the input key and hardware ID with the server to unlock the software. |

#### 1.3.2 Use Case Diagrams

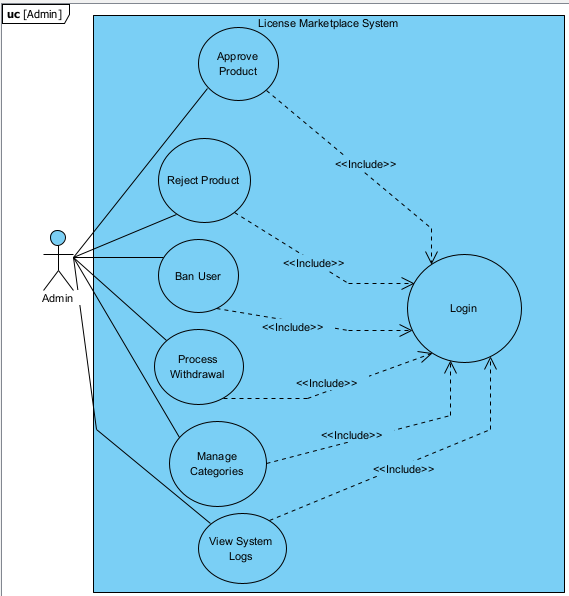
##### 1.3.2.1 UCs for Buyer

**

##### 1.3.2.2 UCs for Developer

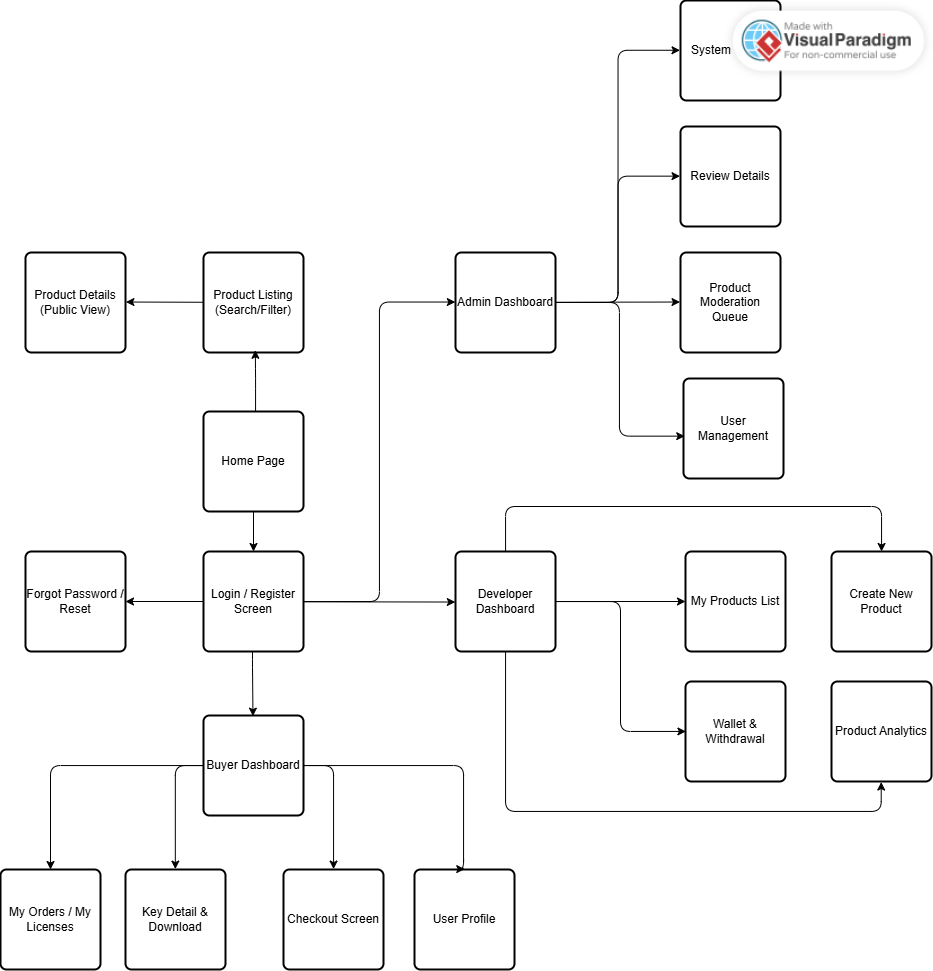
**

##### 1.3.2.3 UCs for Admin



### 1.4 System Functionalities

#### 1.4.1 Screens Flow



#### 1.4.2 Screen Authorization

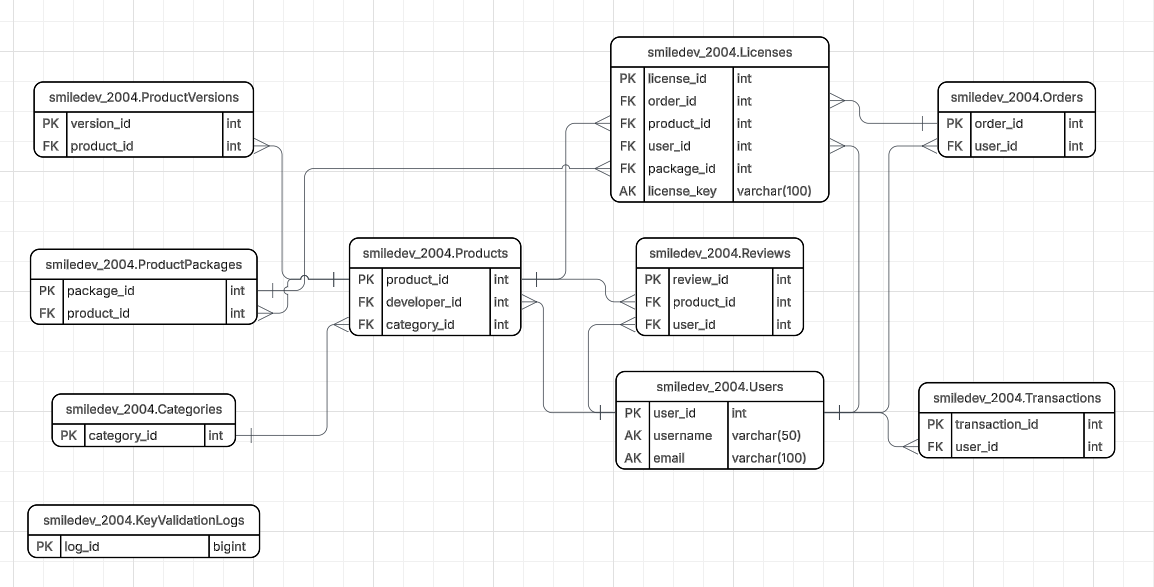
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Screen / Feature** | **Guest** | **Buyer** | **Developer** | **Administrator** |
| **Public Screens** | | | | |
| Home Page | X | X | X | X |
| Login / Register | X |  |  |  |
| Product Listing (Search/Filter) | X | X | X | X |
| Product Details (Public View) | X | X | X | X |
| Forgot Password | X |  |  |  |
| **Buyer Screens** | | | | |
| Buyer Dashboard |  | X |  |  |
| My Licenses / My Orders |  | X |  |  |
| Key Detail & Download |  | X |  |  |
| Checkout / Payment |  | X |  |  |
| User Profile (Buyer View) |  | X |  |  |
| **Developer Screens** |  |  |  |  |
| Developer Dashboard |  |  | X |  |
| Create New Product |  |  | X |  |
| Manage Products List |  |  | X |  |
| Wallet & Withdrawal |  |  | X |  |
| Product Analytics |  |  | X |  |
| **Admin Screens** | | | | |
| Admin Dashboard |  |  |  | X |
| Product Moderation Queue |  |  |  | X |
| Review Product Details |  |  |  | X |
| User Management (Ban/Unban) |  |  |  | X |
| System Configuration |  |  |  | X |
| Transaction History (All) |  |  |  | X |

#### 1.4.3 Non-UI Functions

*[Provide the descriptions for the non-screen system functions, i.e batch/cron job, service, API, etc.]*

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Feature** | **System Function** | **Description** |
| 1 | <<Feature Name>> | <<Function Name1>> | <<Function Name1 Description>> |
| 2 | … |  |  |

### 1.5 Entity Relationship Diagram



**Entities Description**

|  |  |  |
| --- | --- | --- |
| **#** | **Entity** | **Description** |
| 1 | Users | Stores account information for all system roles (Admin, Developer, Buyer). It manages authentication credentials, profile details, and the digital wallet balance. |
| 2 | Products | Contains core information about the software listed by Developers (e.g., name, description, demo video). It serves as the central entity linking Developers to Buyers. |
| 3 | ProductVersions | Manages the versioning history of software (e.g., v1.0, v1.1). It stores file paths for both Source Code (for Admin security checks) and Build Files (for User downloads). |
| 4 | ProductPackages | Defines the pricing models and duration limits for each product (e.g., "1-Month Subscription", "Lifetime License"). |
| 5 | Categories | Manages software classifications (e.g., System Tools, Game Assets, UI Kits) to help users categorize and filter products effectively. |
| 6 | Orders | Records purchase transactions made by Buyers. It tracks the total amount, payment method, and the current payment status (Pending/Completed). |
| 7 | Licenses | (Core Entity) Stores the unique license keys generated upon successful orders. It manages critical DRM data including expiration dates, activation status, and the bound Hardware ID. |
| 8 | Transactions | Logs the history of all financial movements within the system, including deposits, purchase deductions, developer revenue credits, and withdrawals. |
| 9 | Reviews | Stores feedback and ratings (1-5 stars) provided by Buyers for purchased products, helping to establish product quality and trust. |
| 10 | ActivityLogs | An audit trail that records critical actions performed by Administrators and Developers (e.g., Approving a product, Changing configuration) for security and accountability. |
| 11 | KeyValidationLogs | Logs the history of API requests from client software when validating licenses. Used for monitoring usage frequency and detecting potential piracy attempts. |

## 2. Use Case Specifications

### 2.1 Feature: Product Management

#### 2.1.2 UC-07: Submit New Product

| Primary Actors | Developer |
| --- | --- |
| Secondary Actors | 3rd Party Security Service (Virus Scanner) |
| Description | As a Developer, I want to upload my software (source code, build files, and details) so that it can be reviewed by the Admin and listed for sale on the marketplace. |
| Preconditions | 1. The Developer must be logged in to the system.2. The Developer account must be active (not banned). |
| Postconditions | 1. A new product record is created in the database with status "Pending".2. Source code and build files are stored securely on the server.3. An asynchronous request is sent to the 3rd Party Security Service to scan the uploaded files. |
| Normal Sequence/Flow | User Actions:Developer accesses the "Create New Product" screenDeveloper enters product metadata (Name, Description, Category)Developer uploads the required files: Source Code (.zip) and Build File (.exe/.msi).Developer configures the pricing packages (e.g., Monthly, Lifetime).Developer clicks the "Submit" button.System Responses:System validates the input data (file size, file format, required fields).System uploads files to the storage server.System records the product information in the database.System triggers an API call to the 3rd Party Security Service to initiate a malware scanSystem displays a success message: "Product submitted successfully and is pending review." |
| Alternative Sequences/Flows | A1. Invalid File Format:At step 6, if the uploaded file is not in the allowed format (e.g., .bat, .sh), the System displays an error message "Invalid file format" and terminates the flow. The Developer remains on the screen to correct the file.A2. Missing Pricing Config:At step 6, if no price package is defined, the System prompts the Developer to add at least one pricing option. |

### 2.2 Feature: Purchase & Licensing

#### 2.1.2 UC-07: Submit New Product

|  |  |
| --- | --- |
| **Primary Actors** | **Buyer** |
| **Secondary Actors** | Payment Gateway (e.g., VNPay/PayPal) |
| **Description** | As a Buyer, I want to purchase a license package for a specific software so that I can download and use it legally. |
| **Preconditions** | 1. The Buyer is logged in.  2. The selected Product is in "Approved" status.  3. The Product has valid pricing packages configured. |
| **Postconditions** | 1. A new Order is created with status "Completed".  2. A unique License Key is generated and assigned to the Buyer.  3. The Developer's wallet balance is credited (revenue).  4. The Buyer receives the download link. |
| **Normal Sequence/Flow** | User Actions:  1. Buyer views the Product Detail page and clicks "Buy Now" on a specific package.  2. System displays the Order Summary (Price, Tax, Total).  3. Buyer confirms and selects a payment method.  4. System redirects Buyer to the Payment Gateway interface.  5. Buyer enters payment credentials and confirms transaction.System Responses:  6. Payment Gateway processes the transaction and returns a "Success" signal to the System.  7. System updates the Order status to "Completed".  8. System calls the License Key Generator service to create a unique key.9. System saves the License Key to the database (linked to the Order).  10. System redirects Buyer to the "My Licenses" screen displaying the new Key and Download button. |
| **Alternative Sequences/Flows** | A1. Payment Failed:At step 6, if the Payment Gateway returns "Failed" or "Insufficient Funds":- System updates Order status to "Failed".- System displays an error message "Payment failed, please try again."- System redirects Buyer back to the Checkout screen.  A2. Transaction Timeout:If the user takes too long at step 5, the session expires. The System cancels the pending order and redirects the user to the Home page. |

### 2. 3 Feature: DRM System (Digital Rights Management)

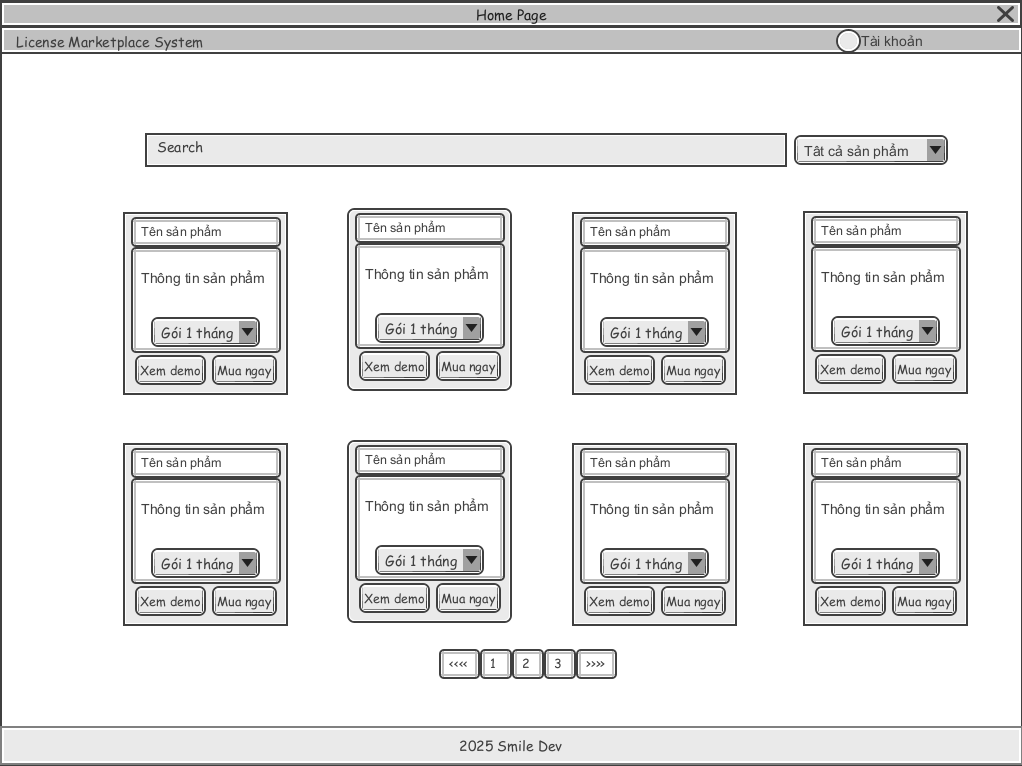
#### 2. 3.1 UC-14: Activate License (Check Key)

|  |  |
| --- | --- |
| **Primary Actors** | **Client Application (Acting on behalf of User)** |
| **Secondary Actors** | None |
| **Description** | When the user opens the software, the Client App communicates with the System API to validate the license key and bind it to the hardware, ensuring the key is valid and not pirated. |
| **Preconditions** | 1. The User has installed the software.  2. The User has a valid License Key.  3. The User's device has an internet connection. |
| **Postconditions** | 1. If valid: The software unlocks features. The Hardware ID is bound to the Key in the database (if first time).  2. If invalid: The software denies access. |
| **Normal Sequence/Flow** | User/App Actions:  1. User launches the software and enters the License Key.  2. Client App retrieves the device's Hardware ID (HWID).  3. Client App sends an API request (Key + HWID) to the System.System Responses:  4. System checks if the Key exists in the database.  5. System checks if the Key is expired (Current Date > Expiration Date).  6. System checks the HWID binding:    a. If HWID in DB is NULL: System updates the DB with the current HWID.    b. If HWID in DB matches current HWID: Proceed.  7. System returns a "Success/Valid" token to the Client App.  8. Client App unlocks the interface for the user. |
| **Alternative Sequences/Flows** | A1. Key Not Found / Expired:At step 4 or 5, if the check fails, System returns "Invalid/Expired" status. Client App displays "License is invalid or expired" and locks the software.  A2. Hardware Mismatch (Piracy Detected):At step 6, if the HWID in DB is different from the current HWID (meaning key is already used on another device), System returns "Hardware Mismatch". Client App displays "This key is already used on another device." |

## 3. Functional Requirements

### 3.1 Feature: Public Marketplace (Guest & User)

### 3.1.1 Screen: Home Page & Product Listing

**

This screen allows the Guest/Buyer to:

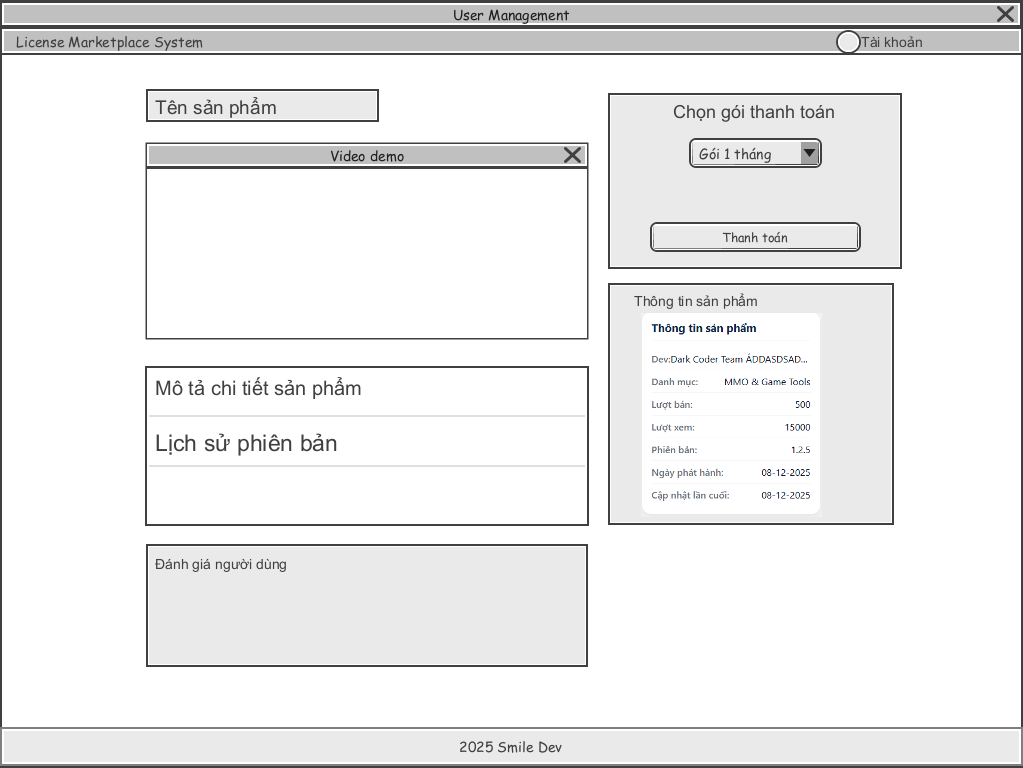
* **View Product List:** view list of current available software products on the marketplace.
* **Filter Product List:** filter products by categories (e.g., System Tools, Games) or price ranges.
* **Search Products:** enter keyword(s) to search products by their names or descriptions.
* **Sort Product List:** sort the product list (ascending, descending) by selecting options (e.g., Price: Low to High, Top Rated, Newest).

On the screen, s/he can also:

* **View Product Summary:** see quick information like price, rating, and thumbnail before clicking.
* **Choose to go to the Product Details** screens for viewing full information, video demos, and purchasing options by clicking the **Product Name** or **Thumbnail** link.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| **Header Section** | |
| **Search Bar** | Input text. Max length: 100 chars. Users enter keywords to search for products. |
| **Login / Register Link** | Hyperlink/Button. Redirects to the Login Screen. Visible only to Guest users. |
| **Dashboard Link** | Hyperlink. Redirects to the user's Dashboard. Visible only to Logged-in users. |
| **Filter Sidebar** | |
| **Category Filter** | Checkbox List. Displays all active categories from the database. Users can select multiple. |
| **Price Range Filter** | Two Input Number fields (Min - Max). Allows users to filter products within a specific budget. |
| **Product Grid** | |
| **Sort Dropdown** | Dropdown list. Options: "Newest", "Price: Low to High", "Price: High to Low", "Best Selling". |
| **Product Thumbnail** | Image. Displays the cover image of the software. Clicking it navigates to Product Details. |
| **Product Name** | Label/Link. Displays the name of the software. Clicking it navigates to Product Details. |
| **Price Label** | Label. Displays the price (e.g., "100.000 đ") or "Free". |
| **Rating Stars** | Icon set (1-5 stars). Displays average user rating. |

### 3.1.2 Screen: Product Details & Purchase



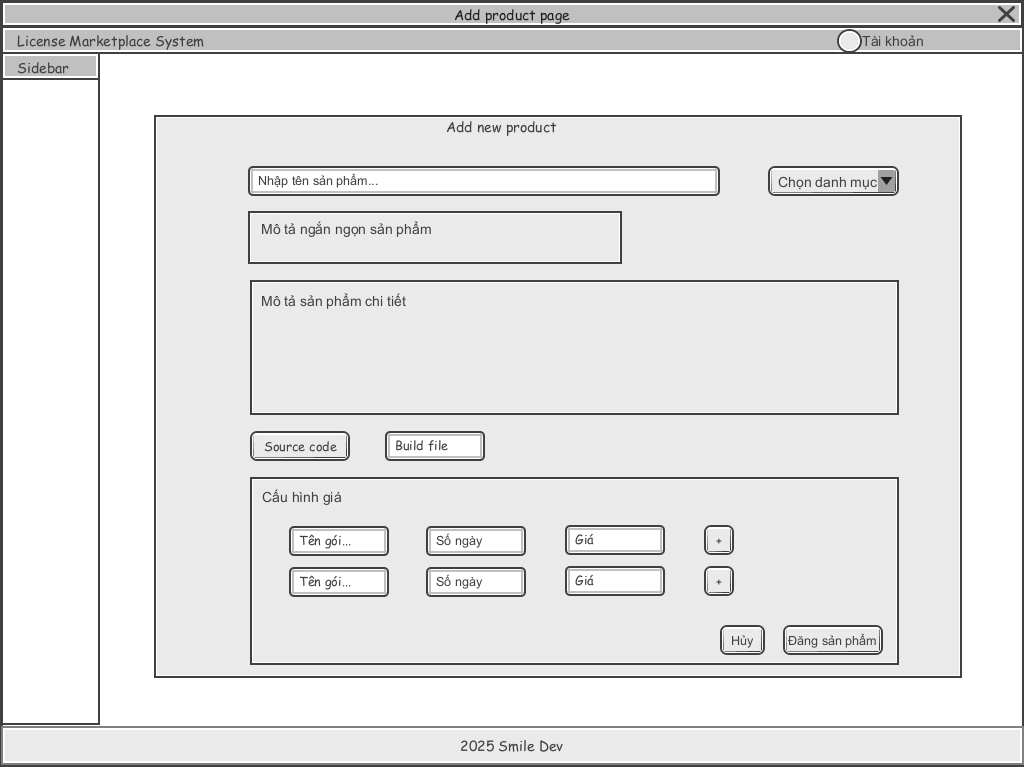
This screen allows the Guest/Buyer to:

* **View Full Information:** Read detailed description (HTML supported), view screenshots, and watch the demo\_video\_url.
* **Select License Package:** Choose a license duration suitable for their needs (e.g., 1 Day, 1 Month, Lifetime).
* **Purchase Product:** Click to buy the selected package. If the balance is insufficient, redirect to Deposit page.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| **Product Info Section** | |
| **Demo Video Player** | Video Player. Plays the video from demo\_video\_url to prove the software works. |
| **Description Content** | Rich Text View. Displays full product details, usage instructions, and features. |
| **Purchase Section** | |
| **Package Selector** | Dropdown/Radio Button. Lists available packages from ProductPackages table (e.g., "1 Month - 50,000 VND", "Lifetime - 1,000,000 VND"). |
| **Price Display** | Label. Updates dynamically based on the selected Package Selector. |
| **"Buy Now" Button** | Button. 1. Checks if User is logged in.2. Checks if wallet\_balance >= Price.3. Triggers the order creation process. |
| **Review Section** |  |
| **User Reviews** | List. Displays comments and ratings from other users in Reviews table. |

### 3.2 Feature: Product Management (Developer)

### 3.2.1 Screen: Create New Product



This screen allows the Developer to:

* **Enter Product Information:** input detailed metadata such as product name, category, short description, and full rich-text description.
* **Upload Software Files:** select and upload the required Source Code file (for security scanning) and Build/Installer file (for user download).
* **Configure Pricing:** add multiple pricing packages (e.g., Monthly, Lifetime) with specific durations and costs.
* **View Validation Errors:** see real-time feedback on invalid file formats, size limits, or missing required fields before submission.

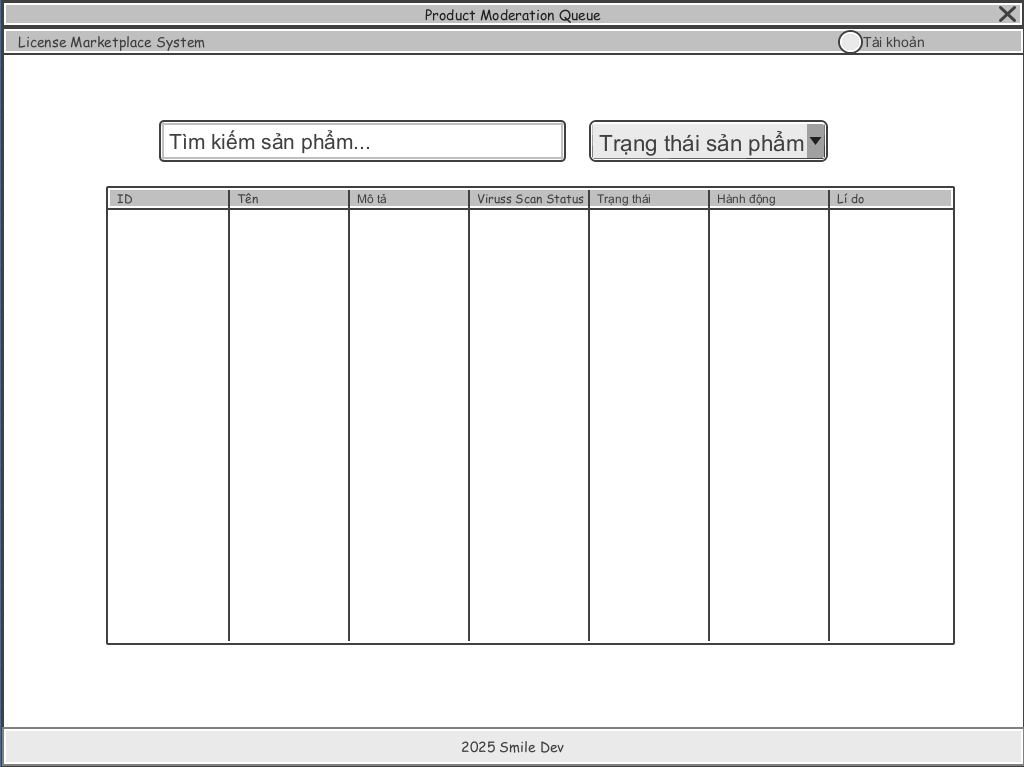
On the screen, s/he can also:

* **Submit Product:** trigger the system to save the product data and initiate the automated virus scanning process.
* **Choose to go to the Product List:** cancel the creation process and return to the *Manage Products* screen by clicking the **Cancel** or **Back** button.

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| **Product Information** | |
| **Product Name** | Input Text. Required. Max length: 150 chars. Unique within the developer's portfolio. |
| **Category** | Dropdown. Required. User selects one value from the system categories list. |
| **Full Description** | Rich Text Editor (HTML). Required. Allows text formatting, images for details. |
| **Demo Video URL** | Input Text. Optional. Must be a valid URL format (YouTube/Vimeo link). |
| **File Uploads** | |
| **Source Code File** | File Picker. Required. Allowed extensions: .zip, .rar. Used for Admin review & Virus Scan. |
| **Build/Installer File** | File Picker. Required. Allowed extensions: .exe, .msi. Used for Buyer download. |
| **Pricing Configuration** | |
| **Package Name** | Input Text. Required. E.g., "Monthly Subscription", "Lifetime". |
| **Duration (Days)** | Input Number. Optional. If empty, it means "Lifetime". |
| **Price ($)** | Input Number. Required. Min value: 0. Format: Decimal (10,2). |
| **Submit Button** | Action Button. Triggers validation and upload. Disabled if fields are empty. |

### 3.3 Feature: Admin Dashboard (Role Admin)

### 3.3.1 Screen: Product Moderation Queue (Duyệt sản phẩm)



This screen is the safety gate of the system. It allows the Admin to:

* **View Pending Submissions:** See a list of new software or updates submitted by Developers waiting for approval .
* **Audit Security:** Check the automated **Virus Scan Result** (integrated with VirusTotal/SonarQube) and download the Source Code/Build File for manual inspection in a Sandbox environment .
* **Approve/Reject:** Make the final decision. If rejected, the Admin must provide a specific reason for the Developer to fix .

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| **Submission List** | |
| **Filter Status** | Dropdown. Default selected: "Pending". Other options: "Approved", "Rejected". |
| **Product Name** | Label. Name of the submitted software. |
| **Developer Name** | Label. Name of the uploader. |
| **Submission Date** | Label. Date and time of upload. |
| **Audit Detail (Popup/Page)** | |
| **Virus Scan Status** | Label/Badge. Displays the result from the automated API (e.g., "Clean", "Infected", "Warning") . |
| **Scan Report Link** | Hyperlink. Redirects to the detailed report on VirusTotal (external link) . |
| **Source Code Download** | Button. Downloads the .zip source code file. Only visible to Admin for code review . |
| **Build File Download** | Button. Downloads the .exe/.dmg file to test in a Sandbox/VM environment . |
| **Action Section** | |
| **Approve Button** | Button. Changes product status to Approved. Triggers the system to generate Product ID and enable Key generation . |
| **Reject Button** | Button. Opens the "Rejection Reason" input field. |
| **Rejection Reason** | Text Area. Required if "Reject" is clicked. The Admin inputs why the product failed (e.g., "Malware detected", "Demo video fake") . |

### 3.3.2 Screen: System Overview & User Management

This screen allows the Admin to:

* **Monitor System Health:** View total revenue, total active keys, and new user registrations .
* **Manage Users:** View list of all users (Devs and Buyers).
* **Ban/Unban Users:** Lock accounts that violate policies (e.g., scammers, distributing keys illegally) .

|  |  |
| --- | --- |
| **Field Name** | **Description** |
| **Statistics Cards** | |
| **Total Revenue** | Label. Sum of all completed transactions (Marketplace Commission + Dev Revenue). |
| **Active Licenses** | Label. Total number of keys currently in Active status. |
| **Pending Approvals** | Label. Number of products waiting in the Moderation Queue. |
| **User Management Table** | |
| **Search User** | Input Text. Search by Username or Email. |
| **User Role** | Label. Displays role: Developer, User, or Admin. |
| **Wallet Balance** | Label. Current money available in the user's wallet. |
| **Account Status** | Badge. Active (Green) or Banned (Red). |
| **"Ban/Unban" Button** | Button. Toggles the is\_active status of the user account. Banned users cannot login or use their keys . |
| **Activity Log** | |
| **Recent Activities** | List/Table. Shows recent critical actions (e.g., "Admin A approved Product B", "User C reported scam") fetched from ActivityLogs table . |

## 4. Non-Functional Requirements

### 4.1 External Interfaces

#### 4.1.1 Software Interfaces (APIs & Services)

The system must interact with the following external services:

* **Virus Scanning Service:** The system shall communicate with **VirusTotal API** (or SonarQube) to automatically scan uploaded source code/build files for malware before Admin review.
  + *Input:* File hash or binary file.
  + *Output:* Scan result (Clean/Infected).
* **Payment Gateways:** The system must integrate with **MOMO, VNPAY, and PAYPAL** APIs to process user deposits and purchases.
* **Developer SDK Integration:** The system provides a RESTful API endpoint for Client Applications (software sold on the market) to validate License Keys.
  + *Protocol:* HTTPS (JSON).
  + *Data exchange:* License Key, Hardware ID (HWID)

#### 4.1.2 Hardware Interfaces

**Client Hardware Identification:** The system (specifically the SDK) must be able to read unique hardware identifiers (HWID) from the User’s computer (e.g., Mainboard Serial, CPU ID) to implement the "Anti-Sharing" mechanism (DRM)

#### 4.1.3 Communication Interfaces

 **Email Service:** The system shall send emails for account verification, password reset, and purchase receipts (Order Confirmation).

 **File Storage:** Large files (Build files, Source code) shall be stored on an external object storage service (e.g., AWS S3) using **Presigned URLs** (temporary links active for 5-10 minutes) for secure downloads.

### 4.2 Quality Attributes

#### 4.2.1 Usability

This section includes all those requirements that affect usability.

* **For Guest/Buyers:**
  + **Ease of Access:** Guests must be able to view product details, filter by category/price, and search without logging in to maximize SEO and marketing effectiveness.
  + **Purchase Flow:** A typical user should be able to complete a purchase (Select Package -> Pay -> Get Key) within **3 minutes** (assuming they have funds in the wallet).
* **For Developers:**
  + **SDK Documentation:** The system must provide comprehensive **API Documentation** and code examples (C#, Java, Python) so Developers can integrate the "Key Check" form into their software without needing complex reverse engineering skills.
  + **Dashboard:** Developers should see the "Rejection Reason" clearly if their product is rejected by Admin.
* **For Admins:**
  + **Visual Cues:** The moderation screen must clearly highlight "Infected" files (Red warning) based on the Virus Scan result to prevent accidental approval

#### 4.2.2 Performance

The system’s performance characteristics are outlined in this section.

* **License Validation Response Time (Critical):**
  + The API for checking license keys (/api/validate-key) must respond within **200 milliseconds** under normal load. This is crucial to ensure the User's software starts up quickly without lagging at the "Checking License..." popup.
* **Throughput & Concurrency:**
  + The system must support at least **1,000 concurrent requests** for key validation per second (since multiple users might open their software simultaneously).
  + *Note:* The KeyValidationLogs table is designed without Foreign Keys to optimize write speed for high-throughput logging.
* **File Download Speed:**
  + File downloads are offloaded to S3/CDN via Presigned URLs, ensuring the main server is not bottled-necked by large file transfers.
* **Database Capacity:**
  + The Licenses and KeyValidationLogs tables must be partitioned or indexed effectively to handle millions of records, as log data grows rapidly over time.

## 5. Requirement Appendix

### 5.1 Business Rules

|  |  |
| --- | --- |
| **ID** | **Rule Definition** |
| BR-01 | Delivery time windows are 15 minutes, beginning on each quarter hour. |
| BR-02 | Deliveries must be completed between 10:00 A.M. and 2:00 P.M. local time, inclusive. |
| BR-03 | All meals in a single order must be delivered to the same location. |
| BR-04 | All meals in a single order must be paid for by using the same payment method. |
| BR-11 | If an order is to be delivered, the patron must pay by payroll deduction. |
| BR-12 | Order price is calculated as the sum of each food item price times the quantity of that food item ordered, plus applicable sales tax, plus a delivery charge if a meal is delivered outside the free delivery zone. |
| BR-24 | Only cafeteria employees who are designated as Menu Managers by the Cafeteria Manager can create, modify, or delete cafeteria menus. |
| BR-33 | Network transmissions that involve financial information or personally identifiable information require 256-bit encryption. |
| BR-86 | Only regular employees can register for payroll deduction for any company purchase. |
| BR-88 | An employee can register for payroll deduction payment of cafeteria meals if no more than 40 percent of his gross pay is currently being deducted for other reasons. |

### 5.2 System Messages

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Message code** | **Message Type** | **Context** | **Content** |
| 1 | MSG01 | In line | There is not any search result | *No search results.* |
| 2 | MSG02 | In red, under the text box | Input-required fields are empty | *The \* field is required.* |
| 3 | MSG03 | Toast message | Updating asset(s) information successfully | *Update asset(s) successfully.* |
| 4 | MSG04 | Toast message | Adding new asset successfully | *Add asset successfully.* |
| 5 | MSG05 | Toast message | Confirming email of asset hand-over is sent successfully | *A confirmation email has been sent to {email\_address}.* |
| 6 | MSG06 | Toast message | Resetting asset information successfully | *Return asset(s) successfully.* |
| 7 | MSG07 | Toast message | Deleting asset information successfully | *Delete asset(s) successfully.* |
| 8 | MSG08 | In red, under the text box | Input value length > max length | *Exceed max length of {max\_length}.* |
| 9 | MSG09 | In line | Username or password is not correct when clicking sign-in | *Incorrrect user name or password. Please check again.* |