Table Of Contents

[Meeting Minutes 2](#_Toc435049251)

[October 21, 2015 2](#_Toc435049252)

[October 28, 2015 2](#_Toc435049253)

[November 6th 2](#_Toc435049254)

[ERD 3](#_Toc435049255)

[Data Validation 4](#_Toc435049256)

[Data Dictionary 7](#_Toc435049257)

[Design Level Class Diagram 12](#_Toc435049258)

[Business Logic 13](#_Toc435049259)

[Navigation Diagram 19](#_Toc435049260)

[UI Prototypes 20](#_Toc435049261)

[Employees 20](#_Toc435049262)

[Members 24](#_Toc435049263)

[Visitors 28](#_Toc435049264)

[Visitors & Members 29](#_Toc435049265)

[Report Designs 31](#_Toc435049266)

[Deployment Diagram 34](#_Toc435049267)

[Security (Threat Modeling Report) 35](#_Toc435049268)

# Meeting Minutes

## October 21, 2015

#### Meeting canceled due to midterms

## October 28, 2015

Rob and Steve Absent (We had trouble finding a room, they may not have got the memo that we found one)

#### Review

* We had several miniature meetings in this design phase in which we discussed any questions people had, and looked over the prototype template Nicole created
* Looked over deliverables for:
* prototype samples
* Rob had begun report designs, Nicole had reviewed and made suggestions
* Realized we were missing 2 deliverables: 1) security, 2) business logic and data validation

#### During the Meeting

* John volunteered to take care of security
* Nicole volunteered to start business logic and data validation and members agreed that we would complete it together next meeting
* Discussed whether or not to change the game id, because we currently have different ids for the same game on different platforms
* We decided to leave as is and change the report that was showing the same id for a game across all platforms

#### Moving forward

* determined that we are aiming to have all components complete by November 6th

**Next Meeting: November 4th** ### Meeting moved until class time November 6th

## November 6th

Nicole absent (injured), Manny absent (busy)

#### During Meeting

* Completed Data Validation document
* Noted similarities to Dan's Data Dictionary document, Dan planning to adjust

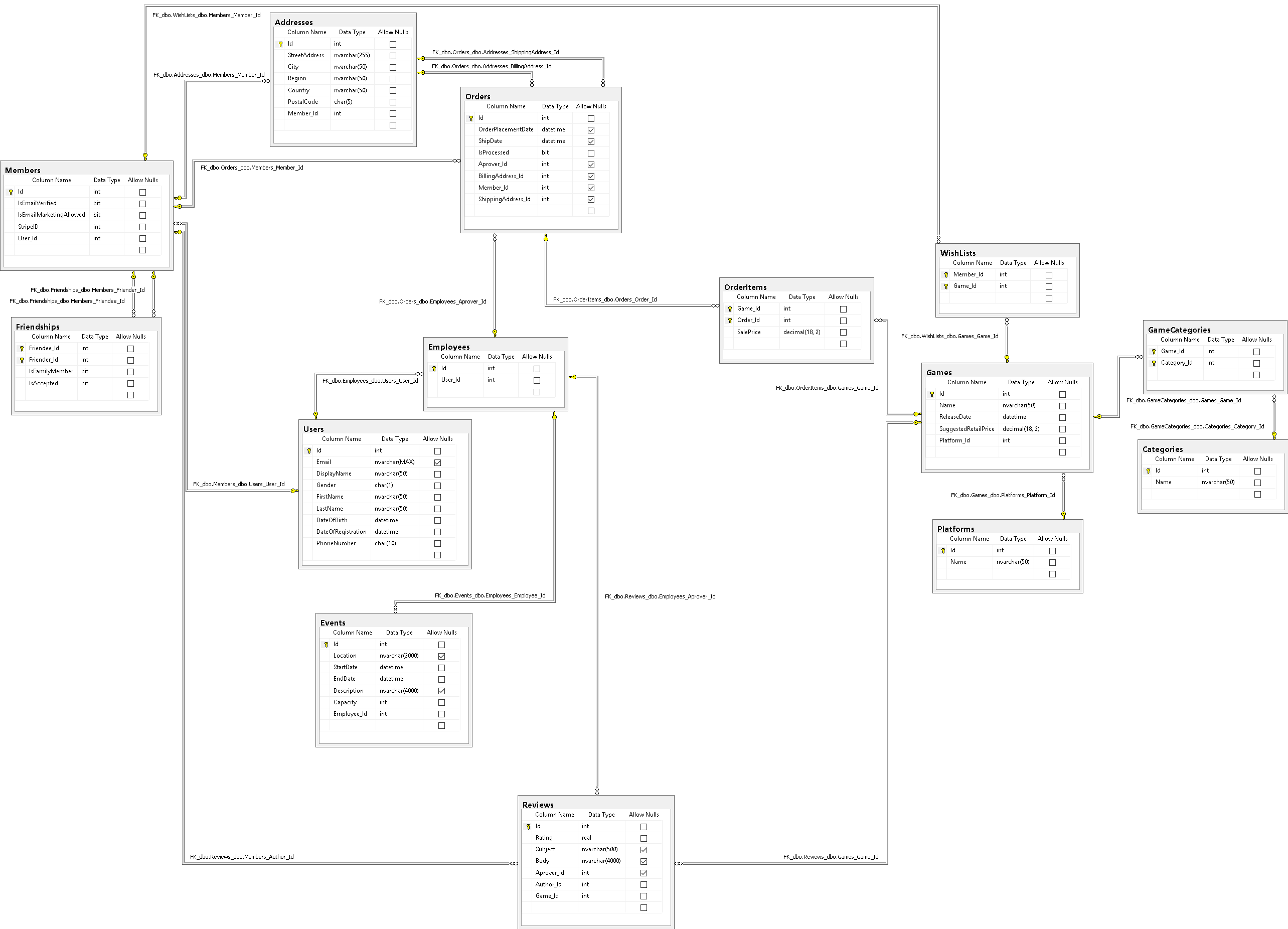
#### Moving Forward

* Peter to put together hand-in document

**Next Meeting November 11th**

# ERD

[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/ER%20Diagram)



# Data Validation

[GitHub Link](https://github.com/SeaSharpe/Design/blob/master/DataValidation.md)

#### User Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| userId | int | 10,000,000 -> 19,999,999 NOT NULL |  |
| email | string | [EMAIL] NOT NULL |  |
| displayName | string | Characters (4, 50) NOT NULL |  |
| gender | string | M, F or O NOT NULL |  |
| firstName | string | Characters (1, 50) NOT NULL |  |
| lastName | string | Characters (1, 50) NOT NULL |  |
| dateOfBirth | dateTime | NOT NULL, Not in future |  |
| registrationDate | dateTime | No Validation | Generated by server |
| phoneNumber | string | [PHONE] NOT NULL | Validated by ASP Regex |

#### Employee Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| employeeId | int | 200,000 -> 299,999 |  |

#### Member Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| memberId | int | 30,000,000 -> 39,999,999 NOT NULL |  |
| stripeId | string | Out Of Scope |  |
| isVerified | boolean | NOT NULL, DEFAULT FALSE | Generated by server |
| prefPlatformId | int | FK Platform Table |  |
| prefCategoryId | int | FK Category Table |  |

#### Friendship Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| friendeeID | int | FK User Table |  |
| frienderID | int | FK User Table |  |
| isFamilyMember | boolean | NOT NULL, DEFAULT FALSE |  |
| isAccepted | boolean | NOT NULL, DEFAULT FALSE |  |

#### 

#### Events Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| eventId | int | 4,000,000 -> 4,999,999, NOT NULL |  |
| employeeId | int | FK Employee Table |  |
| location | string | Characters (0, 2000) |  |
| startDate | dateTime | Not in the past, NOT NULL |  |
| endDate | dateTime | Not before startDate, NOT NULL |  |
| description | string | Characters (0, 4000) |  |
| capacity | int | 0 < VAL < Int(MAX), NOT NULL |  |

#### Address Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| addressId | int | 500,000,000 -> 599,999,999, NOT NULL |  |
| memberId | int | FK Members Table |  |
| address | string | Char (1, 255), NOT NULL | Could validate against Canada Post API |
| city | string | Char (2, 50), NOT NULL |  |
| country | string | Char (2, 50), NOT NULL |  |
| postalCode | string | REGEX, NOT NULL |  |

#### Order Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| orderId | int | 600,000,000 -> 699,999,999, NOT NULL |  |
| memberId | int | FK Members Table NOT NULL |  |
| processorId | int | FK Employees Table | Set after checkout |
| billingAddressID | int | FK Address Table | Set at checkout |
| shippingAddressID | int | FK Address Table | Set at checkout |
| orderPlacementDate | dateTime | System Generated DEFAULT( NOW ) | Set at checkout |
| shipDate | dateTime | Not before order placement | Set at shipped |
| isProcessed | boolean | NOT NULL, DEFAULT FALSE | Set after checkout |

#### 

#### OrderItem Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| gameID | int | FK Game Table |  |
| orderID | int | FK Order Table |  |
| salePrice | decimal | NOT NEGATIVE NOT NULL | Canadian Dollars |

#### Game Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| gameID | int | 7,000,000 -> 7,999,999, NOT NULL |  |
| platformID | int | FK Platform Table, NOT NULL |  |
| name | string | Characters (1, 50), NOT NULL |  |
| releaseDate | dateTime | NOT NULL |  |
| suggestedRetailPrice | decimal | NOT NEGATIVE, NOT NULL |  |
| publisher | string | Characters (1, 50, NOT NULL |  |

#### Platform Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| platformId | int | 800 -> 899 NOT NULL |  |
| name | string | Characters (1, 50) NOT NULL |  |

#### Category Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| categoryId | int | 8,000 -> 8,999 NOT NULL |  |
| name | string | Characters (1, 50) NOT NULL |  |

#### Review Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Validation Rule** | **Notes** |
| reviewId | int | 900,000,000 -> 999,999,999 |  |
| gameId | int | FK Games NOT NULL |  |
| memberId | int | FK Members Table NOT NULL |  |
| approverId | int | FK Employees Table | Null if not yet Assessed |
| rating | float | 0 < VALUE < 1 NOT NULL |  |
| subjectText | string | Characters (1, 500) | Null if just a rating |
| contextText | string | Characters (0, 4000) | Null if just a rating |
| isApproved | boolean | NOT NULL, DEFAULT FALSE |  |

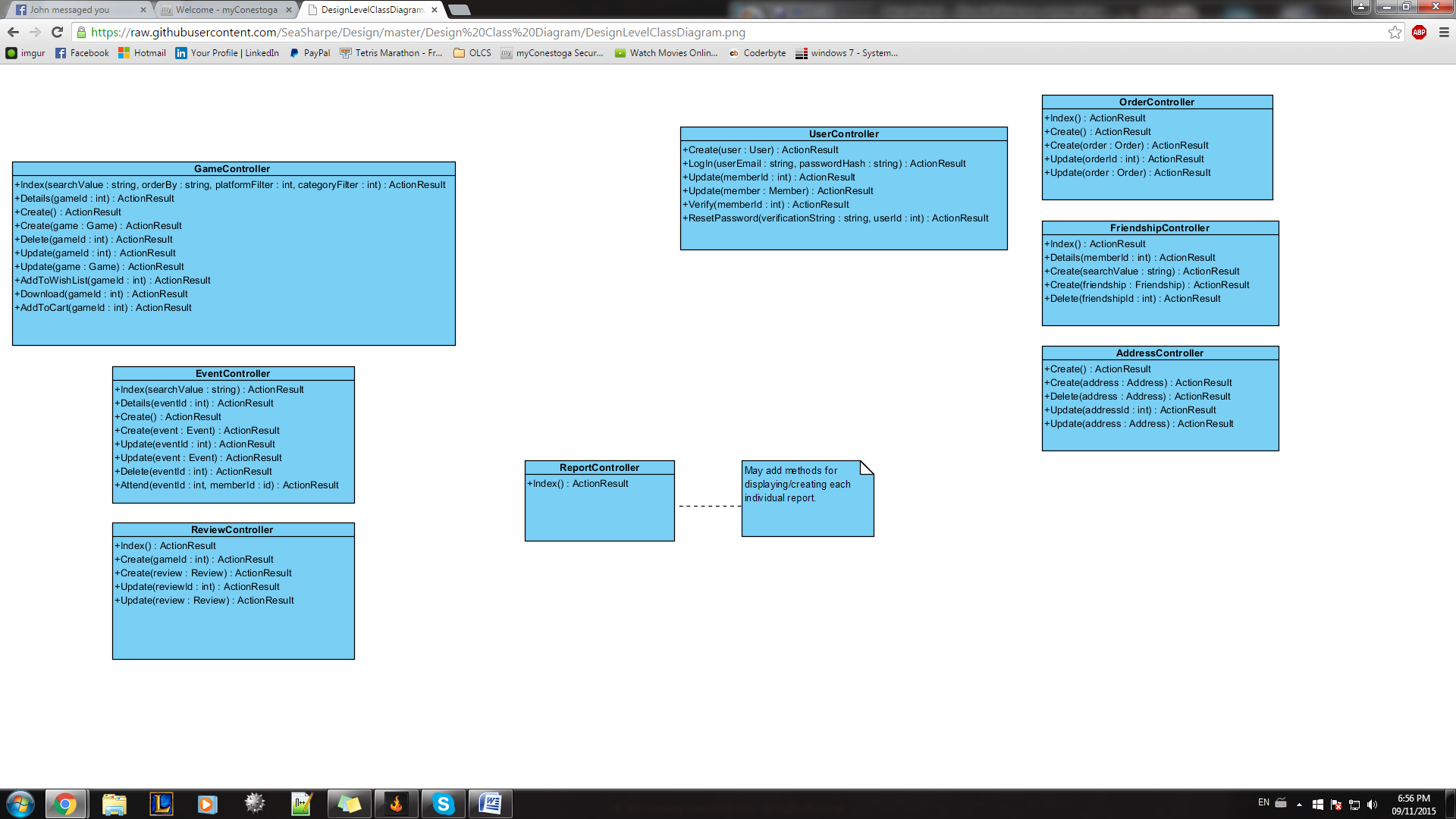
# Data Dictionary

[GitHub Link](https://github.com/SeaSharpe/Design/blob/master/DataDictionary.md)

| **Table** | **PK/FK** | **Field Name** | **Type** | **Unique** | **Nullable** | **Default Value** | **Reqd?** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Users | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Users |  | Email | nvarchar(max) | Yes | No |  | Yes |  |
| Users |  | DisplayName | nvarchar(50) | Yes | No |  | Yes |  |
| Users |  | Gender | char(1) | No | No |  | Yes |  |
| Users |  | FirstName | nvarchar(50) | No | No |  | Yes |  |
| Users |  | LastName | nvarchar(50) | No | No |  | Yes |  |
| Users |  | DateOfBirth | dateTime | No | No |  | Yes |  |
| Users |  | DateOfRegistration | dateTime | No | Yes |  | Yes | Generated by server |
| Users |  | PhoneNumber | char(10) | No | No |  | Yes |  |
| Employees | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Employees | FK | User\_Id | int | Yes | No |  | Yes | ID from User Table |
| Members | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Members |  | IsEmailVerified | boolean | No | No |  | Yes | Generated by server |
| Members |  | IsEmailMarketingAllowed | boolean | No | No |  | Yes |  |
| Members |  | StripeId | int | No | No |  | Yes |  |
| Members | FK | User\_Id | int | Yes | No |  | Yes | ID from User Table |
| Friendships | PK, FK | Friendee\_ID | int | No | No |  | Yes | ID from User Table of request sender |
| Friendships | PK, FK | Friender\_ID | int | No | No |  | Yes | ID from User Table of request recipient |
| Friendships |  | isFamilyMember | bool | No | No |  | Yes |  |
| Friendships |  | isAccepted | bool | No | No | FALSE | Yes |  |
| Events |  | Id | int | Yes | No |  | Yes |  |
| Events |  | Location | nvarchar(2000) | No | Yes |  | Yes |  |
| Events |  | StartDate | dateTime | No | No |  | Yes |  |
| Events |  | EndDate | dateTime | No | No |  | Yes |  |
| Events |  | Description | nvarchar(4000) | No | Yes |  | Yes |  |
| Events | FK | Employee\_Id | int | No | No |  | Yes | ID from Employee Table |
| Events |  | Capacity | int | No | No |  | Yes |  |
| Addresses | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Addresses |  | StreetAddress | nvarchar(255) | No | No |  | Yes |  |
| Addresses |  | City | nvarchar(50) | No | No |  | Yes |  |
| Addresses | FK | Region | nvarchar(50) | No | No |  |  |  |
| Addresses |  | Country | nvarchar(50) | No | No |  | Yes |  |
| Addresses |  | PostalCode | char(5) | No | No |  | Yes |  |
| Addresses | FK | MemberID | int | No | No |  | Yes | ID from Member Table |
| Orders | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Orders |  | OrderPlacementDate | dateTime | No | No |  |  | System Generated DEFAULT( NOW ), Set at checkout |
| Orders |  | ShipDate | dateTime | No | Yes |  |  | Set at shipped |
| Orders |  | IsProcessed | boolean | No | No | FALSE |  | Set after checkout |
| Orders | FK | Approver\_Id | int | No | Yes |  |  | ID from Employee Table, Set after checkout |
| Orders | FK | BillingAddress\_Id | int | No | Yes |  | Yes | ID from Address Table, Set after checkout |
| Orders | FK | Member\_Id | int | No | Yes |  |  | ID from Member Table, Set after checkout |
| Orders | FK | ShippingAddress\_Id | int | No | Yes |  | Yes | ID from Address Table, Set after checkout |
| OrderItems | FK | Game\_ID | int | No | No |  | Yes | ID from Game Table |
| OrderItems | FK | Order\_ID | int | No | No |  | Yes | ID from Order Table |
| OrderItems |  | SalePrice | decimal(18,2) | No | No |  | Yes | Canadian Dollars |
| Games | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Games |  | Name | nvarchar(max) | No | Yes |  | Yes |  |
| Games |  | ReleaseDate | date | No | No |  | Yes |  |
| Games |  | SuggestedRetailPrice | decimal(18,2) | No | No |  | Yes |  |
| Games | FK | Platform\_ID | int | No | Yes |  | Yes | ID from Platform Table |
| WishLists | PK, FK | Member\_Id | int | No | No |  | Yes | ID from Member Table |
| WishLists | PK, FK | Game\_Id | int | No | No |  | Yes | ID from Game Table |
| Platforms | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Platforms |  | Name | nvarchar(50) | Yes | No |  | Yes |  |
| Categories | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Categories |  | Name | nvarchar(max) | Yes | Yes |  | Yes |  |
| GameCategories | FK | Game\_ID | int | No | No |  | Yes | ID from Game Table |
| GameCategories | FK | Category\_Id | int | No | No |  | Yes | ID from Category Table |
| Reviews | PK | Id | int | Yes | No | AutoIncrement | Yes |  |
| Reviews |  | Rating | real | No | No |  | Yes |  |
| Reviews |  | Subject | nvarchar(500) | No | Yes |  | Yes | Null if just a rating |
| Reviews |  | Body | nvarchar(4000) | No | Yes |  | Yes | Null if just a rating |
| Reviews | FK | Approver\_Id | int | No | Yes |  | Yes | ID from Employee Table, Null if review is not yet assessed |
| Reviews | FK | Author\_ID | int | No | No |  | Yes | ID from Member Table |
| Reviews | FK | Game\_ID | int | No | No |  | Yes | ID from Game Table |

# Design Level Class Diagram

[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/Design%20Class%20Diagram)



# Business Logic

[GitHub Link](https://github.com/SeaSharpe/Design/blob/master/BusinessLogic.md)

#### Visitor Signs Up for Account

* Visitor must not have an account registered for that E-mail address
* Sign up fields must not be empty
* Sign up fields must follow data validation rules
* Signs up for account
* Commit to database

#### Member Logs In

* User must have an account created
* Email and password fields much not be empty
* Password entered must match stored password for specified user
* User logs in to site

#### Member/Visitor Views Events

* Login is not required
* User selects Events link
* View list of events
* Select event name to view specific event

#### Member/Visitor Views Game Details

* Login is not required
* User selects Shop link
* Search for games
* Select the game name to display game details
* View reviews/ratings

#### Member Profile

* Must be logged in as Member
* Set up profile information
* Opt in / opt out of promotional E-mails
* Set platform preference
* Set game category preference
* Enter/Edit “ship to” address
* Delete “ship to” address
* Edit address
* User information and address fields must not be empty
* User information and address fields must follow validation rules
* Reset password
* Commit to database

#### Member Purchase Games

* Must be logged in as Member
* Member must have game selected
* Add game to cart
* View cart
* Checkout
* Download games/or select games to be shipped
* Commit to database

#### Member Registers for Event

* Must be logged in as Member
* Member must be viewing event
* Member registers for event
* Commit to database

#### Member Rates/Reviews a Game

* Must be logged in as Member
* Must be on a game’s details page
* Review games - must be approved be employee before commited
* Rate games - Commit to database

#### Member Adds Game to Wish List

* Must be logged in as Member
* Must be on a game’s details page
* Adds game to wish list
* Commit to database

#### Member Adds Another User to Their Friends and Family List

* Must be logged in as Member
* Other user must have an account created
* Search by name
* Add user to Friends and Family List
* Commit to database

#### Member Views Another Member’s Wish List

* Must be logged in as Member
* Other Member must be on Friends and Family List
* View other Member’s wish list

#### Employee Logs In

* User must have an Employee account
* Email and password fields must not be empty
* Entered password must match stored password for specified user
* User logs in to site as Employee

#### Employee Adds Event

* Must be logged in as Employee
* Event must not already exist
* Event fields must not be empty
* Event fields must match validation rules
* Employee adds event
* Commit to database

#### Employee Edits Event

* Must be logged in as Employee
* Must be on an event’s page
* Event fields must not be empty
* Event fields must match validation rules
* Employee edits event
* Commit to database

#### Employee Deletes Event

* Must be logged in as Employee
* Must be on an event’s page
* Employee deletes event
* Removed from database

#### Employee Adds Game

* Must be logged in as Employee
* Game must not already exist
* Game fields must not be empty
* Game fields must match validation rules
* Employee adds game
* Commit to database

#### Employee Edits Game

* Must be logged in as Employee
* Must be on the game’s editable page
* Game fields must not be empty
* Game fields must match validation rules
* Employee edits game
* Commit to database

#### Employee Deletes Game

* Must be logged in as Employee
* Must be on the game’s editable page
* Employee deletes game
* Game removed from database

#### Employee Ships Games/Marks Order as Processed

* Must be logged in as Employee
* Employee must be on orders page
* Employee selects an order
* Employee packages the items on the order
* Employee marks order as processed
* Commit to database

#### Employee Approves Reviews

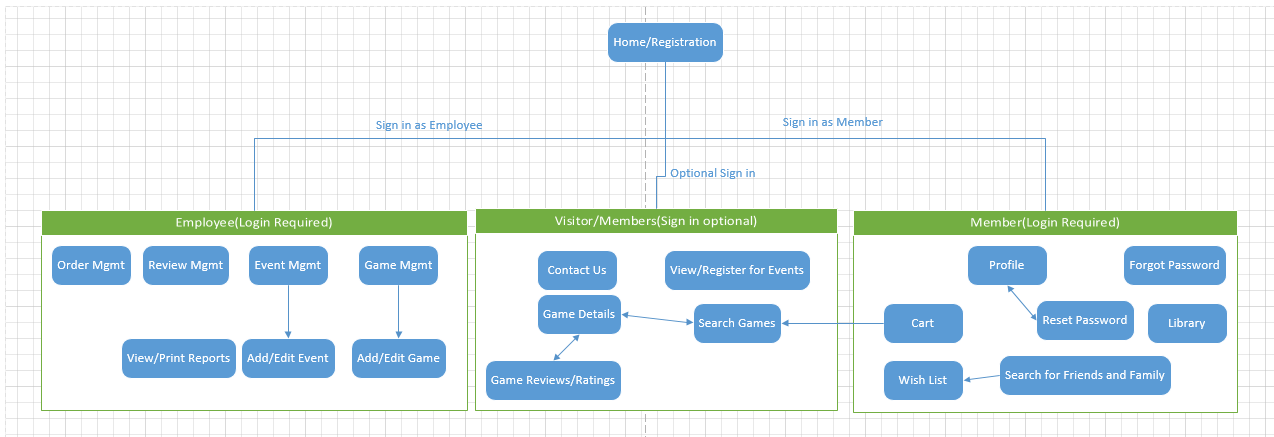
* Must be logged in as Employee
* Employee must be on reviews page
* Employee selects review to review
* Employee reviews review
* Employee approves review
* Commit to database

#### Employee Views Reports

* Must be logged in as Employee
* Employee selects report to view
* Employee views report
* Employee may print report

# Navigation Diagram

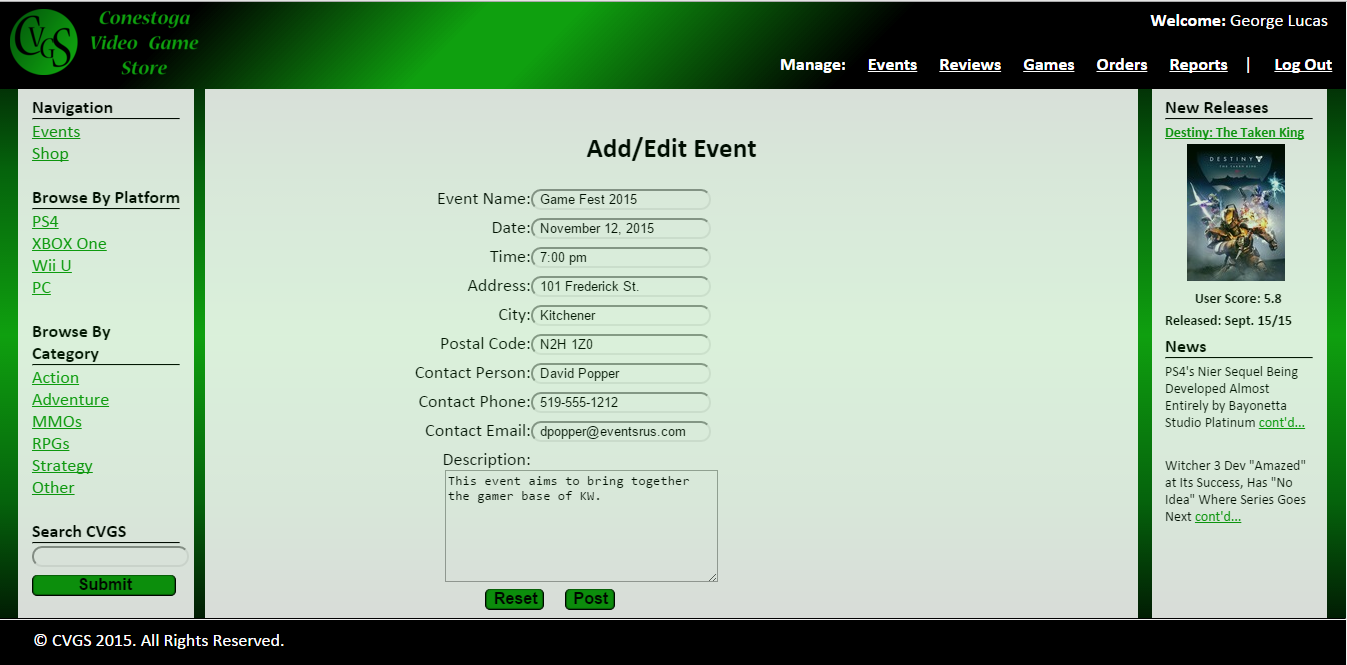
[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/Navigation%20Diagram)

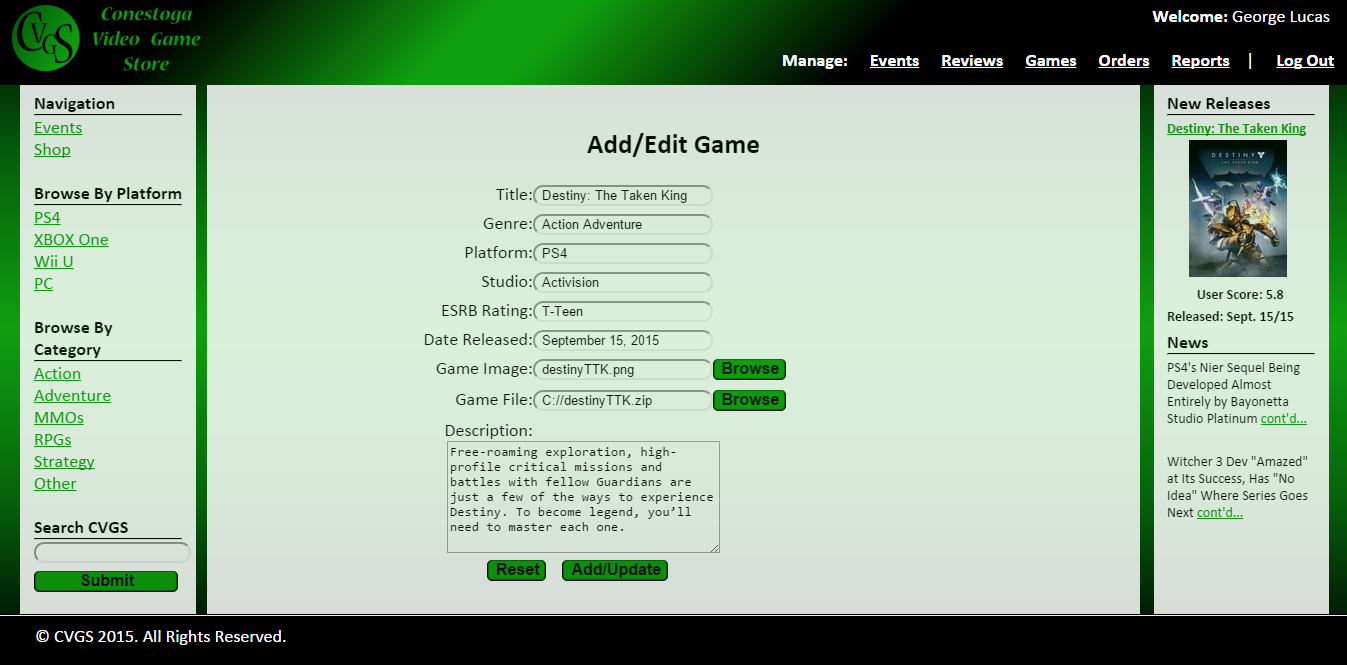
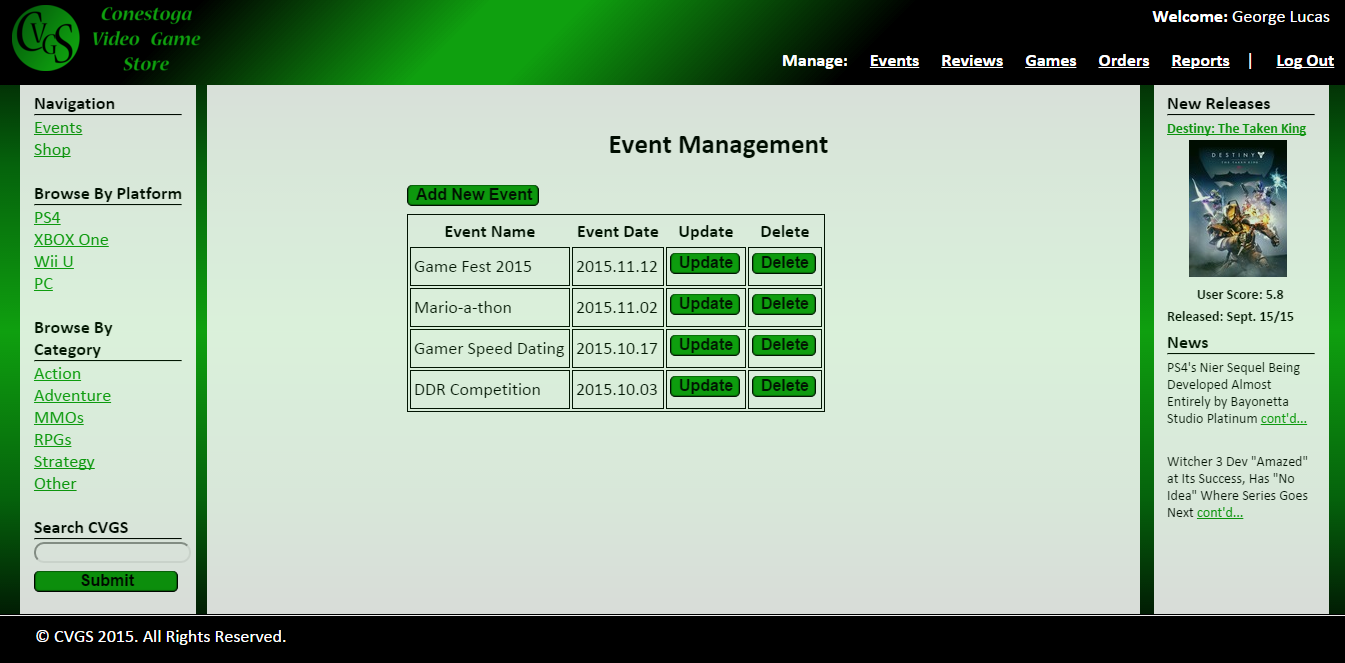


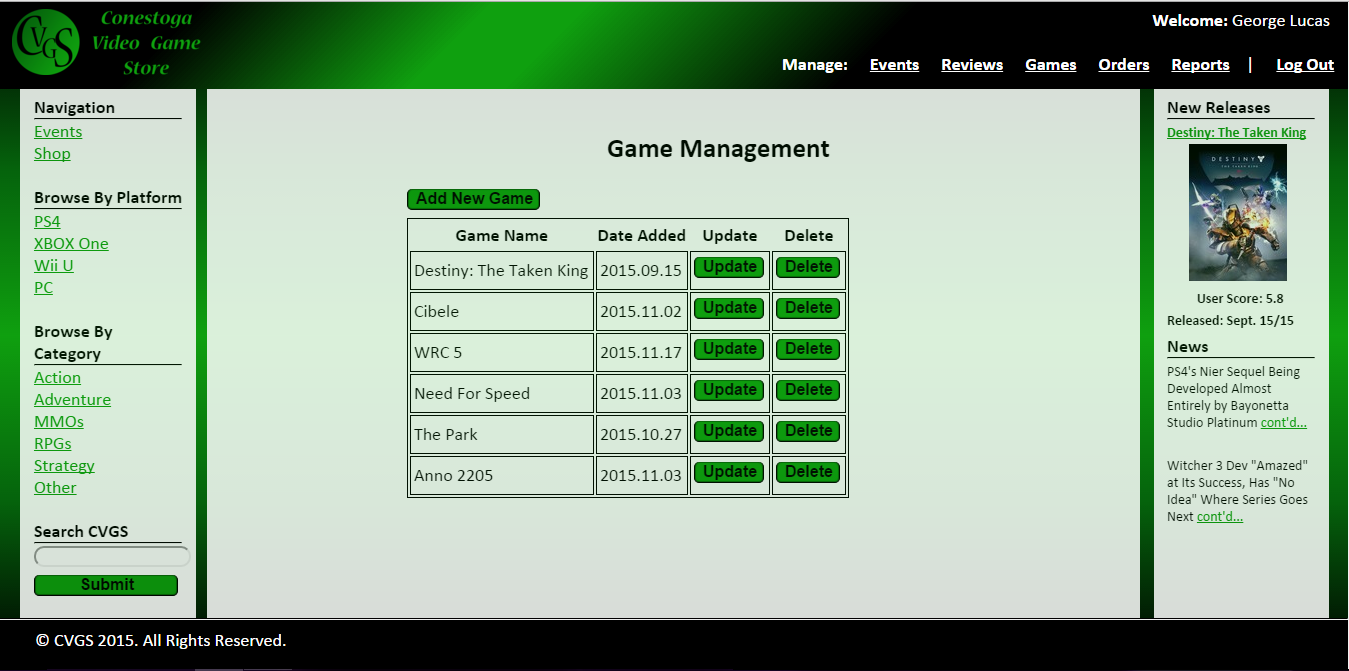
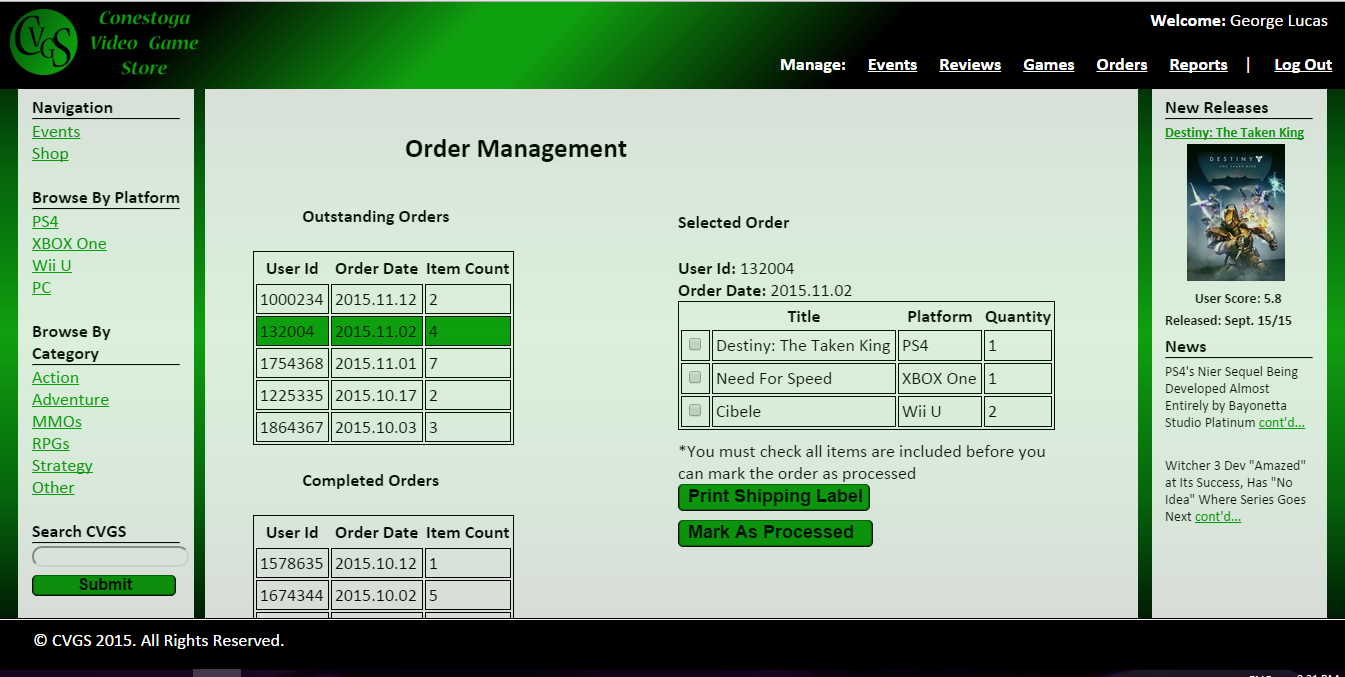
# UI Prototypes

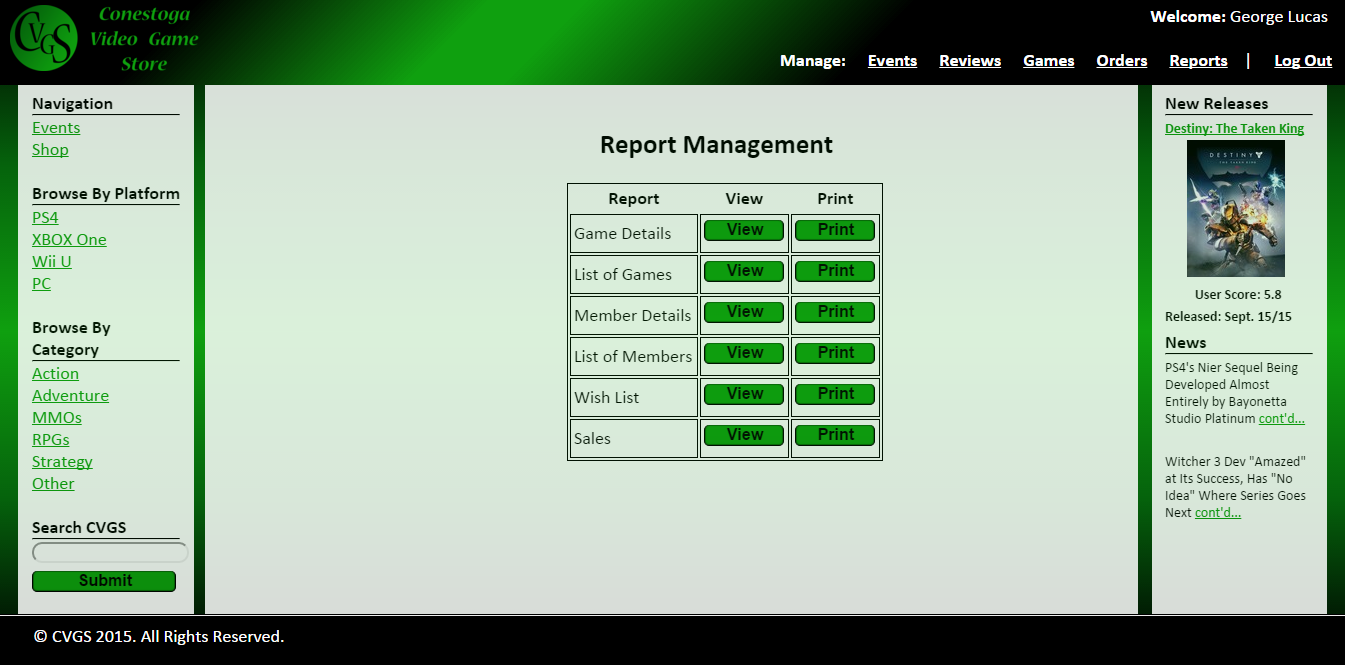
[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/Prototypes)

## Employees

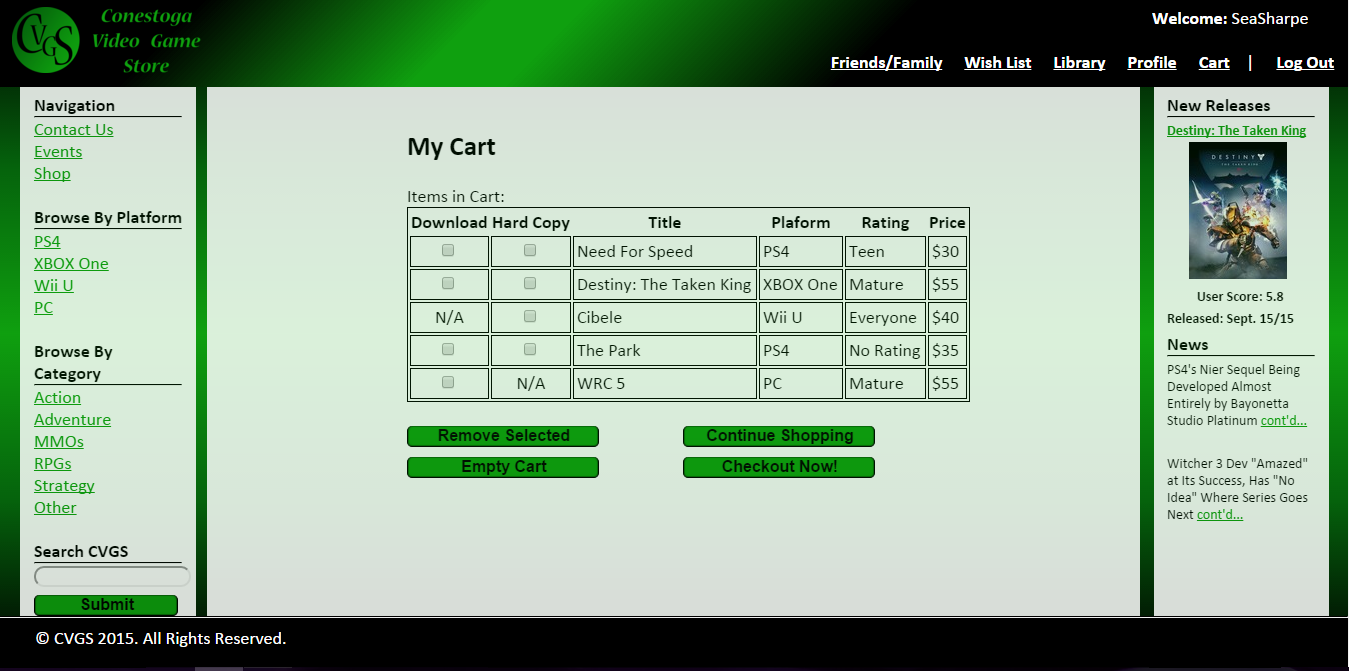


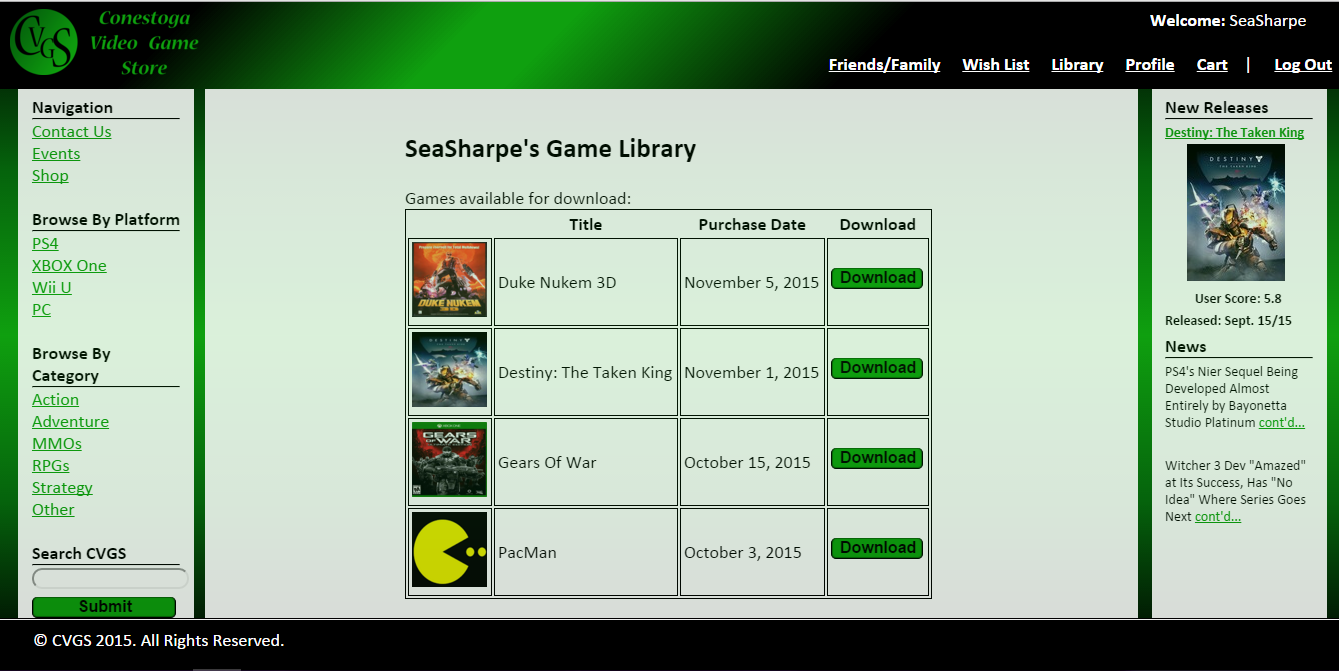
 

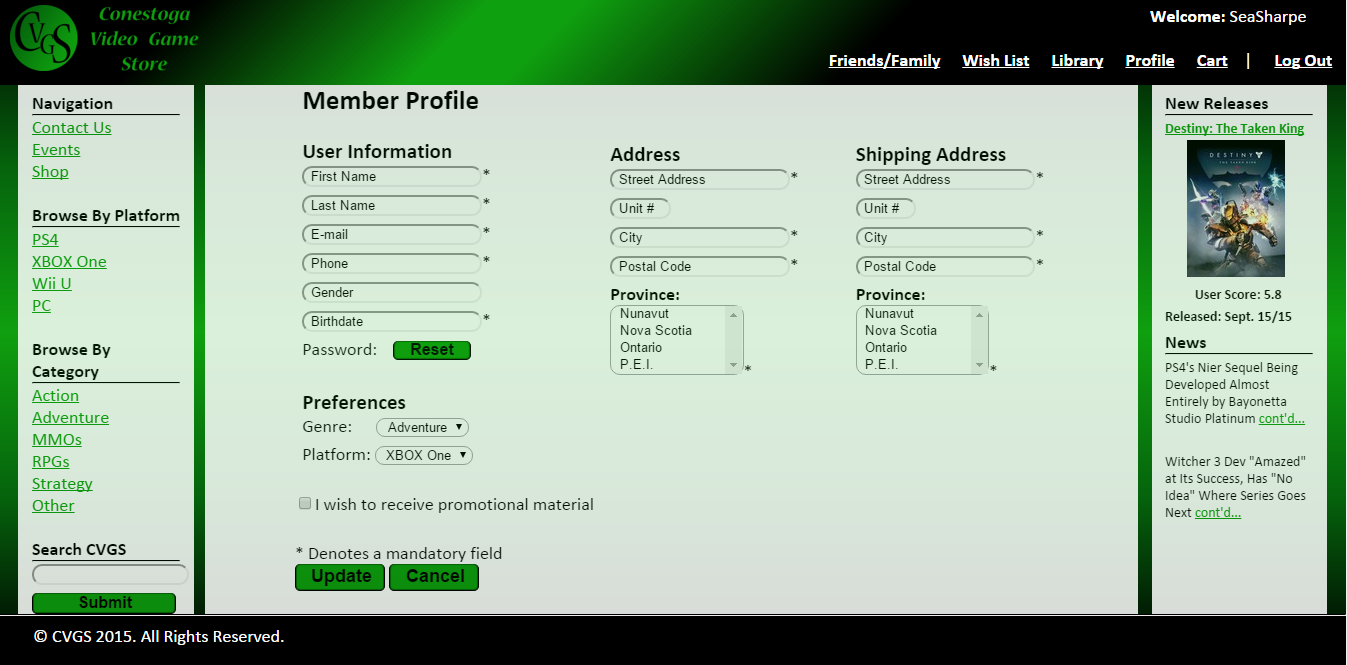
 

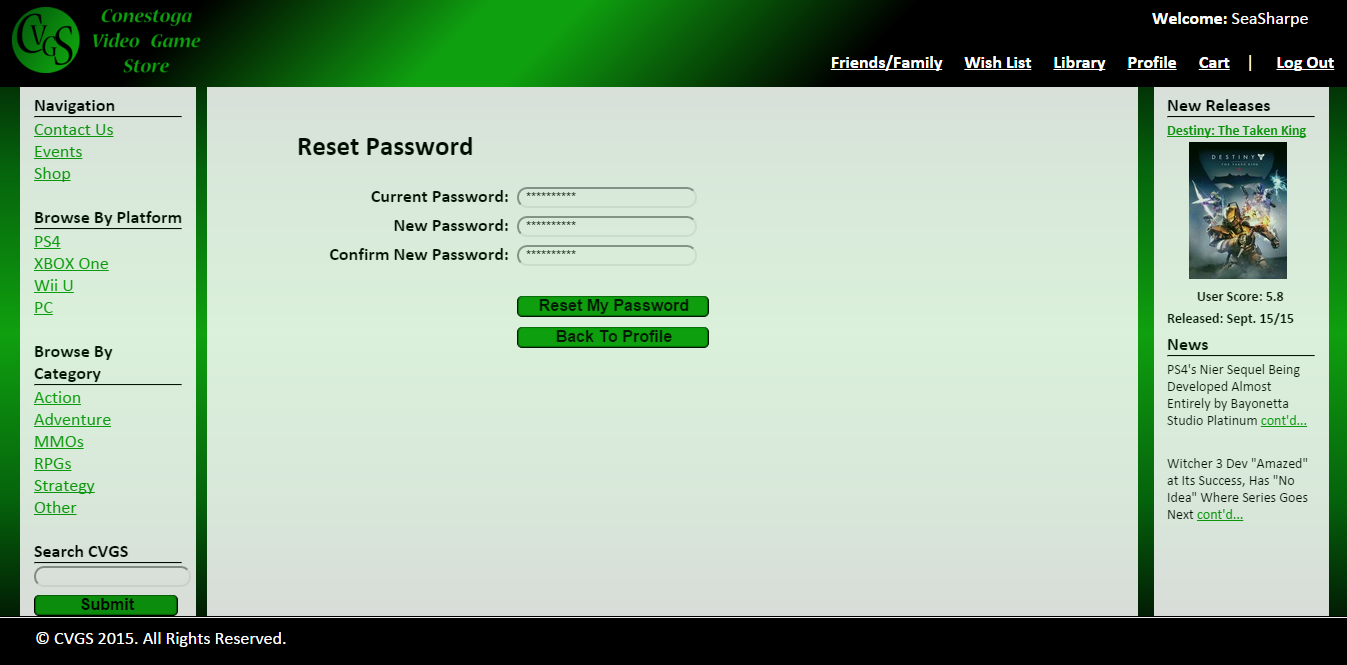
## Members

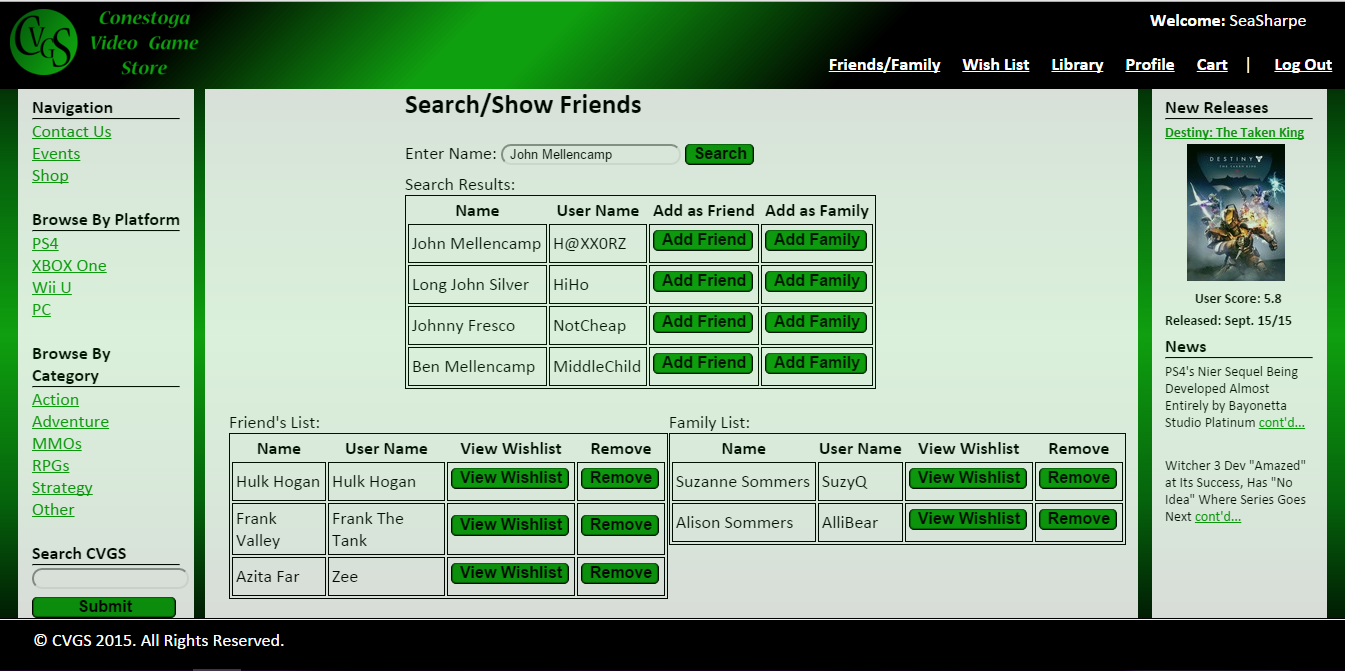






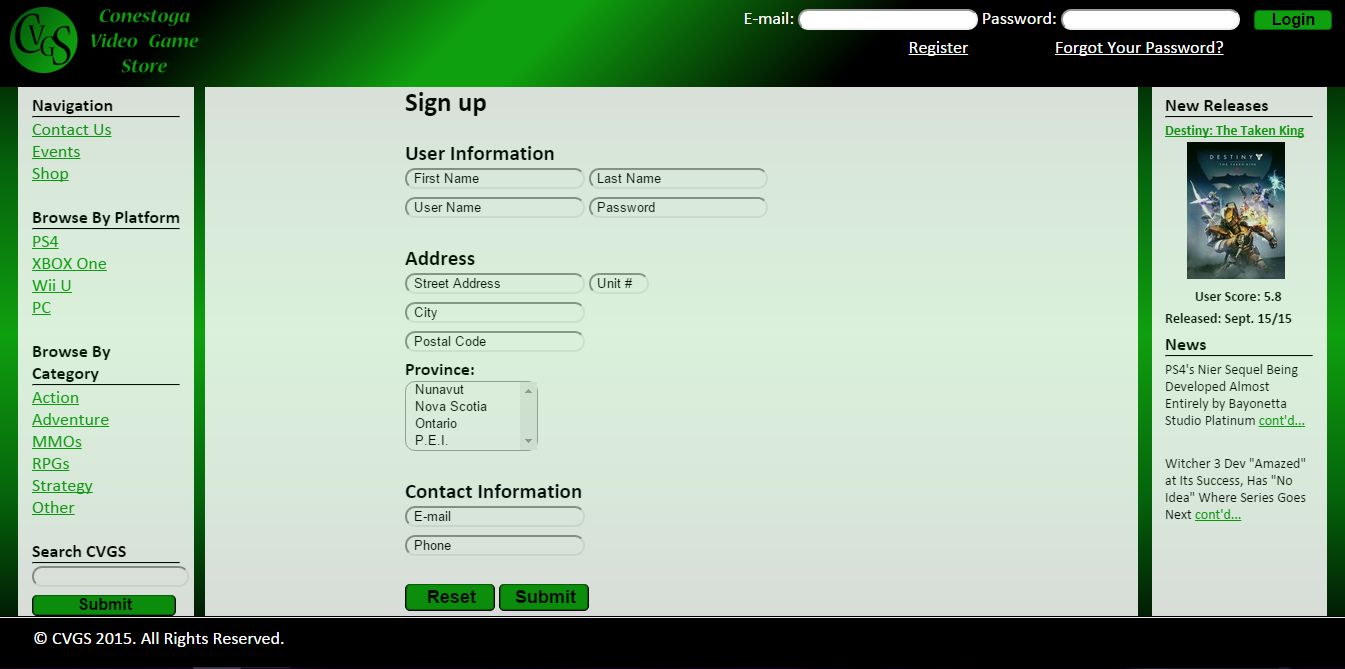




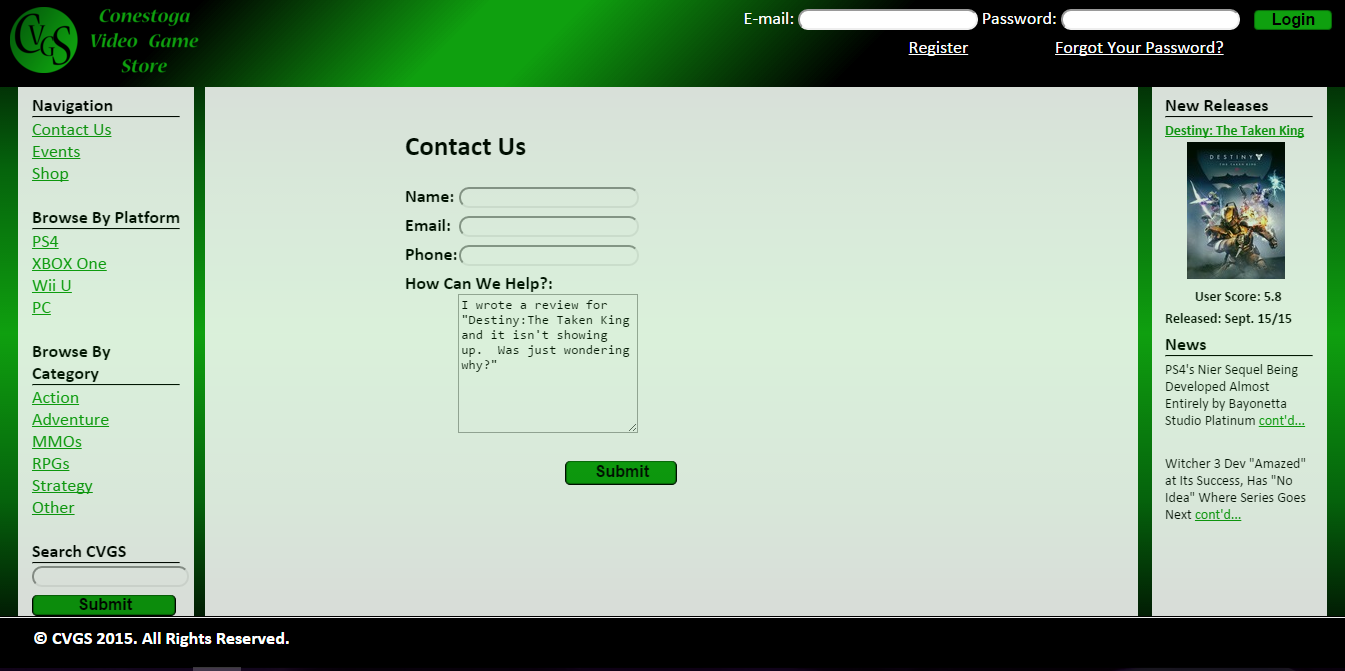


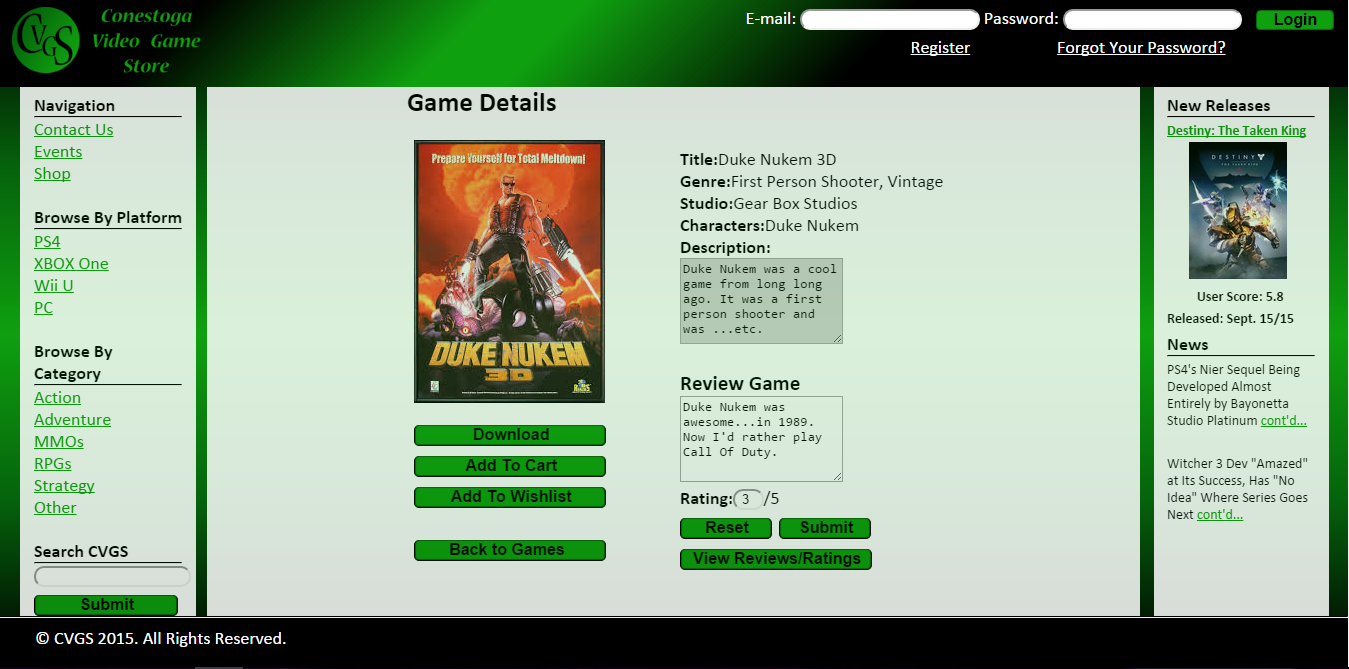


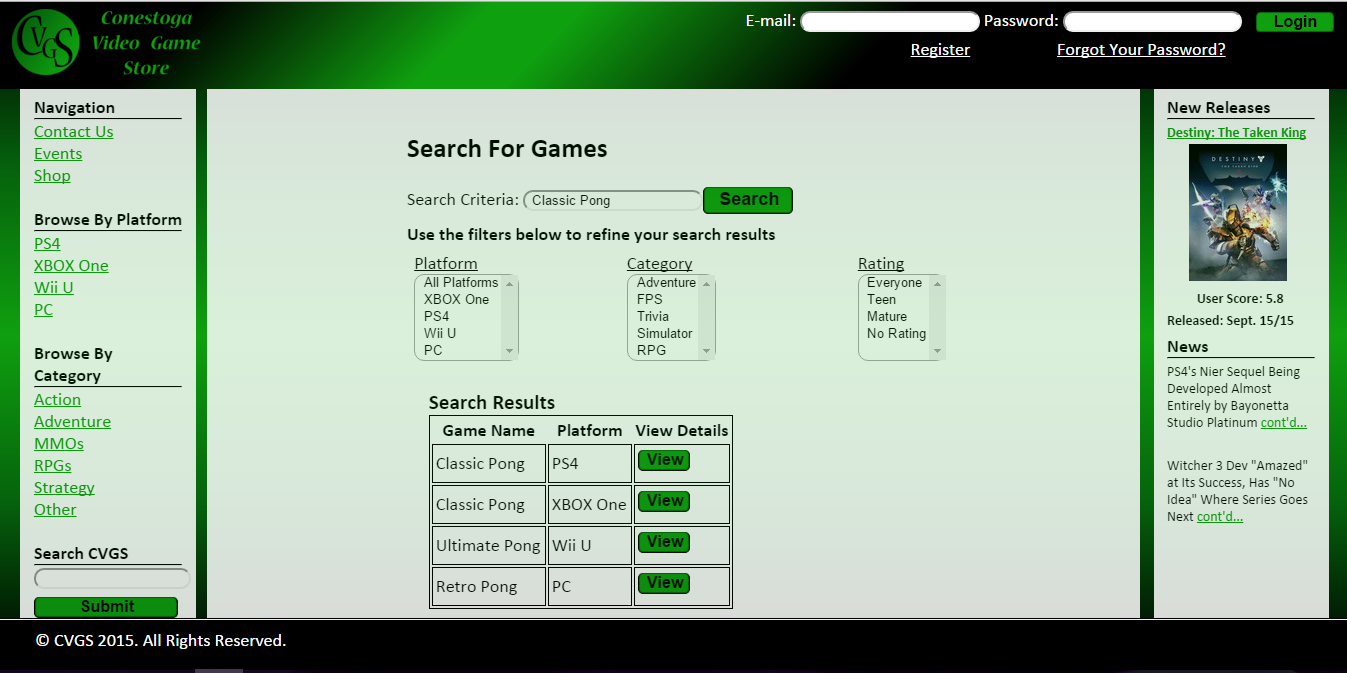
## Visitors

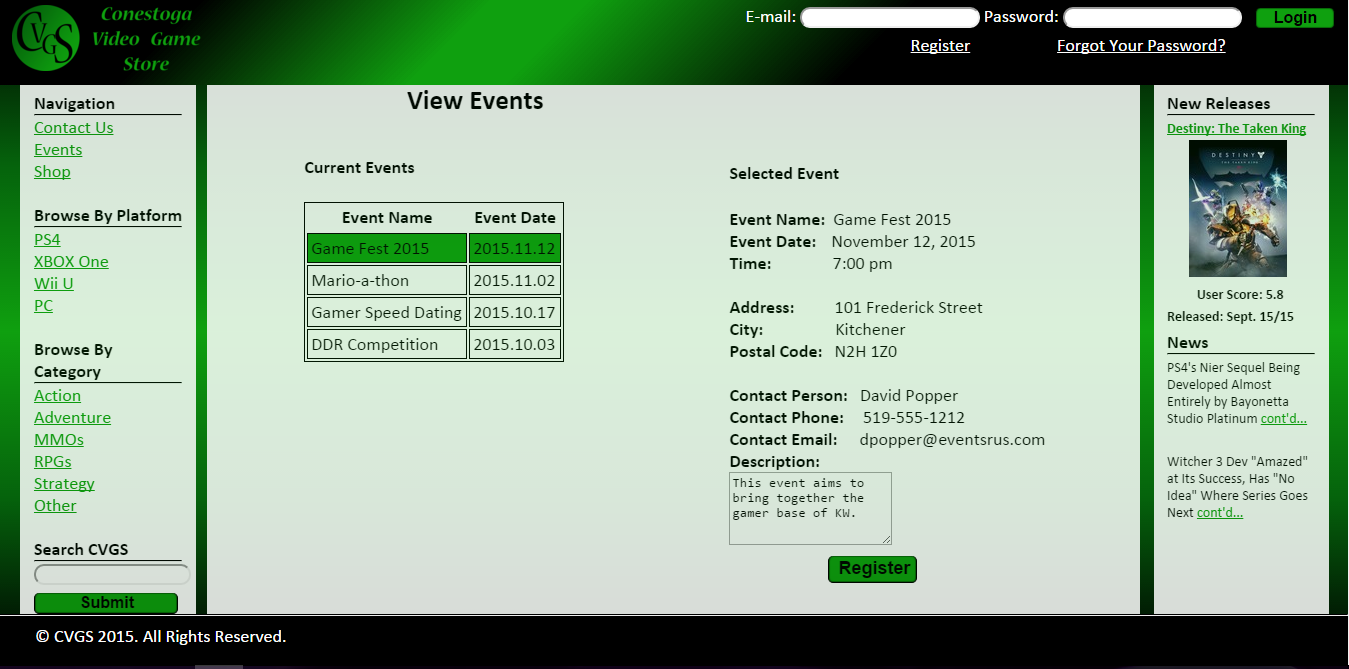


## Visitors & Members





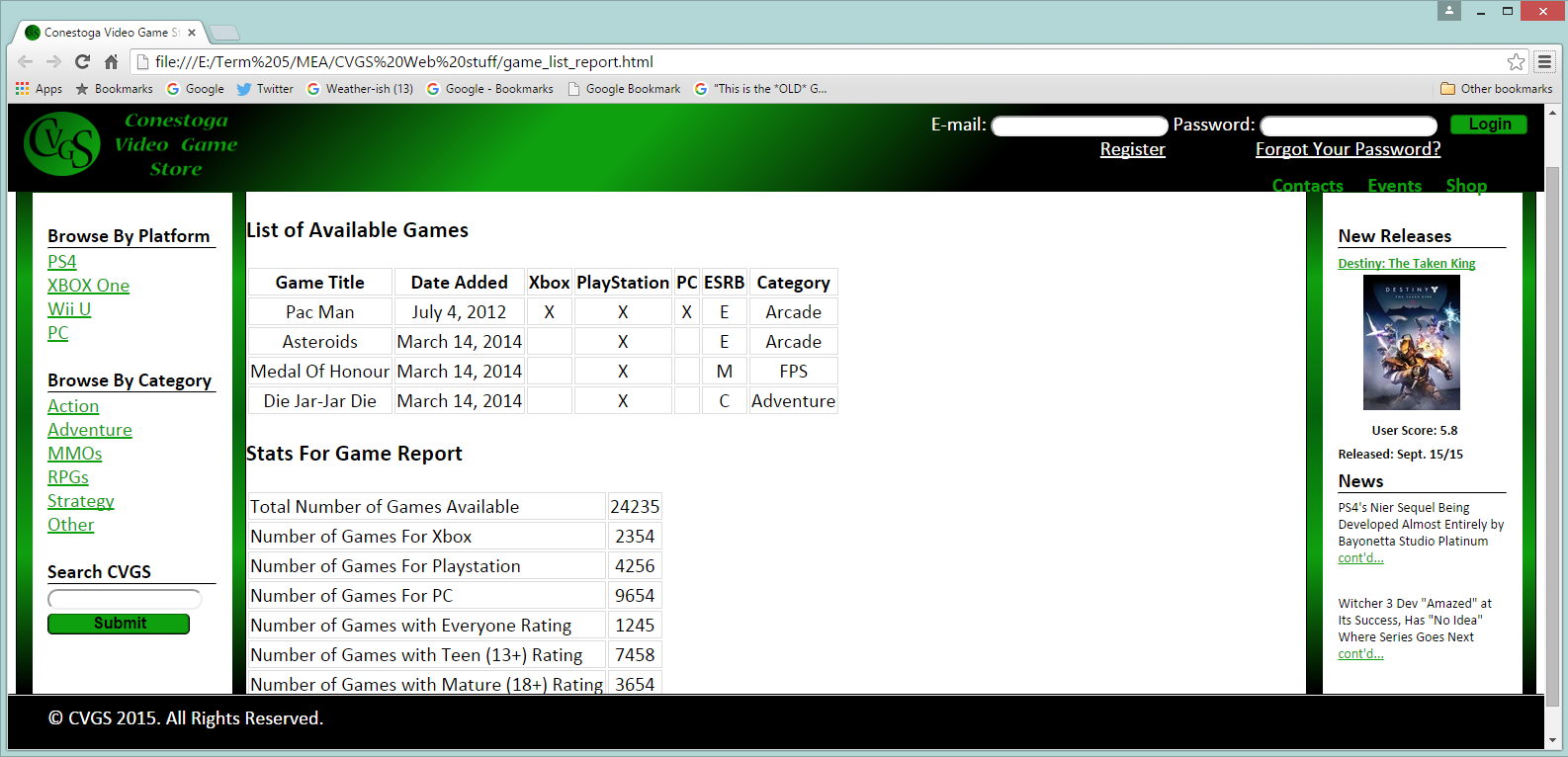




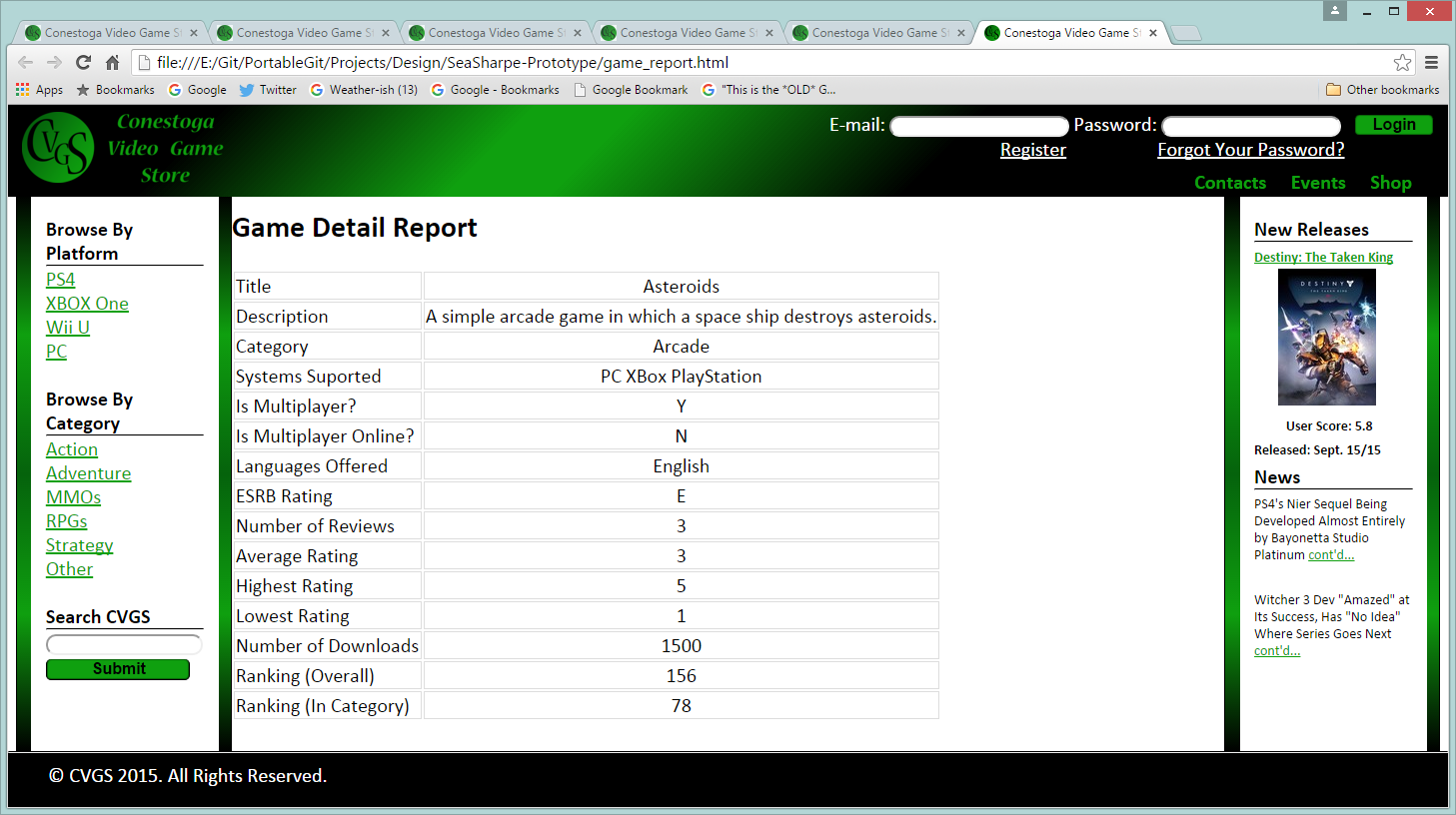
# Report Designs

[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/CVGS_Reports)

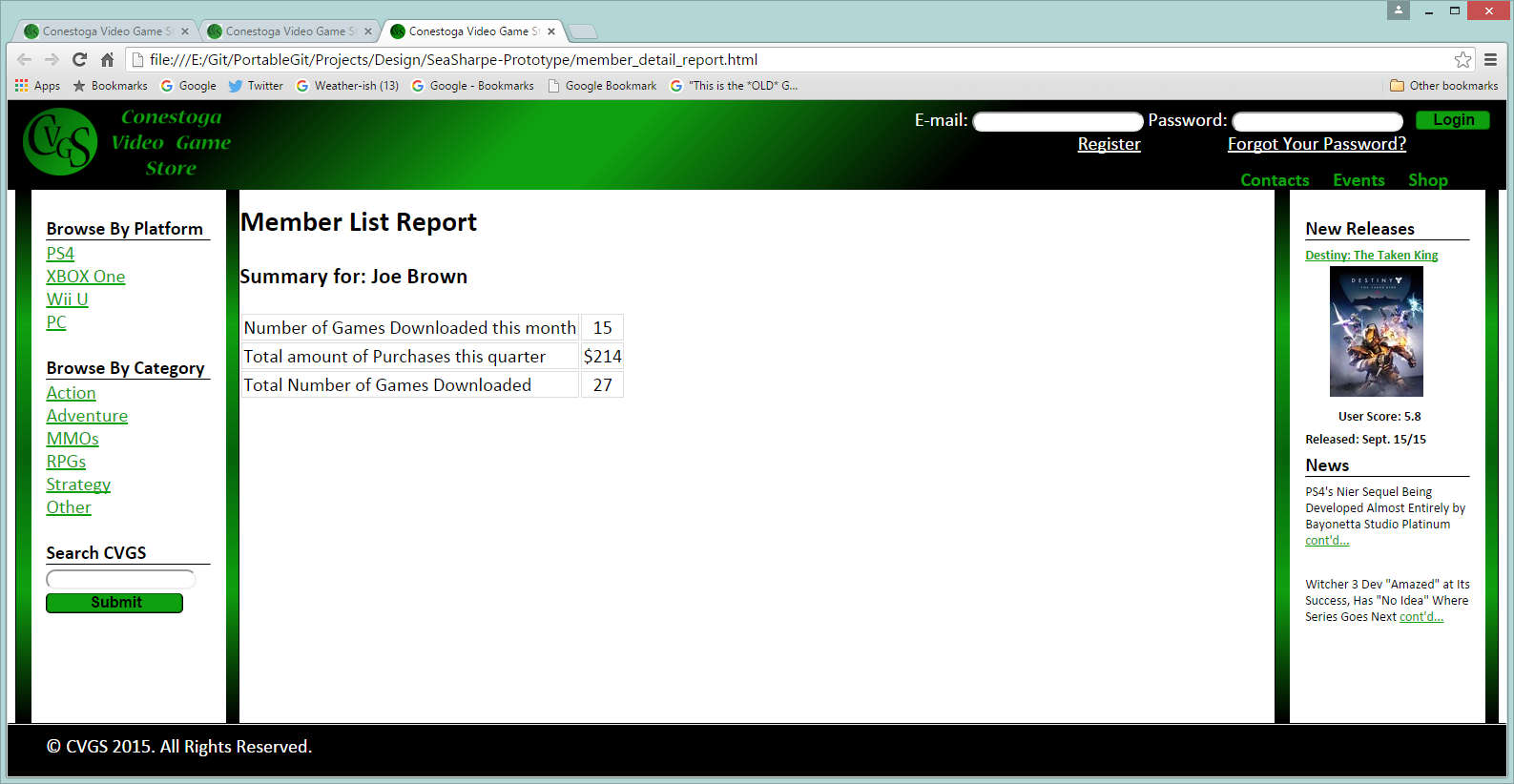
#### Game List Report



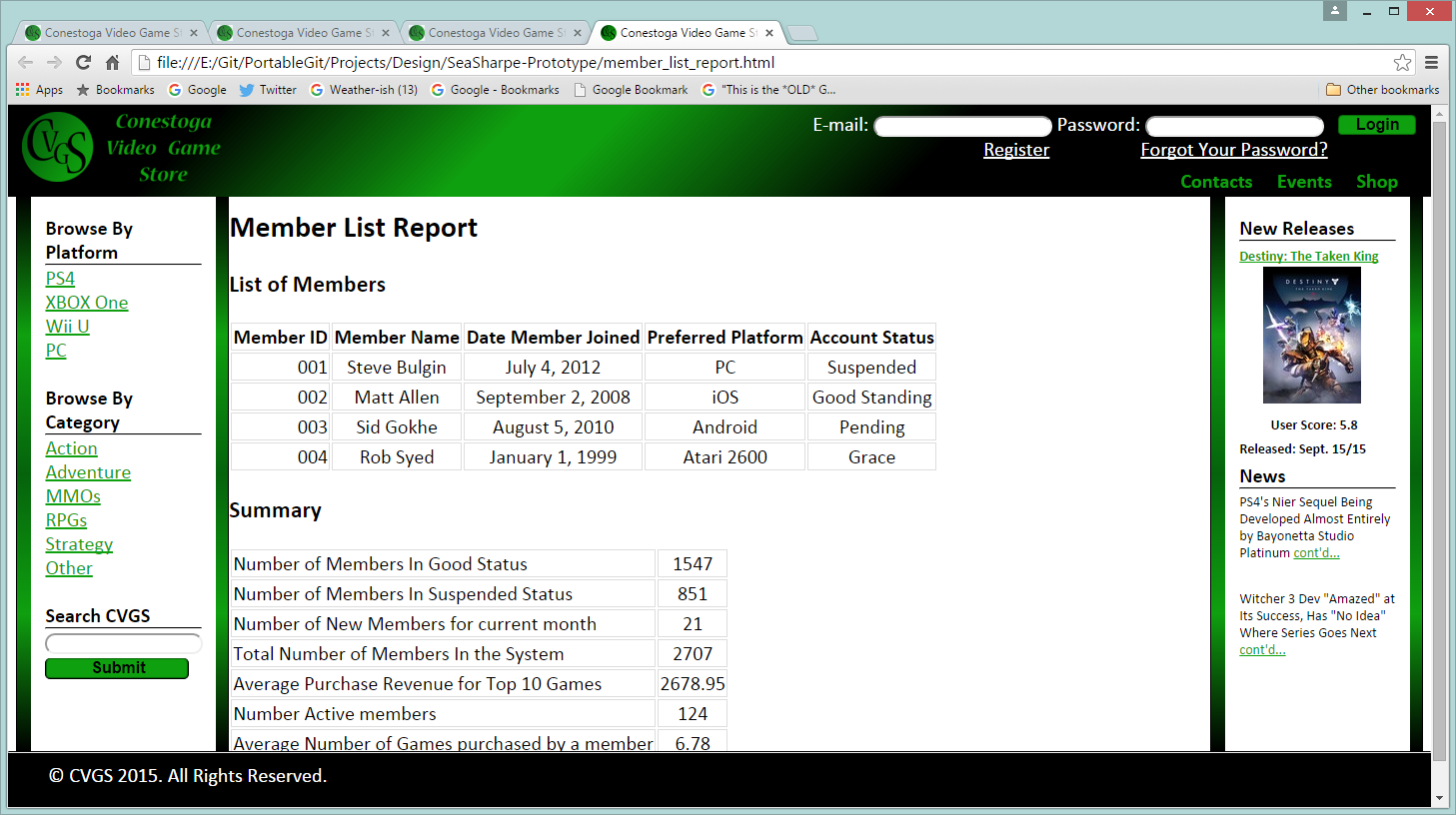
#### Game Report



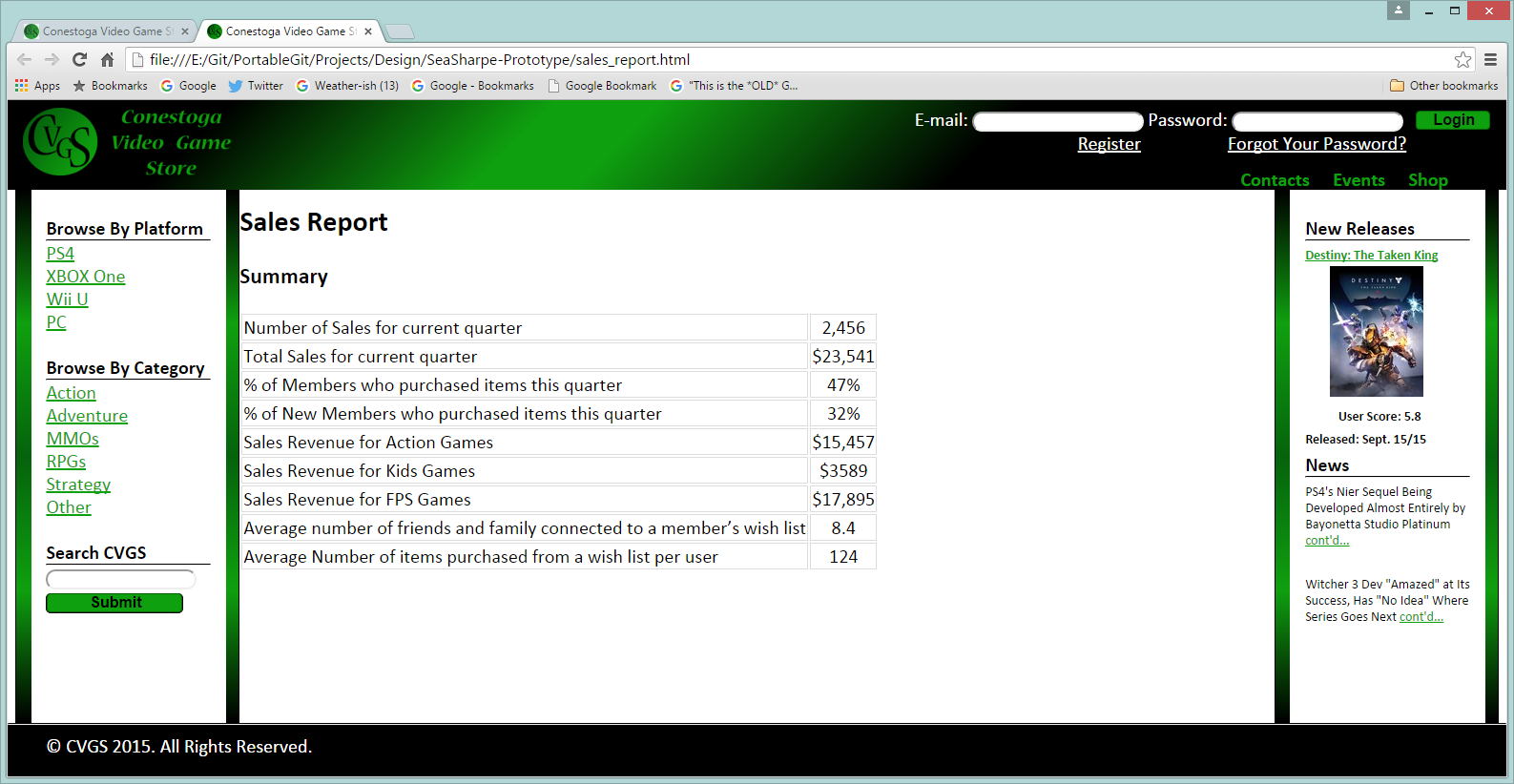
#### Member Detail Report



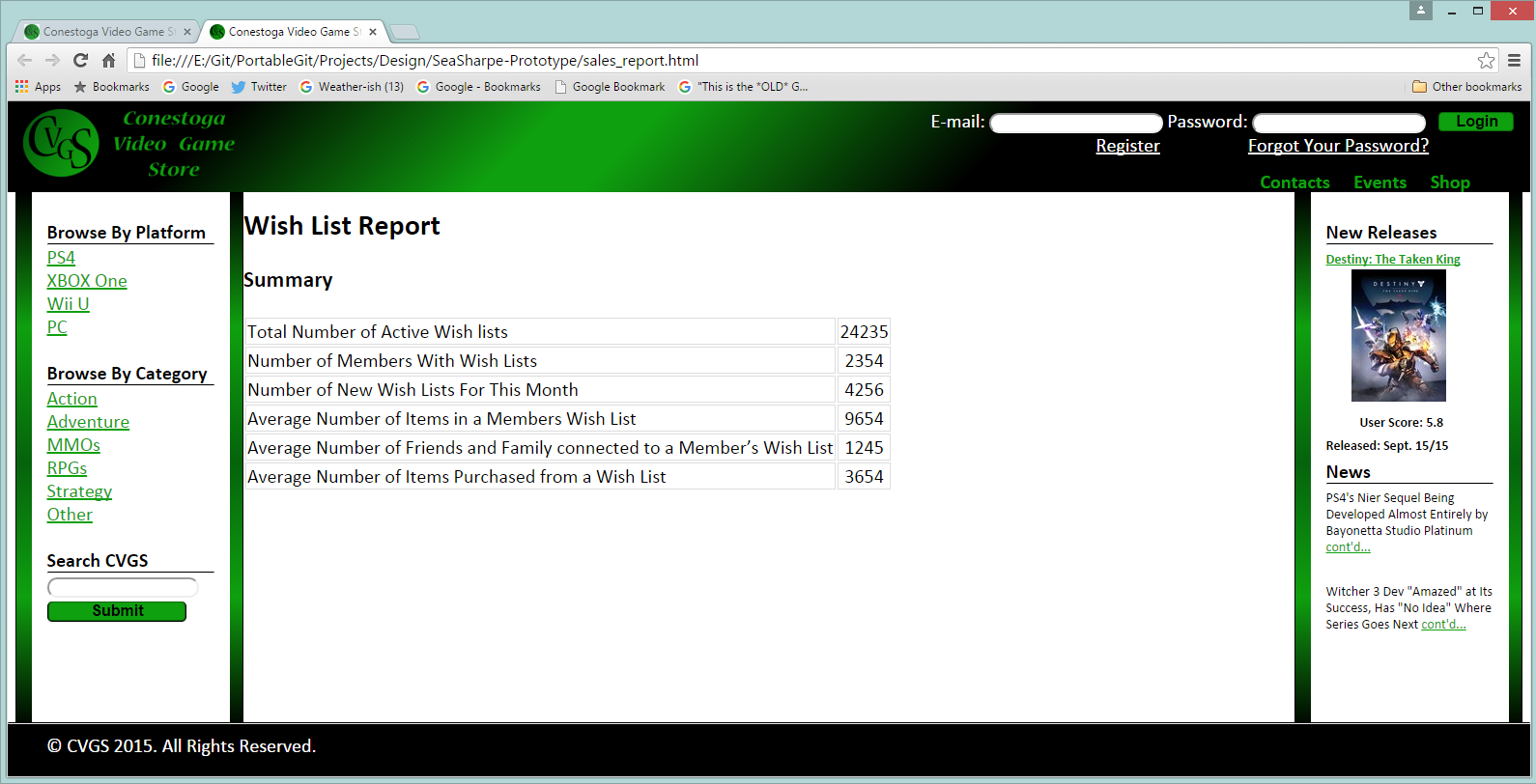
#### Member List Report



#### Sales Report

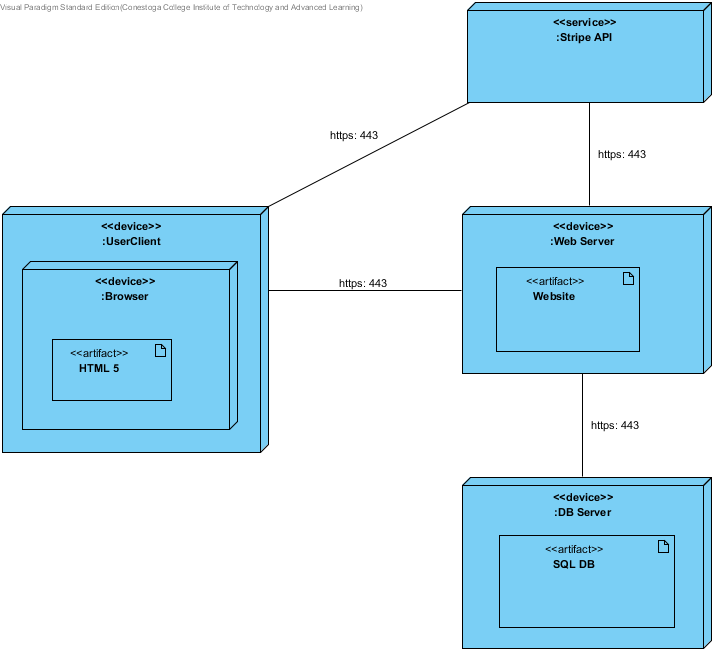


#### Wish List Report



# Deployment Diagram

[GitHub Link](https://github.com/SeaSharpe/Design/tree/master/Deployment%20Diagram)



# Security (Threat Modeling Report)

[GitHub Link](https://github.com/SeaSharpe/Design/blob/master/ThreatModel.md)

**Created on 2015-11-10 5:48:35 PM**

**Threat Model Name:** Conestoga Video Game Store Threat Model

**Owner:** Sea Sharpe

**Reviewer:** Dan Hofstede

**Contributors:** John Steel

**Description:** This service allows customers to purchase and review games, create wish lists and track game-related events.

**Assumptions:** We assume that it is in Stripe's interests to provide a stable, secure service with minimal downtime.

**External Dependencies:** Stripe's Credit Card Payment Service

#### Threat Model Summary:

|  |  |
| --- | --- |
| Not Started | 0 |
| Not Applicable | 8 |
| Needs Investigation | 0 |
| Mitigation Implemented | 37 |
| Total | 45 |
| Total Migrated | 0 |
|  |  |

#### Diagram: Diagram 1

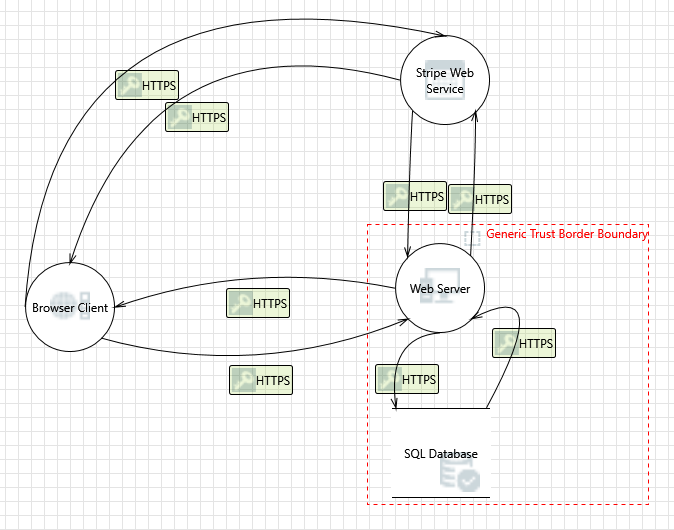
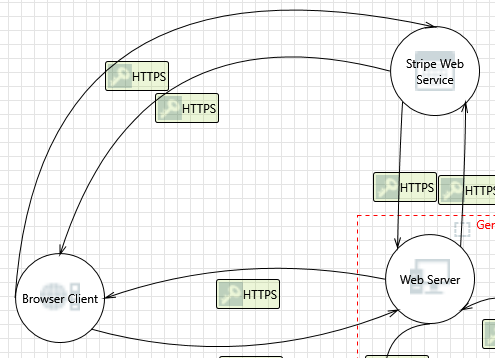


Diagram 1 diagram screenshot

#### Diagram 1 Diagram Summary:

|  |  |
| --- | --- |
| Not Started | 0 |
| Not Applicable | 8 |
| Needs Investigation | 0 |
| Mitigation Implemented | 37 |
| Total | 45 |
| Total Migrated | 0 |

#### Interaction: HTTPS



HTTPS interaction screenshot

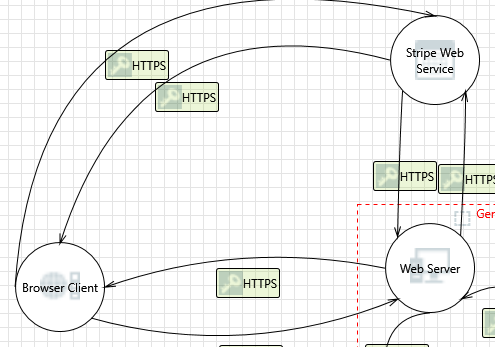
#### 1. Browser Client Process Memory Tampered  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | If Browser Client is given access to memory, such as shared memory or pointers, or is given the ability to control what Stripe Web Service executes (for example, passing back a function pointer.), then Browser Client can tamper with Stripe Web Service. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it. |
| **Justification:** | The Stripe web service is not under our control and the only information that could be compromised in such an event is the customer's information. If the customer's computer was already compromised their keystrokes may already have been logged. We cannot help the customer in such a situation. |

#### 2. Elevation Using Impersonation  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Stripe Web Service may be able to impersonate the context of Browser Client in order to gain additional privilege. |
| **Justification:** | Stripe would only be able to gain the privilege of an account that it has the credentials for. In our scenario, stripe already stores the most crucial information for our users, their credit card information. |

#### Interaction: HTTPS



HTTPS interaction screenshot

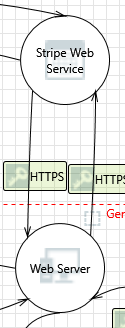
#### 3. Stripe Web Service Process Memory Tampered  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | If Stripe Web Service is given access to memory, such as shared memory or pointers, or is given the ability to control what Browser Client executes (for example, passing back a function pointer.), then Stripe Web Service can tamper with Browser Client. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it. |
| **Justification:** | We have to trust Stripe not to do this. |

#### 4. Elevation Using Impersonation  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Browser Client may be able to impersonate the context of Stripe Web Service in order to gain additional privilege. |
| **Justification:** | Client cannot impersonate stripe without knowledge of their Certificate Private Key. We trust that stripe's private key stays private. Should it be stolen we trust that stripe would revoke their certificate to prevent this kind of attack. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 5. Spoofing the Stripe Web Service Process  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Stripe Web Service may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the source process. |
| **Justification:** | Stripe is identified by it's HTTPS certificate. |

#### 6. Cross Site Scripting  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitization is handled by the Razor 5 and Entity frameworks, even so, tests will be done to ensure this. See Github Implementation Issue #1. |

#### 7. Potential Data Repudiation by Web Server  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Web Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | IIS Access logs will configured to include this information. |

#### 8. Potential Process Crash or Stop for Web Server  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Web Server crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | Stripe does not rely on our web server to operate. |

#### 9. Data Flow HTTPS Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | If an ISP, government or other entity disrupts traffic between our server and clients or stripe we would no longer be able to process sales. We would take all appropriate steps to ensure that this does not occur. |

#### 10. Elevation Using Impersonation  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Stripe Web Service in order to gain additional privilege. |
| **Justification:** | We use stripe so that we don't have to store credit card information on our systems, but the web server could pretend to be stripe (or just be patched to work without stripe) and customer credit card information could be captured. This could do significant harm to our brand if it was discovered. Precautions will be taken to ensure that this does not happen. |

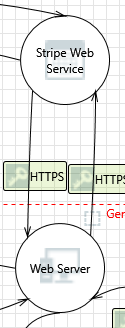
#### 11. Web Server May be Subject to Elevation of Privilege Using Remote Code Execution  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Stripe Web Service may be able to remotely execute code for Web Server. |
| **Justification:** | We will sanitize inputs from Stripe and throw exceptions/log errors when unexpected values are received. |

#### 12. Elevation by Changing the Execution Flow in Web Server  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Web Server in order to change the flow of program execution within Web Server to the attacker's choosing. |
| **Justification:** | User inputs will inevitably effect the flow of our server software but we will ensure that there are not any implementation bugs where a user is able to take actions that they are not authorized to do. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 13. Spoofing the Web Server Process  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Web Server may be spoofed by an attacker and this may lead to unauthorized access to Stripe Web Service. Consider using a standard authentication mechanism to identify the source process. |
| **Justification:** | Stripe provides credentials for the server, as long as those credentials are not leaked, the web service cannot be spoofed in this manner. |

#### 14. Web Server Process Memory Tampered  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | If Web Server is given access to memory, such as shared memory or pointers, or is given the ability to control what Stripe Web Service executes (for example, passing back a function pointer.), then Web Server can tamper with Stripe Web Service. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it. |
| **Justification:** | Web Server is on a private server and does not share memory. |

#### 15. Potential Data Repudiation by Stripe Web Service  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Stripe Web Service claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | Out of Scope |

#### 16. Potential Process Crash or Stop for Stripe Web Service  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Stripe Web Service crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | If the Stripe service has issues, we will no longer be able to complete sales; however, we can still function as a social hub for game & event information. This issue could cost us revenue. We trust that to protect their brand name and minimize financial loss, Stripe will strive to keep their downtime to a minimum. |

#### 17. Data Flow HTTPS Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** I | f an ISP, government or other entity disrupts traffic between our server and clients or stripe we would no longer be able to process sales. We would take all appropriate steps to ensure that this does not occur. |

#### 18. Elevation Using Impersonation  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Stripe Web Service may be able to impersonate the context of Web Server in order to gain additional privilege. |
| **Justification:** | The calls to and callbacks from stripe are too specific and should not lend themselves to elevation of privilege bugs |

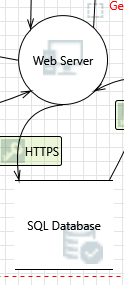
#### 19. Stripe Web Service May be Subject to Elevation of Privilege Using Remote Code Execution  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to remotely execute code for Stripe Web Service. |
| **Justification:** | Attacks against stripe from our server are out of scope. |

#### 20. Elevation by Changing the Execution Flow in Stripe Web Service  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Stripe Web Service in order to change the flow of program execution within Stripe Web Service to the attacker's choosing. |
| **Justification:** | Attacks against stripe from our server are out of scope. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 21. Spoofing of Destination Data Store SQL Database  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SQL Database may be spoofed by an attacker and this may lead to data being written to the attacker's target instead of SQL Database. Consider using a standard authentication mechanism to identify the destination data store. |
| **Justification:** | SQL Database will be authenticated using mutual SSL Authentication. So long as its private key stays private, they will be authentic. |

#### 22. Potential SQL Injection Vulnerability for SQL Database  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | SQL injection is an attack in which malicious code is inserted into strings that are later passed to an instance of SQL Server for parsing and execution. Any procedure that constructs SQL statements should be reviewed for injection vulnerabilities because SQL Server will execute all syntactically valid queries that it receives. Even parameterized data can be manipulated by a skilled and determined attacker. |
| **Justification:** | Sanitization is handled by the Razor 5 and Entity frameworks, even so, tests will be done to ensure this. See Github Implementation Issue #1. |

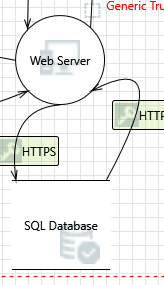
#### 23. Weak Credential Storage  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Credentials held at the server are often disclosed or tampered with and credentials stored on the client are often stolen. For server side, consider storing a salted hash of the credentials instead of storing the credentials themselves. If this is not possible due to business requirements, be sure to encrypt the credentials before storage, using an SDL-approved mechanism. For client side, if storing credentials is required, encrypt them and protect the data store in which they're stored |
| **Justification:** | We implement hashing and salting of user passwords. Actual passwords are never stored in logs or the db and are only in memory for as long as is required to hash them. |

#### 24. Potential Excessive Resource Consumption for Web Server or SQL Database  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Does Web Server or SQL Database take explicit steps to control resource consumption? Resource consumption attacks can be hard to deal with, and there are times that it makes sense to let the OS do the job. Be careful that your resource requests don't deadlock, and that they do timeout. |
| **Justification:** | Auditing should be done on the most time consuming database tasks to find out if they are malicious. If so, steps should be taken to automatically ban or block users from creating malicious requests aimed at tying up resources. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 25. Spoofing of Source Data Store SQL Database  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | SQL Database may be spoofed by an attacker and this may lead to incorrect data delivered to Web Server. Consider using a standard authentication mechanism to identify the source data store. |
| **Justification:** | SQL Database will be authenticated using mutual SSL Authentication. So long as its private key remains private it can be assumed as authentic. |

#### 26. Cross Site Scripting  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitization is handled by the Razor 5 and Entity frameworks, even so, tests will be done to ensure this. See Github Implementation Issue #1. |

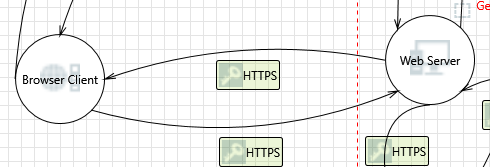
#### 27. Persistent Cross Site Scripting  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a persistent cross-site scripting attack because it does not sanitize data store 'SQL Database' inputs and output. |
| **Justification:** | Sanitization is handled by the Razor 5 and Entity frameworks, even so, tests will be done to ensure this. See Github Implementation Issue #1. |

#### 28. Weak Access Control for a Resource  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Information Disclosure |
| **Description:** | Improper data protection of SQL Database can allow an attacker to read information not intended for disclosure. Review authorization settings. |
| **Justification:** | Extra care will be taken to ensure that requests are authorized. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 29. Spoofing the Browser Client Process  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Browser Client may be spoofed by an attacker and this may lead to unauthorized access to Web Server. Consider using a standard authentication mechanism to identify the source process. |
| **Justification:** | A token stored in an HTTP cookie will be used to authenticate the client. Cookies will timeout periodically to mitigate the risk of being stolen. |

#### 30. Cross Site Scripting  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | The web server 'Web Server' could be a subject to a cross-site scripting attack because it does not sanitize untrusted input. |
| **Justification:** | Sanitization is handled by the Razor 5 and Entity frameworks, even so, tests will be done to ensure this. See Github Implementation Issue #1. |

#### 31. Potential Data Repudiation by Web Server  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Web Server claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | IIS Access logs will configured to include this information. |

#### 32. Potential Process Crash or Stop for Web Server  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Web Server crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | Such problems would cost us business. We will ensure that our platform is as stable as possible. |

#### 33. Data Flow HTTPS Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | If an ISP, government or other entity disrupts traffic between our server and clients or stripe we would no longer be able to process sales. We would take all appropriate steps to ensure that this does not occur. |

#### 34. Elevation Using Impersonation  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to impersonate the context of Browser Client in order to gain additional privilege. |
| **Justification:** | This does not apply in our case. |

#### 35. Web Server May be Subject to Elevation of Privilege Using Remote Code Execution  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Browser Client may be able to remotely execute code for Web Server. |
| **Justification:** | Razor 5 should handle sanitization of all form inputs but testing should be done to verify that user inputs can never allow the user to compromise our web server. |

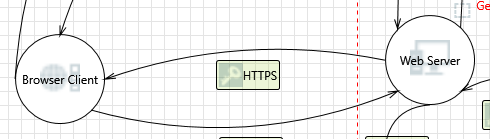
#### 36. Elevation by Changing the Execution Flow in Web Server  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Web Server in order to change the flow of program execution within Web Server to the attacker's choosing. |
| **Justification:** | User inputs will inevitably effect the flow of our server software but we will take care to ensure that there are not any implementation bugs where a user is able to take actions that they are not authorized to do. |

#### 37. Cross Site Request Forgery  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Cross-site request forgery (CSRF or XSRF) is a type of attack in which an attacker forces a user's browser to make a forged request to a vulnerable site by exploiting an existing trust relationship between the browser and the vulnerable web site. In a simple scenario, a user is logged in to web site A using a cookie as a credential. The other browses to web site B. Web site B returns a page with a hidden form that posts to web site A. Since the browser will carry the user's cookie to web site A, web site B now can take any action on web site A, for example, adding an admin to an account. The attack can be used to exploit any requests that the browser automatically authenticates, e.g. by session cookie, integrated authentication, IP whitelisting, … The attack can be carried out in many ways such as by luring the victim to a site under control of the attacker, getting the user to click a link in a phishing email, or hacking a reputable web site that the victim will visit. The issue can only be resolved on the server side by requiring that all authenticated state-changing requests include an additional piece of secret payload (canary or CSRF token) which is known only to the legitimate web site and the browser and which is protected in transit through SSL/TLS. See the Forgery Protection property on the flow stencil for a list of mitigations. |
| **Justification:** | Razor 5 has mitigated this with Anti-Forgery tokens. These can make testing with Curl / Postman difficult and may be turned off during development. We must ensure that these tokens are enabled in production. |

#### Interaction: HTTPS



HTTPS interaction screenshot

#### 38. Spoofing the Web Server Process  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Spoofing |
| **Description:** | Web Server may be spoofed by an attacker and this may lead to unauthorized access to Browser Client. Consider using a standard authentication mechanism to identify the source process. |
| **Justification:** | The web server is authenticated by the HTTPS Certificate and our private key. |

#### 39. Web Server Process Memory Tampered  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Tampering |
| **Description:** | If Web Server is given access to memory, such as shared memory or pointers, or is given the ability to control what Browser Client executes (for example, passing back a function pointer.), then Web Server can tamper with Browser Client. Consider if the function could work with less access to memory, such as passing data rather than pointers. Copy in data provided, and then validate it. |
| **Justification:** | Web Server is on a private server and does not share memory. |

#### 40. Potential Data Repudiation by Browser Client  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Repudiation |
| **Description:** | Browser Client claims that it did not receive data from a source outside the trust boundary. Consider using logging or auditing to record the source, time, and summary of the received data. |
| **Justification:** | IIS Access logs will configured to include this information. |

#### 41. Potential Process Crash or Stop for Browser Client  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | Browser Client crashes, halts, stops or runs slowly; in all cases violating an availability metric. |
| **Justification:** | We will do testing to ensure that our site does not cause browsers to become unstable. |

#### 42. Data Flow HTTPS Is Potentially Interrupted  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Denial Of Service |
| **Description:** | An external agent interrupts data flowing across a trust boundary in either direction. |
| **Justification:** | If an ISP, government or other entity disrupts traffic between our server and clients or stripe we would no longer be able to process sales. We would take all appropriate steps to ensure that this does not occur. |

#### 43. Elevation Using Impersonation  [State: Not Applicable]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Browser Client may be able to impersonate the context of Web Server in order to gain additional privilege. |
| **Justification:** | This does not apply in our case. |

#### 44. Browser Client May be Subject to Elevation of Privilege Using Remote Code Execution  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | Web Server may be able to remotely execute code for Browser Client. |
| **Justification:** | It certainly could (and does) execute javascript code on the client that support it. Existing XSS Mitigations will ensure that only authorized code is sent to browsers. |

#### 45. Elevation by Changing the Execution Flow in Browser Client  [State: Mitigation Implemented]  [Priority: High]

|  |  |
| --- | --- |
| **Category:** | Elevation Of Privilege |
| **Description:** | An attacker may pass data into Browser Client in order to change the flow of program execution within Browser Client to the attacker's choosing. |
| **Justification:** | It certainly could (and does) execute javascript code on the client that support it. Existing XSS Mitigations will ensure that only authorized code is sent to browsers. |