The Orchid Arcade

Contents

[1. Description of the Application 3](#_Toc180915694)

[Overview 3](#_Toc180915695)

[Definitions 3](#_Toc180915696)

[Product Functions 3](#_Toc180915697)

[User Classes and Characteristics 3](#_Toc180915698)

[2. Detailed Description and Use Cases 4](#_Toc180915699)

[User Use Cases 4](#_Toc180915700)

[Developer/Publisher Use Cases 8](#_Toc180915701)

[3. Requirements specification 10](#_Toc180915702)

[3.1 External interface specifications 10](#_Toc180915703)

[3.2 Functional requirements 10](#_Toc180915704)

[3.3 Non-functional requirements 14](#_Toc180915705)

[4. Misuse cases 15](#_Toc180915706)

[4.1 Spoofing Misuse cases 16](#_Toc180915707)

[4.2 Tampering Misuse Cases 17](#_Toc180915708)

[4.3 Repudiation Misuse Cases 18](#_Toc180915709)

[4.4 Information Disclosure Misuse Cases 19](#_Toc180915710)

[4.5 Denial of Service (DoS) Misuse Cases 21](#_Toc180915711)

[4.6 Elevation of Privilege Misuse Cases 21](#_Toc180915712)

[4.7 Final STRIDE table 22](#_Toc180915713)

[5. References: 23](#_Toc180915714)

Table of Figures

[Figure 1 User and developer general use case 4](#_Toc180914791)

[Figure 2 User account management use case 4](#_Toc180914792)

[Figure 3 Browsing and purchasing games use case 6](#_Toc180914793)

[Figure 4 User library management use case 7](#_Toc180914794)

[Figure 5 Developer game publishing and management use case 9](#_Toc180914795)

[Figure 6 Spoofing misuse case 1 16](#_Toc180914796)

[Figure 7 Spoofing Misuse case 2 17](#_Toc180914797)

[Figure 8 Cross site scripting 17](#_Toc180914798)

[Figure 9 SQL injection 18](#_Toc180914799)

[Figure 10 Man In the Middle 18](#_Toc180914800)

[Figure 11 User repudiation 19](#_Toc180914801)

[Figure 12 Payment system repudiation 19](#_Toc180914802)

[Figure 13 Sniffing misuse case 20](#_Toc180914803)

[Figure 14 Information disclosure on database 21](#_Toc180914804)

[Figure 15 Denial of Service misuse case 21](#_Toc180914805)

# 1. Description of the Application

## Overview

"The Orchid Arcade" (TOA) is a web-based game distribution platform designed to offer a curated selection of cozy and relaxing games. It serves both gamers and indie developers by providing a marketplace where users can purchase, download, and play games, while developers can publish and manage their games. The platform supports various functionalities, including user management, game management, secure transactions, game updates, and community engagement through reviews and ratings.

## Definitions

* **User**: A person who uses the platform to browse, buy, download, and play games.
* **Developer/Publisher**: A person or entity that publishes games on the platform and manages their game listings.

## Product Functions

The primary functions of "The Orchid Arcade" include:

* **User Management**: Account creation, login, and profile management for users and developers.
* **Game Store**: Browsing, searching, and purchasing games by genre, price range, publisher, etc.
* **Game Library**: Downloading, installing, and updating purchased games.
* **Game Publishing**: Developers can upload, manage, and update their games, including descriptions and pricing.
* **Community Features**: Users can leave reviews and rate games to help others in their purchasing decisions.

## User Classes and Characteristics

* **Users**: Casual gamers interested in cozy and relaxing games. Capabilities include account management, browsing and purchasing games, managing game libraries, and engaging in community features.
* **Developers/Publishers**: Indie game developers and publishers aiming to list their games on the platform. They require access to publishing tools, sales analytics, and community engagement options.

# 2. Detailed Description and Use Cases

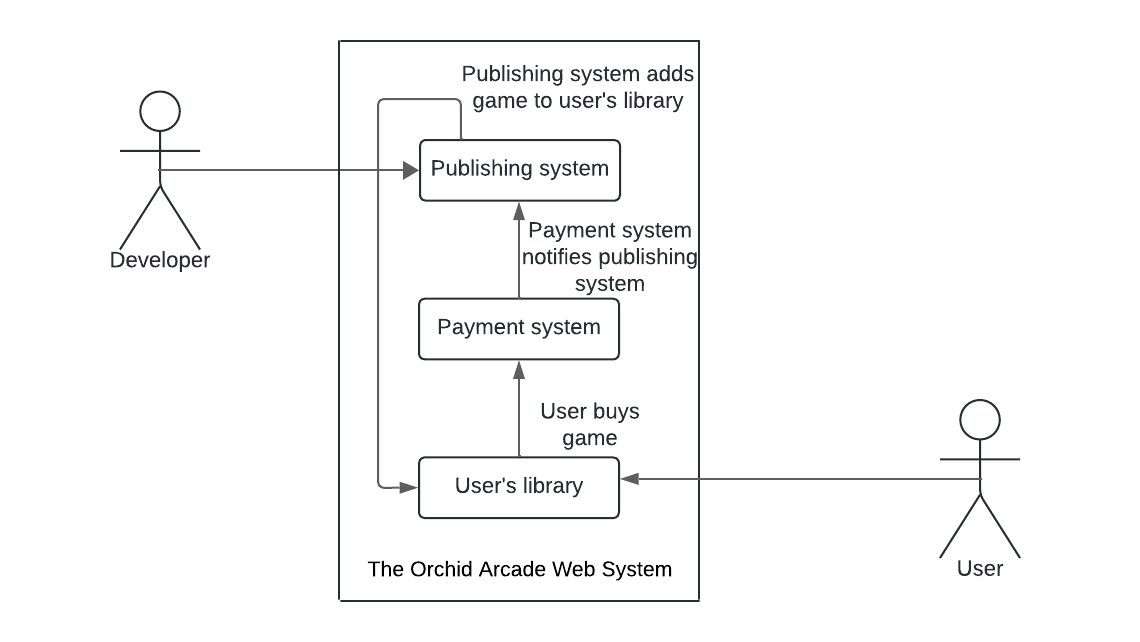


Figure 1 User and developer general use case

## User Use Cases

* **Account Management**:

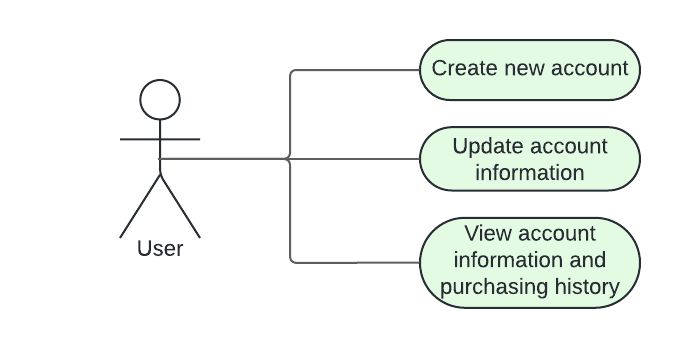


Figure 2 User account management use case

**Brief Description**

The user accesses The Orchid Arcade and performs account-related actions such as creating, editing, and deleting their account, managing their profile, or viewing purchase history.

**Initial Step-By-Step Description**

Before this use case can be initiated, the user has already accessed The Orchid Arcade website.

1. The user selects the account management option.
2. The system displays account management functionalities (create, edit, delete account, change password, update information, etc.).
3. The user chooses a specific action (e.g., create an account, update profile).
4. The system processes the request and updates the user account.
5. If the action is successful, the system confirms the update.
6. If the user views purchase history, the system displays transaction details.

**Alternate and Error Flows**

1. If the email provided during account creation is already registered, the system prompts the user to log in or use a password recovery process.
2. If a server or network issue occurs during the process, the system shows an error message and suggests trying again later.

Xref: Section 3.2.1, Account Management.

* **Browsing and Purchasing Games**:

A diagram of a game

Description automatically generated

Figure 3 Browsing and purchasing games use case

**Brief Description**  
The user browses games available on The Orchid Arcade, views detailed game pages, and purchases selected games.

**Initial Step-By-Step Description**

Before this use case can be initiated, the user has already accessed The Orchid Arcade and logged in (if necessary).

1. The user searches for games by genre, popularity, new releases, or developer.
2. The system displays the results based on the search criteria.
3. The user selects a game and views its detailed page, including reviews and ratings.
4. The user chooses to purchase the game.
5. The system processes the payment securely.
6. Upon successful payment, the system adds the game to the user's library.

**Alternate and Error Flows**

1. If the user has insufficient funds or their payment method is declined, the system prompts them to add a different payment method.
2. If an error occurs during payment processing, the system notifies the user and suggests retrying later.

* **Game Library Management**:

A diagram of a check library

Description automatically generated

Figure 4 User library management use case

**Brief Description**  
The user manages their purchased games, downloading, installing, uninstalling, or updating games in their library.

**Initial Step-By-Step Description**  
Before this use case can be initiated, the user has already purchased one or more games.

1. The user selects the game library option.
2. The system displays the user’s purchased games.
3. The user chooses to download, install, uninstall, or update a game.
4. The system processes the selected action.
5. The system confirms successful download, installation, or update.

**Alternate and Error Flows**

1. If a download is interrupted due to a network error, the system pauses the download and allows the user to retry.
2. If there is insufficient disk space for downloading a game, the system notifies the user and suggests freeing up space.

Xref: Section 3.2.3, Game Library Management.

* **Community Engagement**:

**Brief Description**  
The user interacts with the community by leaving reviews and ratings for games they have purchased on The Orchid Arcade.

**Initial Step-By-Step Description**  
Before this use case can be initiated, the user has already purchased and played one or more games.

1. The user selects a purchased game from their library.
2. The system displays an option to leave a review and rating for the game.
3. The user writes a review and assigns a rating.
4. The system submits the review and rating, making it visible to other users.
5. The system confirms successful submission of the review.

**Alternate and Error Flows**

1. If the user has not purchased the game, the system does not allow them to leave a review or rating.
2. If there is a network or server issue, the system shows an error message and suggests retrying later.

Xref: Section 3.2.6, Community Engagement.

## Developer/Publisher Use Cases

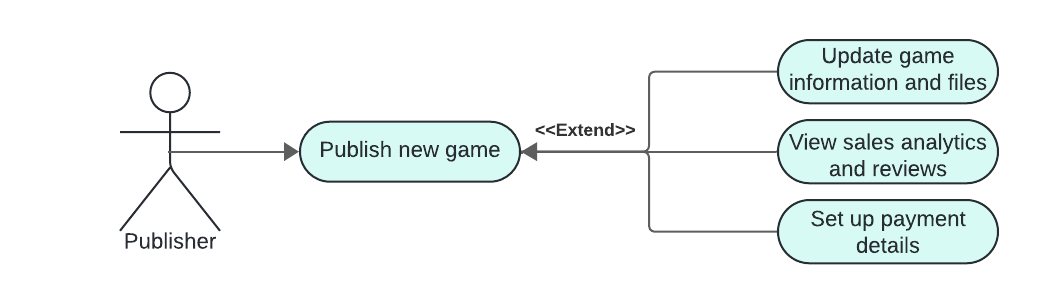


Figure 5 Developer game publishing and management use case

* **Game Publishing and Management**:

**Brief Description**  
The developer uploads and manages games on The Orchid Arcade, including game files, descriptions, and prices.

**Initial Step-By-Step Description**  
Before this use case can be initiated, the developer has created an account and logged in.

1. The developer selects the publishing option.
2. The system displays fields for uploading game files, descriptions, screenshots, and trailers.
3. The developer uploads the game and inputs the required information.
4. The system verifies and publishes the game to the store.
5. The developer can manage or update the game after publishing.

**Alternate and Error Flows**

1. If the uploaded file format is not supported, the system notifies the developer to upload a supported format.
2. If the price format is invalid, the system prompts the developer to enter a valid price.

Xref: Section 3.2.4, Game Publishing and Management.

* **Sales and Revenue Management**:

**Brief Description**  
The developer views sales reports and manages revenue generated from games published on The Orchid Arcade.

**Initial Step-By-Step Description**  
Before this use case can be initiated, the developer has published one or more games.

1. The developer selects the sales and revenue management option.
2. The system displays sales reports, revenue generated, and payment details.
3. The developer views and manages payment information.

**Alternate and Error Flows**

1. If an error occurs while generating a sales report, the system shows an error message and suggests retrying it after some time.

Xref: Section 3.2.5, Sales and Revenue Management.

# 3. Requirements specification

## 3.1 External interface specifications

The only external interface for The Orchid Arcade is the integration with an external payment provider (such as PayPal, Stripe, or another service) to securely process all user transactions. The interface will handle real-time communication with the payment provider for purchase transactions and verification of payment statuses and should accept credit/debit cards and digital wallets.

## 3.2 Functional requirements

**3.2.1 Manage User Account**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Manage User Account |
| **XRef** | Section 3.2.1, Account Management |
| **Trigger** | User selects the account management option. |
| **Precondition** | User has accessed The Orchid Arcade. |
| **Basic Path** | 1. User selects account management.  2. System displays account options (create, edit, delete).  3. User chooses and completes a specific action.  4. System processes the request.  5. System confirms the action (account created, updated, or deleted). |
| **Alternative Paths** | If creating an account:  - System prompts for personal information.  - System checks if the email is already registered.  - If so, prompts the user to log in or recover the password. |
| **Postcondition** | The account is successfully created, updated, or deleted, and the system reflects the changes. |
| **Exception Paths** | If a server or network error occurs, the system displays an error message and suggests trying again later. |

**3.2.2 Browsing and Purchasing Games**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Search and Buy Games |
| **XRef** | Section 3.2.2, Browsing and Purchasing Games |
| **Trigger** | User selects the search or browse option. |
| **Precondition** | User has accessed The Orchid Arcade. |
| **Basic Path** | 1. User selects search or browse by genre, popularity, new releases, or developer.  2. System displays search results.  3. User selects a game and views its details.  4. User selects the purchase option.  5. System processes payment and adds the game to the user’s library. |
| **Alternative Paths** | - If searching by different criteria (genre, popularity, etc.), the system adjusts the results.  - If the game is added to a wishlist instead of being purchased, the system reflects the change. |
| **Postcondition** | The game is added to the user’s library upon successful payment. |
| **Exception Paths** | - If insufficient funds or declined payment, the system prompts the user to change the payment method. |
| - If a payment processing error occurs, the system suggests retrying later. |

**3.2.3 Game Library Management**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Manage Game Library |
| **XRef** | Section 3.2.3, Game Library Management |
| **Trigger** | User accesses their game library. |
| **Precondition** | User has purchased games and logged into their account. |
| **Basic** Path | 1. User opens the game library.  2. System displays a list of purchased games.  3. User selects a game to download, install, uninstall, or update.  4. System processes the action and confirms completion. |
| **Alternative Paths** | If the user selects to update a game, the system checks for available updates and processes the request. |
| **Postcondition** | The game is downloaded, installed, uninstalled, or updated successfully. |
| **Exception Paths** | - If the download is interrupted due to a network error, the system pauses the download and provides a retry option.  - If there is insufficient disk space, the system shows an error message. |

**3.2.4 Game Publishing and Management**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Publish and Manage Games |
| **XRef** | Section 3.2.4, Game Publishing and Management |
| **Trigger** | Developer selects the publishing option. |
| **Precondition** | Developer has logged in to their account. |
| **Basic Path** | 1. Developer selects the publishing option.  2. System displays fields to upload game files, descriptions, and media.  3. Developer uploads the game and inputs details.  4. System verifies and publishes the game to the store.  5. Developer can manage or update the game after publishing. |
| **Alternative Paths** | If a developer chooses to update an existing game, the system reflects the changes in the game store. |
| **Postcondition** | The game is successfully uploaded and published in the store. |
| **Exception Paths** | - If the file format is not supported, the system displays an error and prompts for a valid file format.  - If an invalid price is input, the system prompts the developer to enter a valid price. |

**3.2.5 Sales and Revenue Management**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Manage Sales and Revenue |
| **XRef** | Section 3.2.5, Sales and Revenue Management |
| **Trigger** | Developer accesses the sales and revenue management page. |
| **Precondition** | Developer has published one or more games. |
| **Basic Path** | 1. Developer selects the sales and revenue option.  2. System displays sales reports and revenue data.  3. Developer views or manages payment information. |
| **Alternative Paths** | None |
| **Postcondition** | Developer successfully views sales reports and revenue details. |
| **Exception Paths** | - If an error occurs while generating the sales report, the system displays an error message and suggests retrying later. |

**3.2.6 Review and rate games**

|  |  |
| --- | --- |
| **Field** | **Description** |
| **Use Case Name** | Review and Rate Games |
| **XRef** | Section 3.2.6, Review and Rate Games |
| **Trigger** | User selects a purchased game from their library to leave a review and rating. |
| **Precondition** | User has purchased and played the game. |
| **Basic Path** | 1. User opens a purchased game.  2. System displays an option to leave a review and rating.  3. User submits a review and assigns a rating.  4. System processes the review and makes it visible to other users. |
| **Alternative Paths** | None |
| **Postcondition** | The review and rating are successfully submitted and visible to others. |
| **Exception Paths** | - If the user has not purchased the game, the system prevents them from leaving a review.  - If a network error occurs, the system displays an error message and suggests trying again later. |

## 3.3 Non-functional requirements

* Performance Requirements: Every operation made on the application, including payments, searches and updates, should take no more than 5 seconds under normal network conditions.
* Usability Requirements: Users should receive immediate feedback on transaction outcomes (success, failure, or pending status) for every operation made on the application
* Reliability Requirements: All the interfaces must have a minimum uptime of 99.9%, ensuring that the services are available without interruption.
* Security Requirements:
  + Authentication and Authorization: Implement a robust authentication and authorization system with support for two-factor authentication (2FA) to secure access.
  + Input Sanitization: All user inputs must be sanitized to prevent injection attacks, including SQL injection and cross-site scripting (XSS).
  + Logging and Auditing: Log all user transactions and critical events, such as login attempts and changes to account or application data, for auditing and traceability.
  + Rate Limiting: Apply rate limiting on requests to protect against abuse, such as excessive login attempts or resource-intensive actions.
  + Session Management: Ensure secure session handling, including session expiration and protection against session hijacking.
  + Data Encryption:
    - Encryption in Transit: Enforce HTTPS across all data exchanges to protect data integrity and confidentiality in transit.
    - Encryption at Rest: Encrypt all sensitive data stored in the database, including personally identifiable information (PII) and payment information.

# 4. Misuse cases

We used the Microsoft Threat tool to give some pointers on possible threats to our application using the following model:

A diagram of a web server

Description automatically generated

Figure 6 Flow modeling of the application

According to this some of the feasible attacks on the application might be the following:

## Spoofing Misuse cases

* Spoofing the Client External Entity

**Actors Involved**: Attacker, Web Application, Customer

Since no authentication mechanisms are in place, an attacker may impersonate legitimate users, gaining unauthorized access to accounts and sensitive information.

* Spoofing the Developer External Entity

**Actors Involved**: Attacker, Web Application, Developer

Since no authentication mechanisms are in place, an attacker could gain control over game listings, altering game availability or pricing.

A diagram of a web application

Description automatically generated

Figure 6 Spoofing misuse case 1

* Spoofing the Web Application Process

**Actors Involved**: Attacker, Web Application, User

Since HTTPS is not being enforced currently, the web application is not being authenticated to the user via the certificate, meaning that an attacker could spoof the web service identity to steal user information or credentials.

A diagram of a person with text

Description automatically generated

Figure 7 Spoofing Misuse case 2

## Tampering Misuse Cases

* Cross Site Scripting

**Actors Involved**: Attacker, Web Application, User

The web server 'Web Application' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input when a developer enters game information.

A diagram of a computer program

Description automatically generated

Figure 8 Cross site scripting

* SQL Injection for SQL Database

**Actors Involved**: Attacker, Web Application, User

An attacker could inject malicious SQL code, accessing or manipulating stored data because the inputs are not being sanitized.

A close-up of a credit card

Description automatically generated

Figure 9 SQL injection

* Data Flow Compromised

**Actors Involved**: Attacker, Web Application, User

Since HTTPS is not being enforced an attacker can read or modify the data being transmitted.

A diagram of a person with a line

Description automatically generated

Figure 10 Man In the Middle

## Repudiation Misuse Cases

* External Entity Client Potentially Denies Receiving Data

**Actors Involved**: Attacker, Web Application, Client

Since there is no logging in place, a user can deny having purchased a game and try to ask for money returns even though they did purchase the game.

A diagram of a game

Description automatically generated

Figure 11 User repudiation

* External Entity Payment System Potentially Denies Receiving Data

**Actors Involved**: Web Application, Payment system

The external payment system could deny receiving payment data from the user since there are no logs or records of the information being sent.

A close-up of a sign

Description automatically generated

Figure 12 Payment system repudiation

## Information Disclosure Misuse Cases

* Data Flow Sniffing

**Actors Involved**: Attacker, Web Application, User

An attacker could sniff all the packets with the information being sent by the users, including private information like credit card information; this is because HTTPS is not enforced meaning that the traffic is not encrypted.

A diagram of a person with a credit card

Description automatically generated

Figure 13 Sniffing misuse case

* Reading information from SQL Database

**Actors Involved**: Attacker, Web Application

Since the SQL Database does not encrypt vulnerable information like passwords at rest, an attacker can read all this information in plain text.

A diagram of a person with a blue cylinder

Description automatically generated

Figure 14 Information disclosure on database

## Denial of Service (DoS) Misuse Cases

* **Potential Excessive Resource Consumption for Web Server or SQL Database**

**Actors Involved**: Attacker, Web Application

An attacker could cause high traffic or resource consumption, crashing the application or causing slowdowns.



Figure 15 Denial of Service misuse case

## Elevation of Privilege Misuse Cases

* Elevation Using Impersonation

**Actors Involved**: Attacker, Web Application

Since no session information is being used, an attacker could send their request as if they were a specific customer or developer. This is a similar situation to the one shown in figure 6.

## 4.7 Final STRIDE table

Considering the mentioned abuse cases given by the Microsoft Thread Modeling tool is the final table with each STRIDE category and its possible vulnerabilities:

|  |  |  |
| --- | --- | --- |
| **STRIDE Category** | **Threat Description** | **Threat Level** |
| **Spoofing** | **Spoofing the Client/Developer**: Impersonation to gain unauthorized access to user or developer accounts. | High |
|  | **Spoofing the Web Application**: Attacker mimics the application to steal user credentials or data. | High |
| **Tampering** | **Cross-Site Scripting (XSS)**: Injecting malicious scripts via unvalidated input fields. | Medium |
|  | **SQL Injection**: Direct manipulation of the SQL database through unsanitized inputs. | High |
|  | **Data Flow Compromised**: Attacker intercepts and modifies data in transit due to lack of encryption. | High |
| **Repudiation** | **User Repudiation of Transactions**: Users deny having purchased games without audit trails. | Medium |
|  | **Payment System Repudiation**: Payment provider denies transaction receipt due to lack of records. | Medium |
| **Information Disclosure** | **Data Flow Sniffing**: Attacker intercepts unencrypted network traffic, exposing sensitive data. | High |
|  | **Reading Plaintext Data from Database**: Sensitive data (e.g., passwords) in plaintext is accessible to attackers. | High |
| **Denial of Service (DoS)** | **Excessive Resource Consumption**: High volume of requests overwhelms the server, causing slowdowns or crashes. | High |
| **Elevation of Privileges** | **Elevating privileges to specific user:** An attacker could hijack the account of any user to elevate its privileges. | High |

# 5. References:

* Lane, G. K. C. (2023, January 17). How to write an SRS Document (Software Requirements Specification Document). Perforce Software. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>
* TianyaoHan. (n.d.). GitHub - TianyaoHan/Steam-Recommendation-System: Steam Database Design and Game Recommendation System. GitHub. https://github.com/TianyaoHan/Steam-Recommendation-System