[The coalition] Web App

ANALYSIS AND DESIGN DOCUMENT



http://cdn.segmentnext.com/wp-content/uploads/2016/03/Gears-of-War-4-characters.png

**Team 2**

**Erick Bravo**

**Daniel Oluwadare**

**Kent Lopez**

# Contents

Contents 2

Introduction 6

System Architecture 7

System Overview 7

Architectural Design 7

Decomposition Description 8

Design Rationale 9

Human Interface Design 10

registration 10

INPUT & OUTPUT 10

Actions 11

Pre and Post Conditions 11

Validation 11

LOGIN 12

INPUT & OUTPUT 12

Actions 12

Pre and Post Conditions 13

Validation 13

CHANGE PASSWORD 13

INPUT & OUTPUT 13

Actions 14

Pre and Post Conditions 14

Validation 14

List of games of selected console 14

INPUT & OUTPUT 15

Actions 15

Pre and Post Conditions 15

Validation 16

Search for a game 16

INPUT & OUTPUT 16

Actions 16

Pre and Post Conditions 16

Validation 16

Add a game 17

INPUT & OUTPUT 17

Actions 17

Pre and Post Conditions 18

Validation 18

Create walkthrough 18

INPUT & OUTPUT 19

Actions 19

Pre and Post Conditions 19

Validation 19

Edit Walkthrough 19

INPUT & OUTPUT 20

Actions 21

Pre and Post Conditions 21

Validation 21

View walkthrough 21

INPUT & OUTPUT 22

Actions 23

Pre and Post Conditions 23

Validation 23

Forum Login 23

INPUT & OUTPUT 23

Actions 24

Pre and Post Conditions 24

Validation 24

Create post 24

INPUT & OUTPUT 25

Actions 25

Pre and Post Conditions 26

Validation 26

Edit post 26

INPUT & OUTPUT 27

Actions 27

Pre and Post Conditions 27

Validation 27

Create Group 27

INPUT & OUTPUT 28

Actions 29

Pre and Post Conditions 29

Validation 29

Component Design 29

Registeration Entities 30

Game Catalog Entities 31

Walkthrough Entities 33

Forums Entities 34

Data Design 35

Data Description 35

Registration / Authentication 35

Game catalog 36

Walkthrough 37

Forums 38

DATABASE Dictionary / SCHEMA 39

Registration / Authentication 39

Game catalog 40

Walkthrough 41

Forums 42

Full DAtabase Model 43

Requirements Matrix 44

Appendices 45

# Introduction

Analysis and Design Document (ADD) is a document that minimizes both time and cost, that Engineers will use to develop an application or system. The purpose of an ADD is to outline how things in the project are supposed to work and why they work that way. Knowing this, our ADD will cover how our application will work and how it will be designed. In this document we will divide our topics in in 5 sections: purpose, scope, Definitions and acronyms, References, and overview.

**Purpose**: There are copious amounts of game websites out there. The problem with majority of these sites is that they are convoluted and lack structure of an enticing hub for new gamers. These sites put too much emphasis towards presentation for experienced gamers and not enough on new gamers. The purpose of our site is to present a game hub that is welcoming to both old and new gamers. We will:

* Have a hub specifically designed for new comers
* Have a suggestion box where gamers can suggest new games
* List games by console, rather than listing them all at once

**Scope**: This software will be produced by HTML, CSS JavaScript on the client side portion. On the server side, we will have PHP communicating with MySQL database system. Our software product will have a forum where new comers can ask questions. Users will also be able to ask administrators questions. Users will be able to suggest games and console for the website. We will seek outside help to integrate a cloud server for data overflow. To allow for maximum data flow and security to our servers and users. The document will consist of a breakdown of the following modules to be implemented

* Have a Game Catalog that will manage all content of games and how each are presented and shown to the user. (Daniel Oluwadare)
* Have a Walkthrough/User guide that will display guide walkthroughs for specific games and allow for user input. (Kent Lopez)
* Have a Forums/Discussions Board that will display threads showing discussions for specific games or video game related topics. (Erick Bravo)

**Definitions, acronyms, and abbreviations:**

UML – Unified Modeling Language

Admin- Administrator

UI – User Interface

OS- Operating Systems

SRS- Software Requirements & Specification

ADD- Analysis and Design Document

**References**

None.

**Overview**

The Goal of the ADD is to explain in great detail what we want to accomplish as a gaming websites and this is done through visual diagrams and detail breakdowns of what to expect. The remainder of this document will show our overall intention and then go into further detail how that will be implemented and finally showing you our final product in the end. You will see a breakdown of each function as well as what each function consist of.

# System Architecture

## System Overview

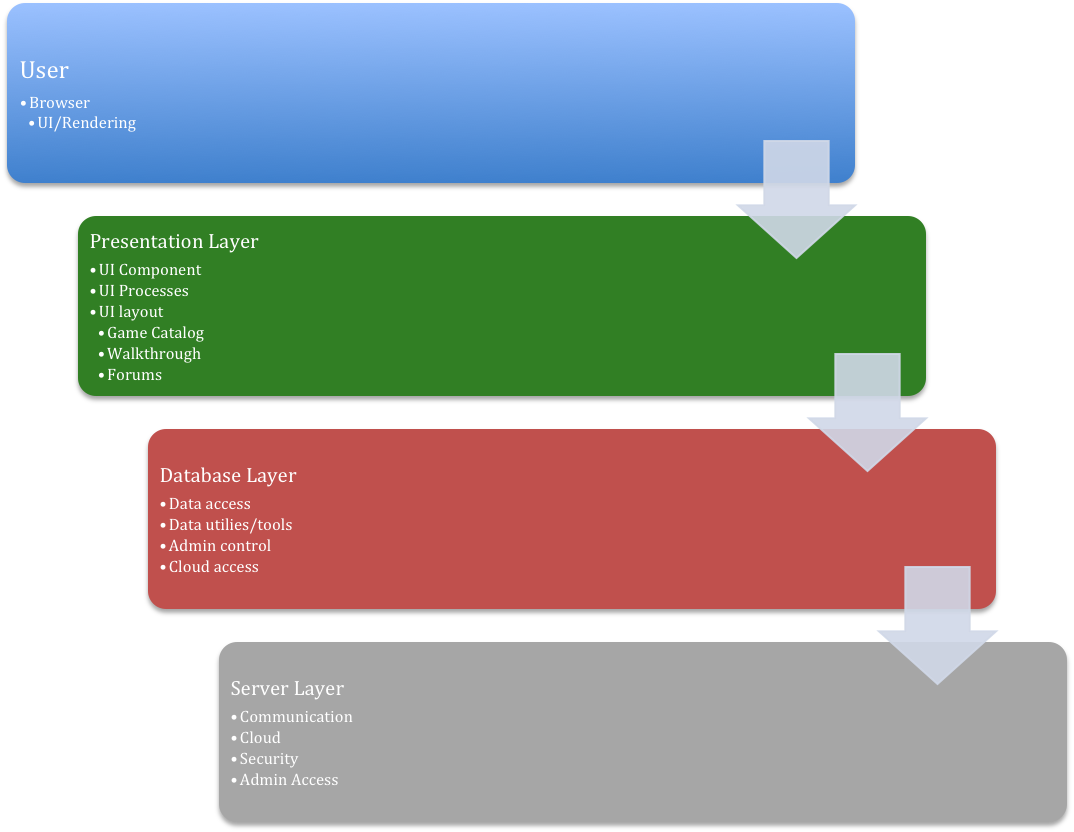
The Goal for this project is to create a gaming website where a community of gamers can come and share their experiences as well as come together on a website to stay up-to-date on the gaming industry. The website will be design to have 2 dedicated parts. One of the user heart felt design of the gaming community in the video game industry. While the other half will be dedicated to meeting the needs and requirement of the administrator, who will manage all content on website.

The websites will entail three main focal points and offer dedicated services to each. The first of the main features will be the Game catalog, which will encompass; news, reviews, ratings and more. Other features to be included will be a discussion/forums platform, where the gaming community can come together for help one another or be part of the discussion of a game. Lastly we will include section for admin and user based walkthrough for most currently released games.

The Goal for this project is to create a more unified gaming community where content is display in an more engaging manor to the user and allows for more user content driven website. This will solve the issue of having to reaching out to other platforms to complete one’s experiences that will be found all in one on this website with user driven content.

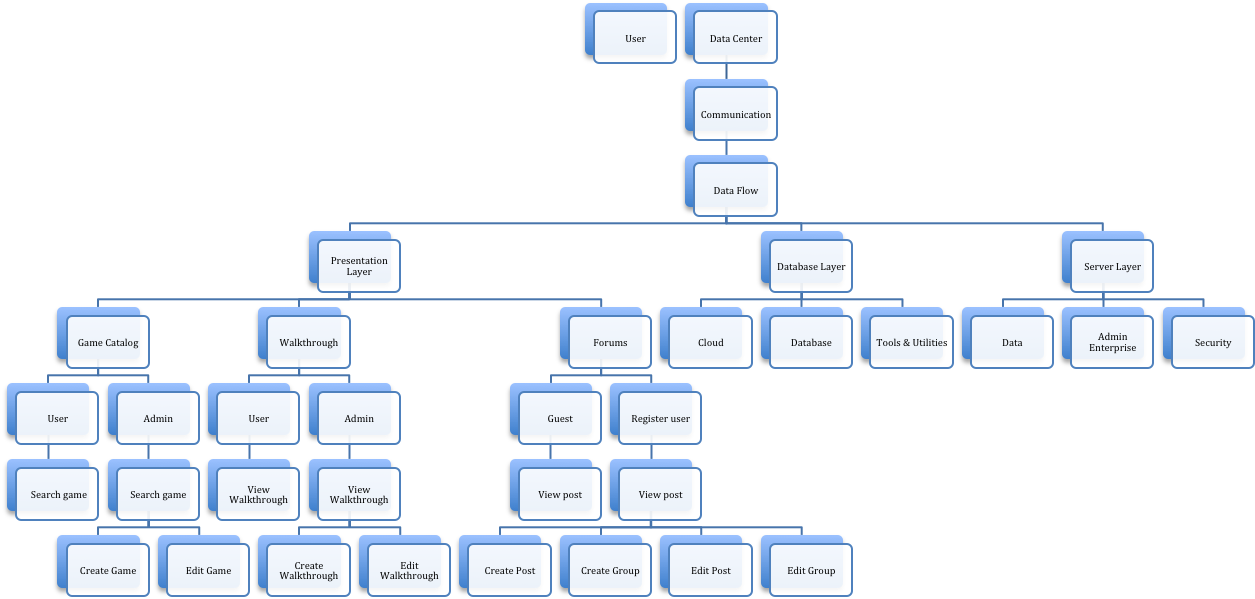
## Architectural Design

In order to reach our goal we are choosing the layer approach that has some elements from a data center approach to better implement the project at hand. It will be a hybrid between the two architectures. As shown below this will be the layer approach will implement and integrate into the website. Each level represents the operations of how each level system interacts with each other sub levels. The **User** **Level** is the top layer where the user defines the browser and the **UI** interacts with the **Presentation Layer**. From here the UI interact to display the interface layout for the user. This is done by, the **Presentation Layer** interacting with the **Database Layer** to retrieve the data. Thus displaying what is being accessed with the help of the **Server layer**. As this is the final layer and the one layer all layers fall on as this allows access to the whole web app. The **Server Layer** will interact with **Data center** when accessing remote data when need or when store excess data. Both the **Server Layer** and **Database Layer** will have access to the **Data Center** as this will be the behind the scenes data flow workings.



## Decomposition Description

At the top level we have the **User.** Depending on who the user is each level will unlock and give access to the next. Admin users will have complete access to each **Hierarchy** as were a register user or guest will only have access to the **Presentation layer Hierarchy**. Each level is broken down by it function as the **Data center** providing the overhaul of the data. As the **Data Center** sends and retrieves data it communications to where the Data flow must go. The **Presentation Layer** is the left most open access to all users depending on their credentials. From here the sub-level are the three main features of the site. Here you can find the **Game Catalog, Walkthrough, and Forums**. Each having a sub-level pertaining to the current user logged-in(**User** or **Admin**). Then to the left of the **Presentation Layer** is the **Database Layer** followed by **Server Layer** where these areas belong to **Admin** control. Each have their sub-level to interact with the site as a whole.



## Design Rationale

When deciding on which architecture model for our web app to fit our goals , we choose a hybrid approach of taking A **Data Center Architecture** and integrating with the **Layer Architecture**. The deciding factor lead to how our data needed to be store and how it would interact each component of our vision and features. We needed a design to where data can be called upon when requested, but at the same time allow for a systematic approach where UI can retrieve their counterpart depending on their necessary needs. We did consider the **Main Program/ Sub Program** approach, but found the call and return nature did not suit our ambitions, nor style of progress. This was due to having to create many sub programs to achieve the final goal. We also looked at **Data Flow Architecture**, but found the style too complex for how setting condition for data flow can be and we saw potential issue with planning user and admin access. All and All we believe the hybrid approach will suit what we are accomplishing here to create a simple, but interactive web application for gamers.

# Human Interface Design

## registration



The registration page will allow new users to register with the system. The user must enter all fields before submitting the form.

### INPUT & OUTPUT

#### Input:

* **First Name**: String representing the first name of user. Minimum of 1 and maximum of 50 characters.
* **Last Name**: String representing last name of user. Min 1 and maximum 50 characters
* **Email**: User’s email address used for communication and verification. Should match standard email pattern “name@server.com”. Minimum of 10 characters, maximum of 50 characters.
* **Username:** string representing the username can be a minimum of 6 characters and a maximum length of 20.
* **Password:** Also a string. Must be a minimum of 8 characters and a maximum of 20. Must contain on capital letter and one number.
* **Re-enter Password:** matches Password field. Used only for verification of password.

#### Output:

Upon successful creation of user, the system will display a welcome message or a warning message. In addition, if the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Welcome [First Name] [Last Name]. Thank you for registering with us!”

Failed: “There is an error with your information. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to store the user’s information in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** User has not registered with the system already. The system should validate that the email and username are unique within the stored user values.

**Post-conditions:** User record is stored successfully. A user session is created.

### Validation

The following validations will be performed on the submitted data:

* **First Name, Last Name:** fields must be at least the minimum specified length and no more than the maximum length.
* **Username:** must be unique
* **Email:** must be unique and must meet standard email address format
* **Password:** Must meet standard strong password policy. The password must be stored using a secure mechanism that will prevent unauthorized access. All passwords will be hashed and stored as such in the database.

## LOGIN



The login page will allow students/faculty to access the system’s features. They can also rest their password or unlock their account.

### INPUT & OUTPUT

#### Input:

* **Username:** string representing the username can be a minimum of 6 characters and a maximum length of 20.
* **Password:** Also a string. Must be a minimum of 8 characters and a maximum of 20. Must contain on capital letter and one number.

#### Output:

A session is created for the successful logged in user to access information granted to the specific user role. Also a message welcoming the user or an error (depending on the credentials provided)

### Actions

Once the user has entered their user name and password, the “Login” button will prompt the site to authenticate the user. Their role will be determined and they will be granted access to the modules available to that class of user. Based on the assigned role, the user might be re-routed to the faculty page or the student page.

### Pre and Post Conditions

**Pre-conditions:** Faculty and students both require a valid user name and password, the user name will be assigned upon acceptance or employment to the institution.

**Post-conditions:** Successful entry of credentials will result in a portal page. Unsuccessful entry of credentials will result in an error page, where the user is required to re-submit credentials.

### Validation

Both the user name and password will be validated by comparing them to the specific user credentials stored in the database.

**Password Security:** The password must be stored using a secure mechanism that will prevent unauthorized access. All passwords will be hashed and stored as such in the database.

## CHANGE PASSWORD



The Change Password page will allow the user to change their password by entering the current one, and the new password.

### INPUT & OUTPUT

#### Input:

* **Current Password:** String. Must be a minimum of 8 characters and a maximum of 20. Must contain on capital letter and one number.
* **New Password:** Also a string. Must be a minimum of 8 characters and a maximum of 20. Must contain on capital letter and one number.
* **Re-type Password:** Same requirements as New Password

#### Output:

If the user enter the correct “current password”, the system will store the new password and return a “Successfully change of password” message.

### Actions

Upon submission of the form, the system will validate the user has entered the correct password. To do so, the system will use the current session username and retrieve the current password from the database. The “Current Password” entered will be verified against the stored password and if successful, it will also validate both “New Password” and “Re-type Password” values are identical.

If all validation passes, then the system will update the user account with the new password entered.

### Pre and Post Conditions

**Pre-conditions:** User is already registered and the account is active.

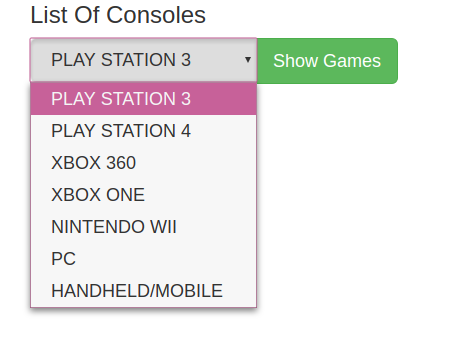
**Post-conditions:** Successful transaction will update the user’s password to the specified new values.

### Validation

Current password will be validated against the stored password. New password will be validated against the “Re-type password” value, which should match. Also, new password must meet strong password policy.

**Password Security:** The password must be stored using a secure mechanism that will prevent unauthorized access. All passwords will be hashed and stored as such in the database.

## List of games of selected console



This section will allow users to select a console and get results of all games available to play for the console.

### INPUT & OUTPUT

#### Input:

**Console**: List of consoles representing available consoles on the web application.

#### Output:

Upon “Show Games” button click, a list of games that are available for the console are displayed.

**Possible messages:**

Successful: List of games

Failed: “There are no games available for the console”.

### Actions

Upon submission of the form, the system will validate the user has entered the correct password. To do so, the system will use the current session username and retrieve the current password from the database. The “Current Password” entered will be verified against the stored password and if successful, it will also validate both “New Password” and “Re-type Password” values are identical.

If all validation passes, then the system will update the user account with the new password entered.

### Pre and Post Conditions

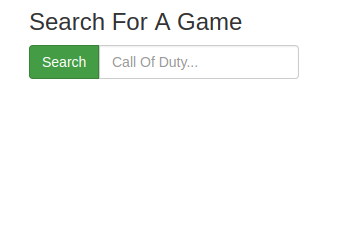
**Pre-conditions:** Games available for console

**Post-conditions:** Successful transaction displays games playable on console

### Validation

Console has to be on the list for selection.

## Search for a game



This section will allow users to search for a game by entering a keyword into the search bar.

### INPUT & OUTPUT

#### Input:

**Game Title**: String length of no less than 1 character representing title of game.

#### Output:

Upon “Search” button click, a list of games that have the entered keyword in their title are displayed.

**Possible messages:**

Successful: List of games

Failed: “No game under the name of title”.

### Actions

Upon submission, the system will search and display games that have the keyword in their title. The system will do so by running a query that checks for matching strings between keyword and all game titles.

### Pre and Post Conditions

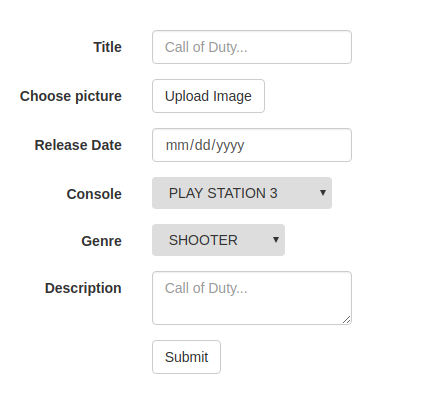
**Pre-conditions:** There are games that have titles with the keyword.

**Post-conditions:** Successful transaction will display games.

### Validation

The system will check for valid characters. Upon submit, the search bar must not be left blank or it will return no results.

## Add a game



This section will allow administrators to add a new game to the database.

### INPUT & OUTPUT

#### Input:

**Title**: String length of no less than 3 characters representing title of game.

**Picture:** JPEG or PNG file representing a picture of the game.

**Release Date**: Date of game release in date format of DD/MM/YYYY.

**Console**: List of game consoles that the game can be played in.

**Genre**: List of genres of the game.

**Description**: String length of no less than 10 characters representing a brief description of the game.

#### Output:

Upon entry game will either display a success message or an error message

**Possible messages:**

Successful: “Game has be successfully added to the database”

Failed: “Game could not be added to the system” with a specific reason.

### Actions

Upon submission of the form, the system will validate the form. The system will do so by checking for valid characters and filled text fields. It will also check for valid date and make sure that description is no less than 15 characters (excluding spaces).

### Pre and Post Conditions

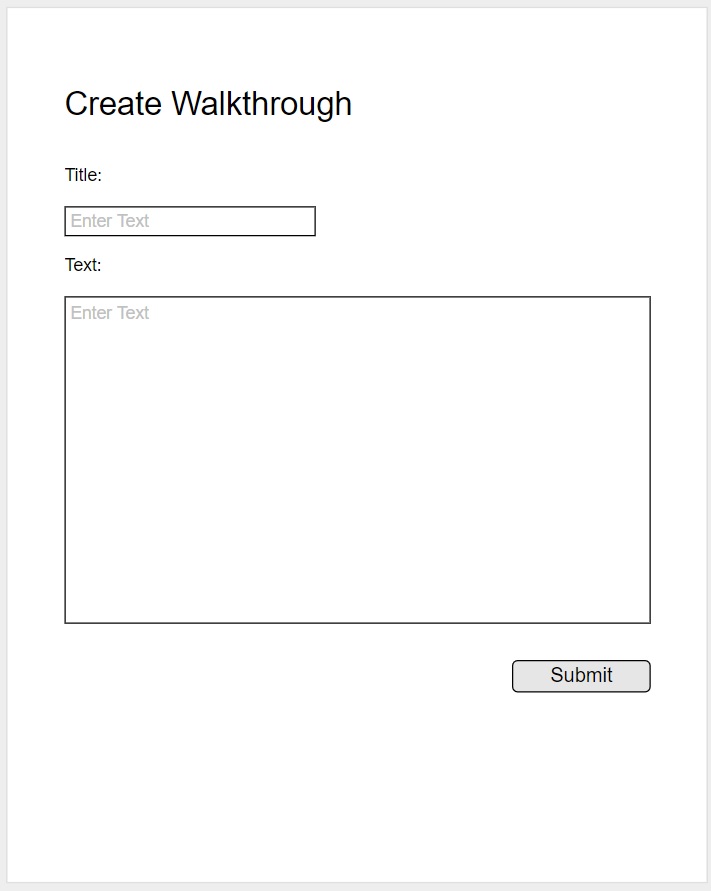
**Pre-conditions:** Game is already in the database

**Post-conditions:** Successful transaction will add new game to the system

### Validation

All fields must be filled and date must match required date format.

## Create walkthrough



The Create Walkthrough page will allow admins to create new walkthroughs in the system. The admin must enter all fields before submitting the form.

### INPUT & OUTPUT

#### Input:

* **Title**: String representing the title of the walkthrough. Min of 1 and maximum of 50 characters.
* **Text**: String representing the text of the walkthrough. Min 1 character and maximum of 8000 characters.

#### Output:

Upon successful creation of walkthrough, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Walkthrough successfully created!”

Failed: “There is an error with the walkthrough. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to store the walkthrough in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** User is an admin.

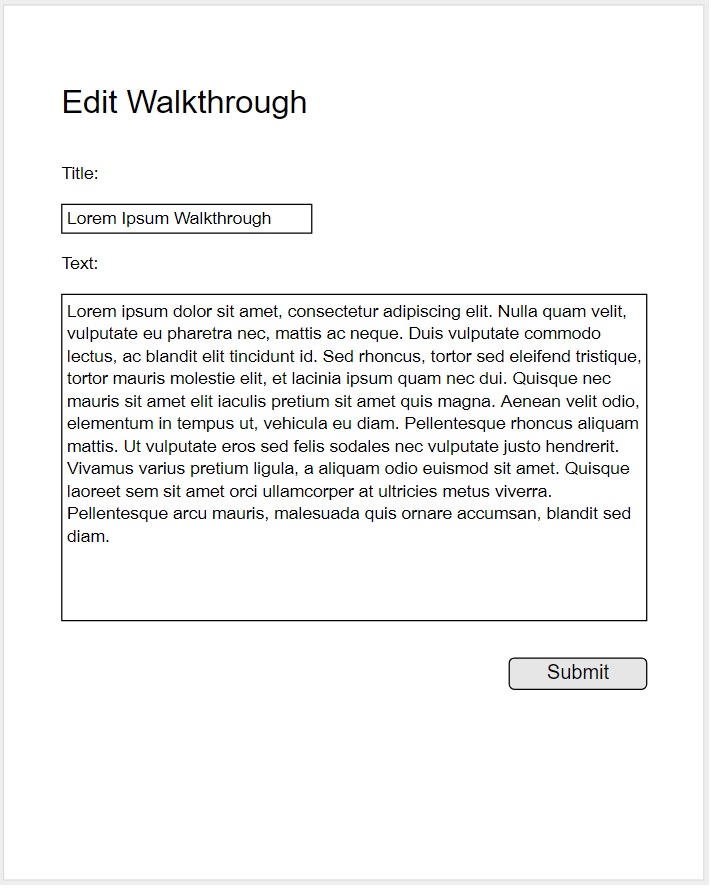
**Post-conditions:** Walkthrough is stored successfully.

### Validation

The following validations will be performed on the submitted data:

* **Title:** field must be at least the minimum specified length and no more than the maximum length.
* **Text:** field must be at least the minimum specified length.

## Edit Walkthrough



The Edit Walkthrough page will allow admins to edit existing walkthroughs.

### INPUT & OUTPUT

#### Input:

* **Title**: String representing the title of the walkthrough. Min of 1 and maximum of 50 characters.
* **Text**: String representing the text of the walkthrough. Min 1 character and maximum of 8000 characters.

#### Output:

Upon successful editing of the walkthrough, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Walkthrough successfully edited!”

Failed: “There is an error with the walkthrough. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to modify the walkthrough in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** Walkthrough exists. User is an admin.

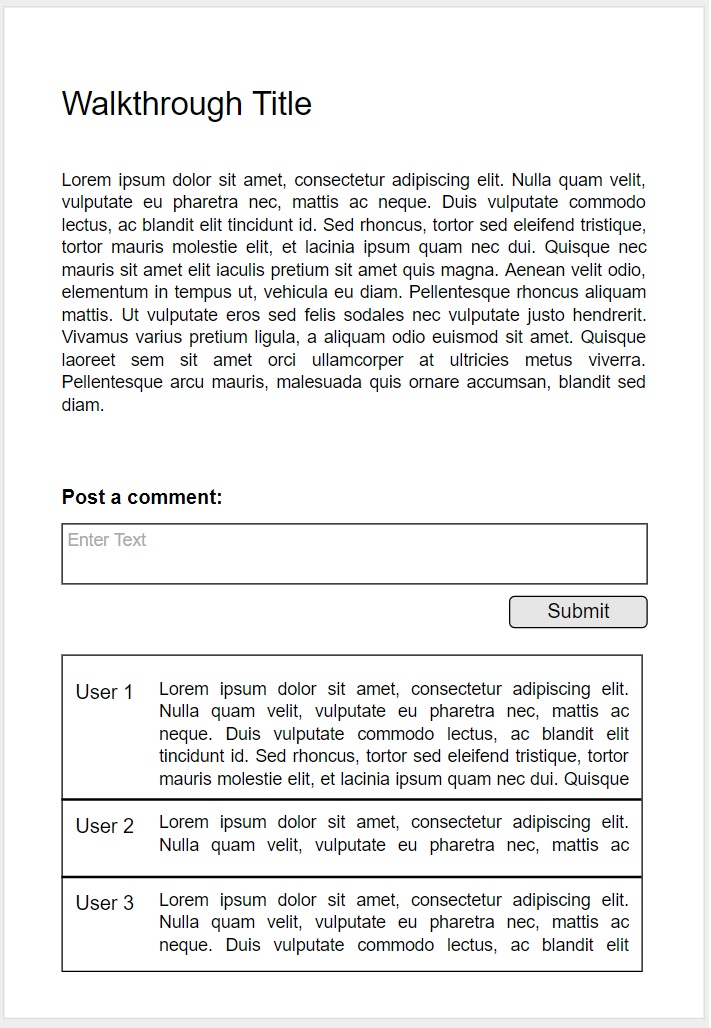
**Post-conditions:** Walkthrough is edited successfully.

### Validation

The following validations will be performed on the submitted data:

* **Title:** field must be at least the minimum specified length and no more than the maximum length.
* **Text:** field must be at least the minimum specified length.

## View walkthrough



The View Walkthrough page will allow the user to view the walkthrough and post comments.

### INPUT & OUTPUT

#### Input:

* **Comment:** String representing the text of the walkthrough. Min 1 character and maximum 255 characters.

#### Output:

If the user successfully submits a comment, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

### Actions

The selected walkthrough is displayed with saved comments.

Upon submitting the comment form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to create a comment for the walkthrough in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** User is already registered and the account is active.

**Post-conditions:** Successful transaction will add a post to the walkthrough.

### Validation

The following validations will be performed on the submitted data:

* **Comment:** field must be at least the minimum specified length and no more than the maximum length.

## Forum Login



The Forum login page will allow User to access the system’s features and give access to posting and editing abilities otherwise can be ignore to view only.

### INPUT & OUTPUT

#### Input:

* **Username:** string representing the username can be a minimum of 6 characters and a maximum length of 20.
* **Password:** Also a string. Must be a minimum of 8 characters and a maximum of 20. Must contain on capital letter and one number.

#### Output:

A session is created for the successful logged in user to access information granted to the specific user role. Also a message welcoming the user or an error (depending on the credentials provided)

### Actions

Once the user has entered their user name and password, the “Login” button will prompt the site to authenticate the user. Their role will be determined and they will be granted access to the modules available to that class of user. Based on the assigned role, the user might be re-routed to the Admin page or the Register User page.

### Pre and Post Conditions

**Pre-conditions:** Admin and Registered User both require a valid user name and password, the user name will be assigned upon acceptance to the site.

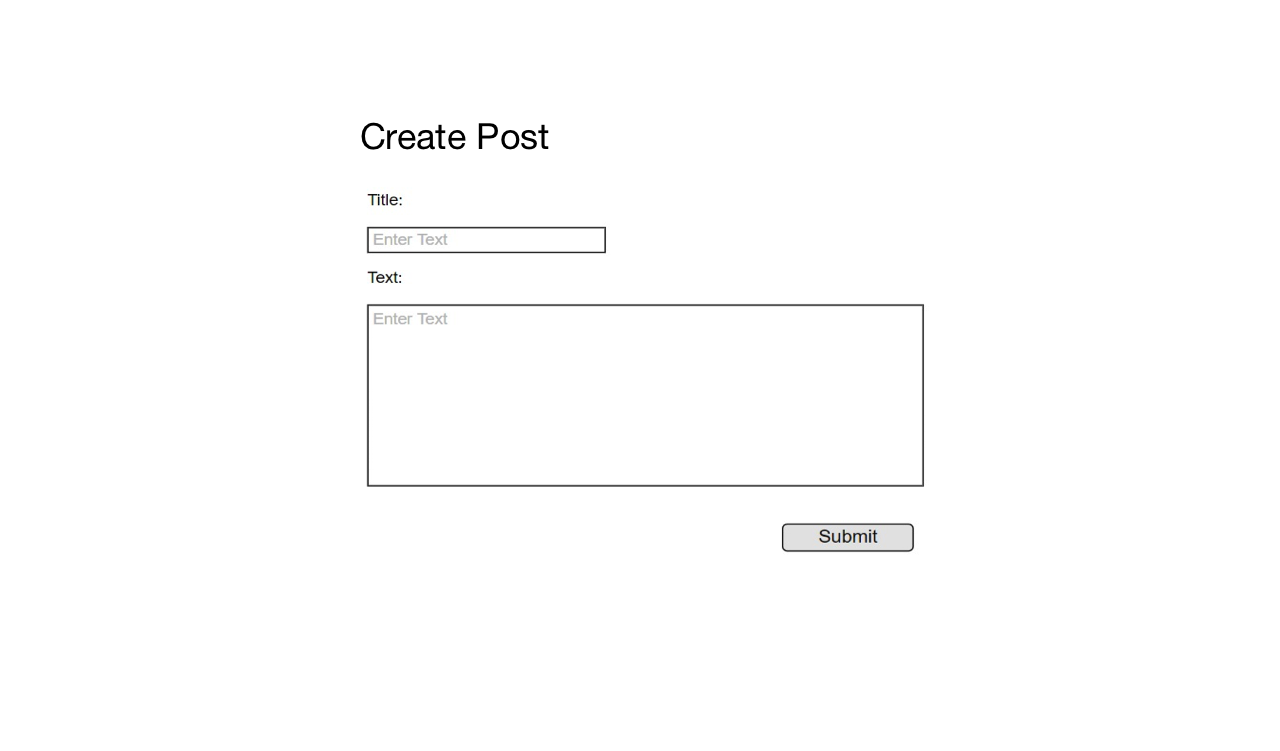
**Post-conditions:** Successful entry of credentials will result in a portal page. Unsuccessful entry of credentials will result in an error page, where the user is required to re-submit credentials or cancel altogether to view only.

### Validation

Both the user name and password will be validated by comparing them to the specific user credentials stored in the database.

**Password Security:** The password must be stored using a secure mechanism that will prevent unauthorized access. All passwords will be hashed and stored as such in the database.

## Create post



The Create Post page will allow user to create new post/thread in the system. The user must enter all fields before submitting the post.

### INPUT & OUTPUT

#### Input:

* **Title**: String representing the title of the Thread. Min of 1 and maximum of 50 characters.
* **Text**: String representing the text of the Thread. Min 1 character and maximum of 2500 characters.

#### Output:

Upon successful creation of Thread, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Thread successfully created!”

Failed: “There is an error with the Thread. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to store the post in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** User is Registered.

**Post-conditions:** Thread is stored successfully.

### Validation

The following validations will be performed on the submitted data:

* **Title:** field must be at least the minimum specified length and no more than the maximum length.
* **Text:** field must be at least the minimum specified length.

## Edit post

****

The Edit Post page will allow Registered User to edit existing Threads/Post.

### INPUT & OUTPUT

#### Input:

* **Title**: String representing the title of the walkthrough. Min of 1 and maximum of 50 characters.
* **Text**: String representing the text of the walkthrough. Min 1 character and maximum of 2500 characters.

#### Output:

Upon successful editing of the post, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Post successfully edited!”

Failed: “There is an error with the Post. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to modify the post in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** Post exists. User is registered and is the user of post.

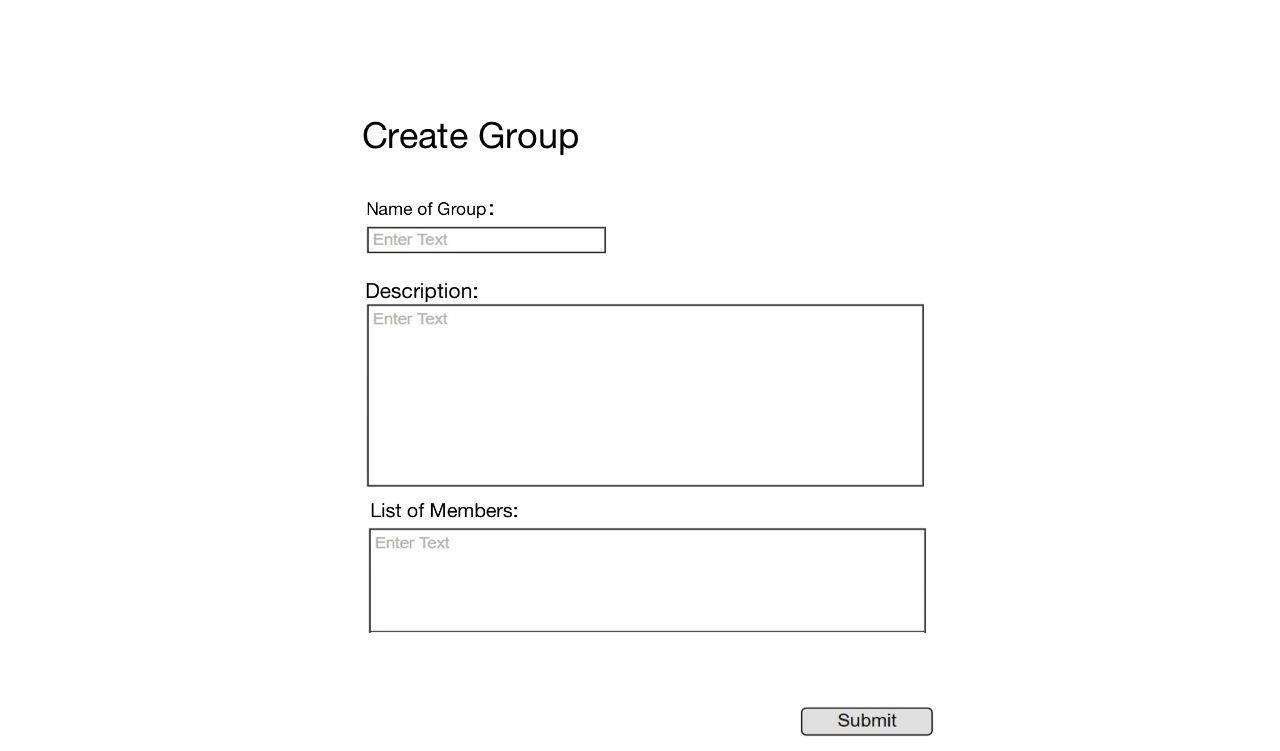
**Post-conditions:** Post is edited successfully.

### Validation

The following validations will be performed on the submitted data:

* **Title:** field must be at least the minimum specified length and no more than the maximum length.
* **Text:** field must be at least the minimum specified length.

## Create Group



The Create Group page will allow user to create new Group in the system. The user must enter all fields before submitting the post and be registered.

### INPUT & OUTPUT

#### Input:

* **Name of Group**: String representing the title of the Thread. Min of 1 and maximum of 50 characters.
* **Description**: String representing the text of the Thread. Min 1 character and maximum of 2500 characters.
* **List of Member**: String representing the text of the Thread. Min 1 character and maximum of 2500 characters.

#### Output:

Upon successful creation of Group, the system will display a confirmation message. If the data entered is not valid, the system will show a list of the fields with invalid data in red font.

**Possible messages:**

Successful: “Group successfully created!”

Failed: “There is an error with the Group. Please review the fields highlighted in red and re-enter the information”

### Actions

Upon submitting the form, the system will validate that all required fields have input, and that each field matches the required data in the required format.

If the validation is successful, the system will run a query to store the Group in the database.

If the validation failed, the system will abort transaction and return an error message for each invalid field.

### Pre and Post Conditions

**Pre-conditions:** User is Registered and Group Name doesn’t not exist already.

**Post-conditions:** Group is stored successfully.

### Validation

The following validations will be performed on the submitted data:

* **Group Name:** field must be at least the minimum specified length and no more than the maximum length.
* **Description:** field must be at least the minimum specified length.
* **Members:** members must exist and be registered.

# Component Design



## Registeration Entities

**User**: The User class represents both customer and staff entities. It is the base class for all user information. This class is responsible for managing the user personal information as well as defining the formatting of names.

User Attributes/Fields:

* User ID: unique identifier for a single User instance
* FirstName: The first name of the user
* LastName: The last name of the user
* Email: a string representing a valid email address in the form of [account]@[domain].[suffix]
* Username: a unique string chosen by the user to gain access to the system
* Password: a string representing the password selected by the user to gain access to the Web site. It should be no less than 8 chararcters
* UserType: An association to the UserType entity. It defines the type of the current user instance

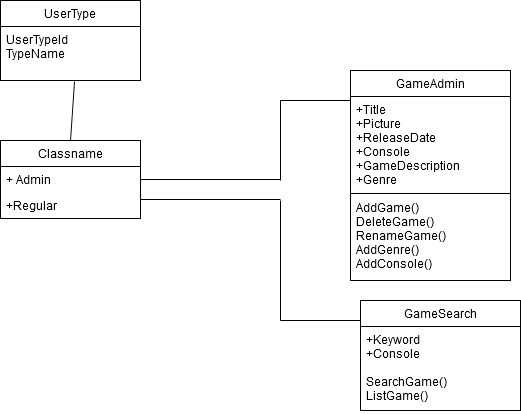
Relationships:

* UserType: Each user has a one to one relationship with a type entity.
* HashingProvider: The User object will depend on the HashingProvider entity which will hash the passwords for the current user object.

**UserType**: Entity that identifies one of the possible types of users. Possible options might be “customer”, “admin”, “guest”

**HashingProvider**: This object will be responsible for hashing the user passwords using one of the standard algorithm. The “provider” will depend on a third party library for the hashing algorithm.

HashingAlgorithms: This package is TBD based on hashing requirements.



## Game Catalog Entities

**GameAdmin**: The GameAdmin class represents the administrators’ action when adding or deleting a game.

GameAdmin Attributes/Fields:

* Title: The title of the game
* Picture: Link to the picture of the game
* ReleaseDate: Date that the game was released in DD/MM/YYYY
* Console: The console that the game can be played in
* GameDescription: Brief description of game
* Genre: String representing a genre of the game

Relationships:

* UserType: UserType Admin is the one that can access this class

**UserType**: Entity that identifies one of the possible types of users. Possible options might be “regular”, “admin”, “guest”

**GameSearch**: The GameSearch class represents a regular user or guest’s action when searching for gmaes

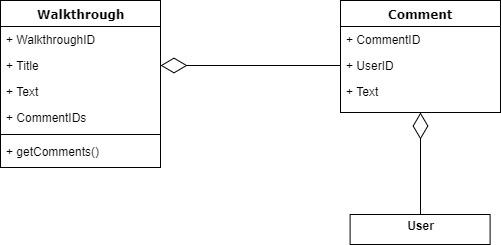
GameSearch Attributes/Fields:

* Keyword: String representation of game title entered by the user
* Console: String representation of available consoles that the user can select from

Relationships:

* UserType: a registered user or guest can search for games

**UserType**: Entity that identifies one of the possible types of users. Possible options might be “regular”, “admin”, “guest”



## Walkthrough Entities

**Walkthrough**: The Walkthrough class contains the data for a specific walkthrough.

User Attributes/Fields:

* Walkthrough ID: unique identifier for a single walkthrough
* Title: contains the title of the walkthrough
* Text: contains the text of the walkthrough
* CommentIDs: contains all of the comment IDs for its comments

Relationships:

* Comment: Each walkthrough has a one to many relationship with the comment class.

**Comment**: The Comment class contains the data for a specific comment.

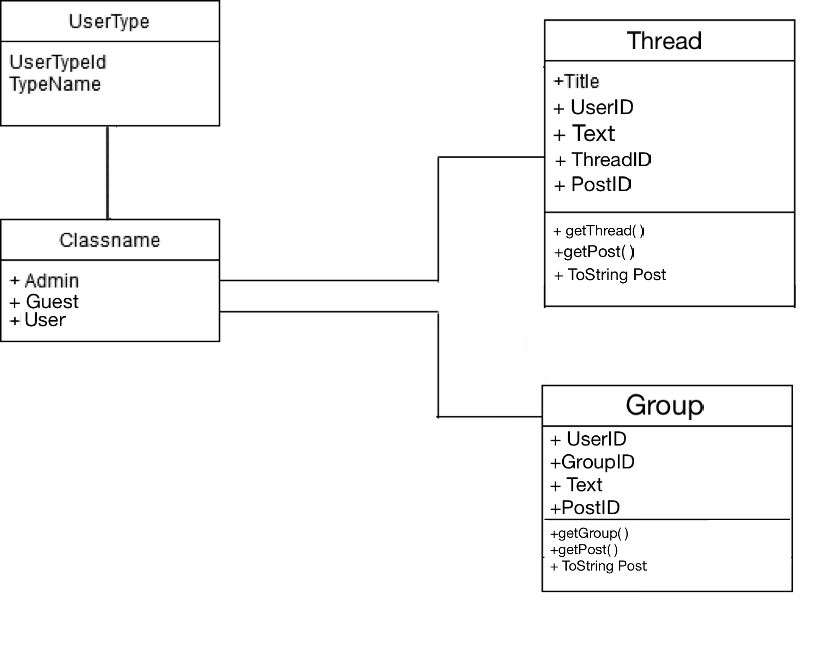
User Attributes/Fields:

* Comment ID: unique identifier for a single comment
* User ID: contains the ID of the author of the comment
* Text: contains the text of the comment

Relationships:

* Walkthrough: Each walkthrough has a one to many relationship with the comment class.
* User: Each comment has a one to one relationship with a user

**User**: Previously defined entity that identifies registered users



## Forums Entities

**Thread**: The Thread class contains the data for a specific Thread.

User Attributes/Fields:

* Thread ID: unique identifier for a Thread
* Title: contains the title of the Thread
* Text: contains the text of the Thread/Post
* PostID: contains all of the post IDs for its user
* User ID: contains the ID of the author of the post or thread

Relationships:

* UserType: Each User type has a one to many relationship with the Thread class.

**Group**: The Group class contains the data for a specific Group.

User Attributes/Fields:

* Group ID: unique identifier for a single Group
* User ID: contains the ID of the author of the post or group
* Text: contains the text of the Group
* PostID: contains all of the post IDs for its user in that group

Relationships:

* **UserType:** Each User type has a one to many relationships with the Group class.

**UserType**: Entity that identifies one of the possible types of users. Possible options might be “user”, “admin”, “guest”

# Data Design

## Data Description

## Registration / Authentication

From the below diagram, we identified a relationship between a user and his/her type. The possible types will be Administrator and Customer. The relationship “IsType” does not include any specific attributes.

Another relationship for the user is the Game Reviews. In this particular relationship, we have unique attributes that are not part of the user and neither are they part of the game. Those attributes are only related to the user



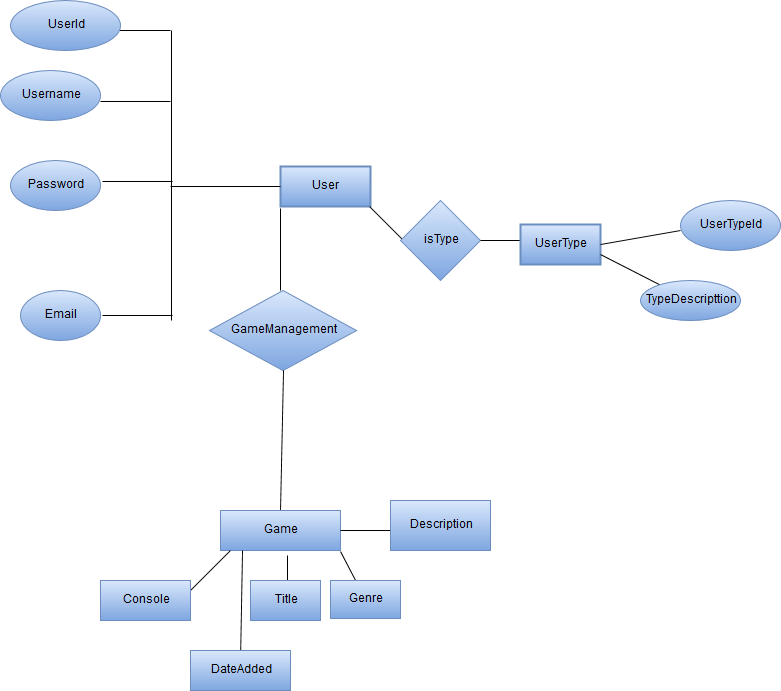
review of a game. For example, the Review Description and the When Reviewed attributes are not part of the user, but they do describe the relationship of the user and game through a review.

Please include in the diagram a numeric quantity on the relationships.

## Game catalog

From the below diagram, we identified a relationship between a user and his/her type. The possible types will be Administrator and Customer. The relationship “IsType” does not include any specific attributes.

Another relationship for the user is the Game Management. In this particular relationship, we have unique attributes that are not part of the user but are parts of the game. Those attributes are only related to game.

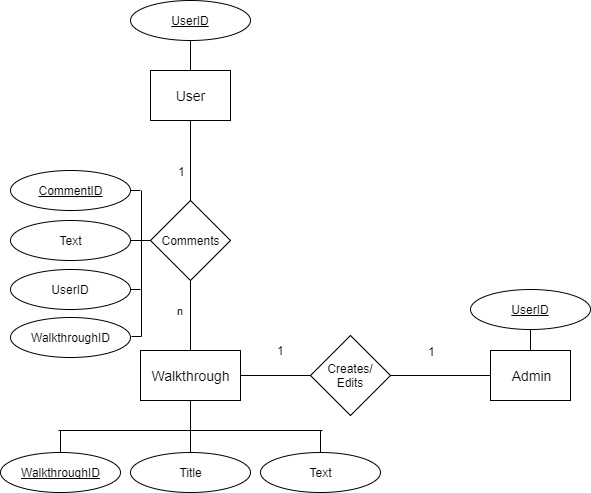


## Walkthrough

From the below diagram, we identified a relationship between a user, a comment, and a walkthrough.

A user comments on a walkthrough. In this particular relationship, we have unique attributes that are not part of the user and neither are they part of the walkthrough. Those attributes are only related to the user comment on a walkthrough.

A user, as an admin, can create or edit walkthroughs

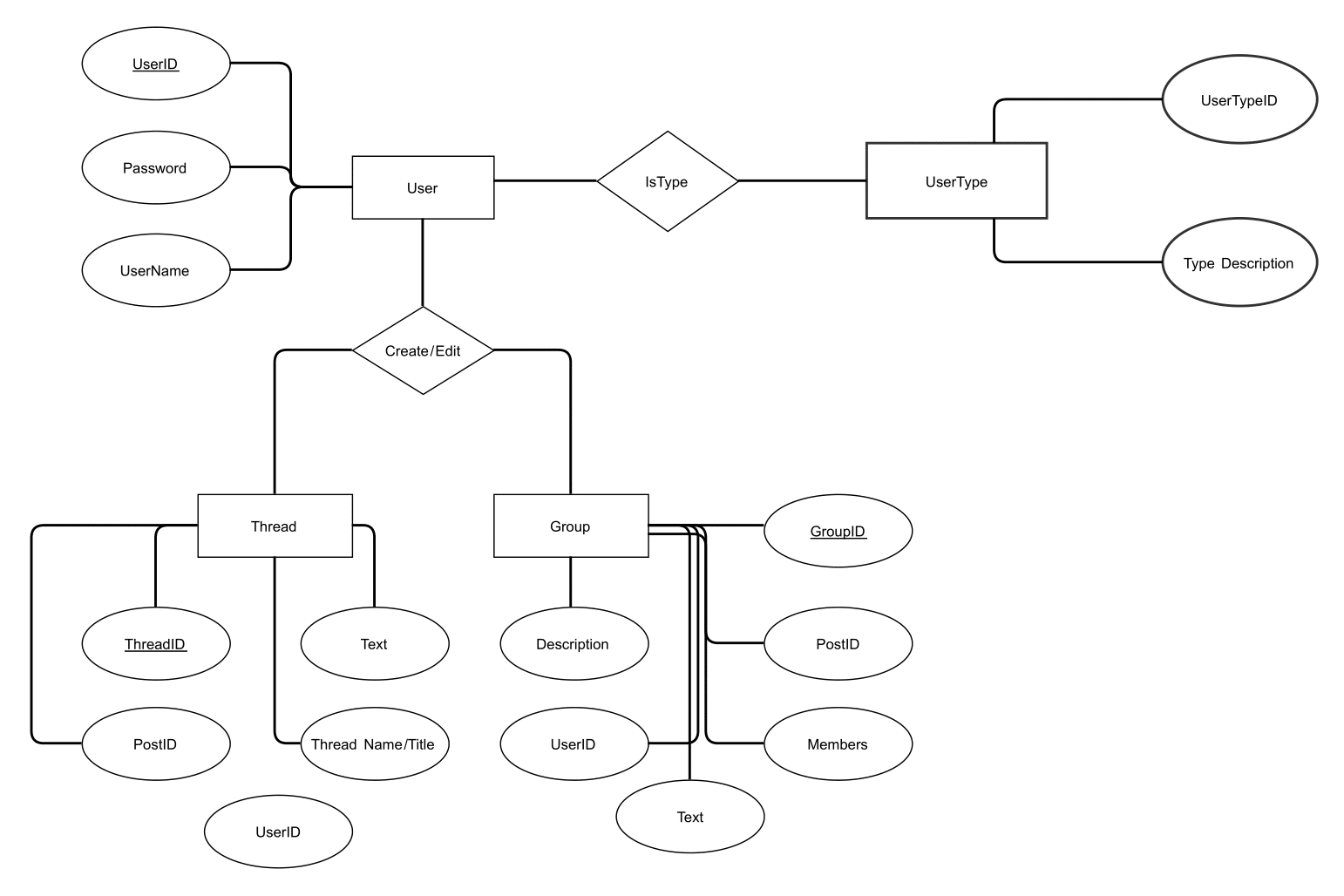


## Forums

From the below diagram, we identified a relationship between a user, a Thread, and a Group

A registered user Can create or edit Groups or Threads. In this particular relationship, we have unique attributes that are not part of the user and neither are they part of the Thread of Group. Those attributes are only related to the user on a Group or on Thread.

A user or admin, can create or edit Groups and Threads



# DATABASE Dictionary / SCHEMA

## Registration / Authentication

The following tables are defined for the Registration/Authentication module

|  |  |  |  |
| --- | --- | --- | --- |
| **User Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **user\_id** | INT | Not null | PK. Unique identifier of a user. |
| **user\_type\_id** | INT | Not null | FK. Relationship key to the User\_Types table |
| **user\_pw** | varchar(50) | Not null | User’s password HASH |
| **first\_name** | varchar(30) | Not null | User’s first name |
| **last\_name** | varchar(30) | Not null | User’s last name |
| **DOB** | Date | Not null | User’s date of birth |
| **SSN** | varchar(9) | Not null | User’s social security number |
| **Phone** | varchar(10) | Not null | User’s phone number |
| **Email** | varchar(50) | Not null | User’s E-mail address |
| **Address** | varchar(50) | Not null | User’s home address |
| **City** | varchar(20) | Not null | City |
| **State** | varchar(2) | Not null | State |
| **Zip\_code** | varchar(5) | Not null | Zip code |

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_Types Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **user\_type\_id** | INT | Not Null | PK. Unique identifier of a user. |
| **type\_name** | varchar(50) | Not null | The name of the user type |
| **type\_description** | varchar(50) | Not null | Description of the user type and its uses |

## Game catalog

The following tables are defined for the Game Catalog module

|  |  |  |  |
| --- | --- | --- | --- |
| **Game** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **Game\_id** | INT | Not null | PK. Unique identifier of a name |
| **game\_title** | varchar(30) | Not null | Title of the game |
| **Game\_desc** | varchar(255) | Not null | Description of the game |
| **Game\_genre** | varchar | Not null | List of genres separated by commas |
| **Date\_added** | Date | Not null | The date the game was added |
| **Game\_console** | varchar(255) | Not null | List of consoles separated by commas |

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_Types Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **user\_type\_id** | INT | Not Null | PK. Unique identifier of a user. |
| **type\_description** | varchar(50) | Not null | Description of the user type and its uses |

## Walkthrough

The following tables are defined for the Walkthrough module

|  |  |  |  |
| --- | --- | --- | --- |
| **Walkthrough Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **walkthrough\_id** | INT | Not null | PK. Unique identifier of a walkthrough. |
| **title** | varchar(50) | Not null | Walkthrough’s title |
| **text** | varchar(8000) | Not null | Walkthrough’s text |

|  |  |  |  |
| --- | --- | --- | --- |
| **Comment Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **comment\_id** | INT | Not Null | PK. Unique identifier of a comment. |
| **walkthrough\_id** | INT | Not null | FK. Relationship key to the walkthrough table. |
| **User\_id** | INT | Not null | FK. Relationship key to the user table. |
| **Text** | varchar(255) | Not null | Comment’s text. |

## Forums

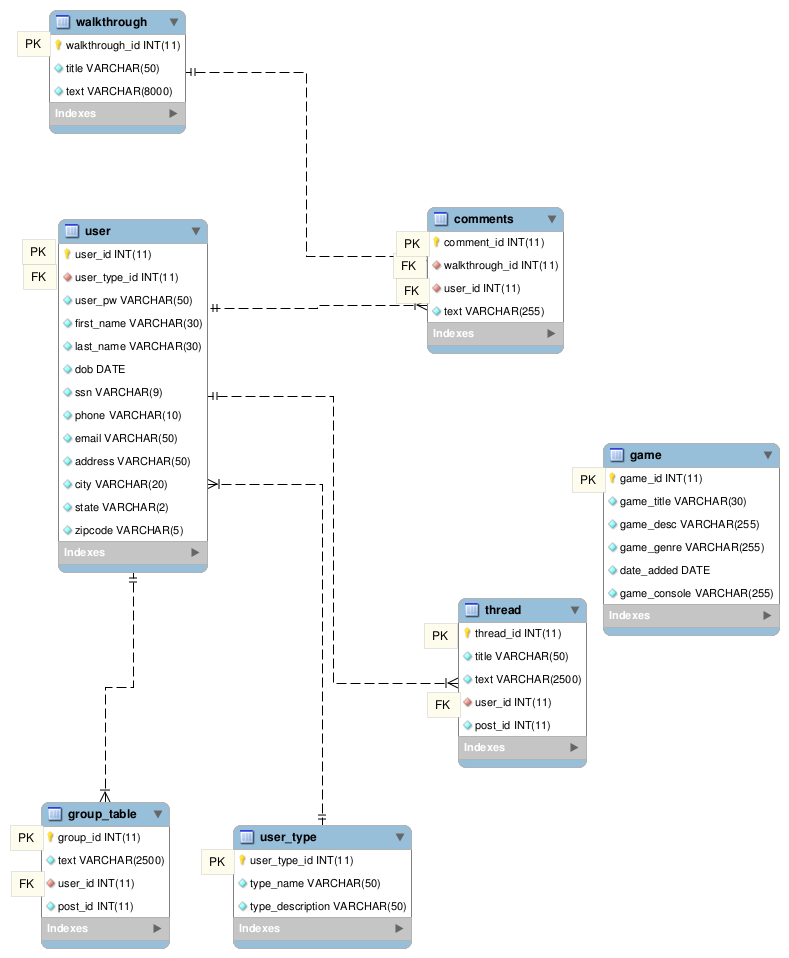
The following tables are defined for the Forums module

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_Types Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **user\_type\_id** | INT | Not Null | PK. Unique identifier of a user. |
| **type\_description** | varchar(50) | Not null | Description of the user type and its uses |

|  |  |  |  |
| --- | --- | --- | --- |
| **Thread Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **Thread\_id** | INT | Not null | PK. Unique identifier of a Thread. |
| **title** | varchar(50) | Not null | Thread’s title |
| **text** | varchar(2500) | Not null | Thread’s / post text |
| **User\_id** | INT | Not null | FK. Relationship key to the user table. |
| **Post\_id** | INT | Not null | CK. Relationship key to the Thread table. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Group Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **Group\_id** | INT | Not null | PK. Unique identifier of a Group. |
| **text** | varchar(2500) | Not null | Group text |
| **User\_id** | INT | Not null | FK. Relationship key to the user table. |
| **Post\_id** | INT | Not null | CK. Relationship key to the Group table. |

# Full DAtabase Model



# Requirements Matrix

|  |  |  |
| --- | --- | --- |
| ***Module*** | ***Use Case*** | ***Design Component*** |
| ***Registration Module*** | *Authentication* | *User Interface: Login*  *ERD Diagram 1*  *Component Model: Registration*  *DB Schema section1* |
|  | *Change Password* | *User Interface: Change Pwd*  *ERD Diagram 1*  *Component Model: Registration*  *DB Schema section1* |
| ***Game Catalog Module*** | *List Games of Selected Console* | *User Interface: List Games of Selected Console*  *ERD Diagram 2*  *Component Model: Game Catalog*  *DB Schema section2* |
|  | *Search For a Game* | *User Interface: Search For a Game*  *ERD Diagram 2*  *Component Model: Game Catalog*  *DB Schema section2* |
|  | *Add a Game* | *User Interface: Add a Game*  *ERD Diagram 2*  *Component Model: Game Catalog*  *DB Schema section2* |
| ***Walkthrough*** | *Create Walkthrough* | *User Interface: Create Walkthrough*  *ERD Diagram 3*  *Component Model: Walkthrough*  *DB Schema section 3* |
|  | *Edit Walkthrough* | *User Interface: Edit Walkthrough*  *ERD Diagram 3*  *Component Model: Walkthrough*  *DB Schema section 3* |
| ***Forums*** | *Create Post* | *User Interface: Create Post*  *ERD Diagram 4*  *Component Model: Forum*  *DB Schema section 4* |
|  | *Edit Post* | *User Interface: Edit Post*  *ERD Diagram 4*  *Component Model: Forum*  *DB Schema section 4* |
|  | *Create Group* | *User Interface: Create Group*  *ERD Diagram 4*  *Component Model: Forum*  *DB Schema section 4* |

# Appendices

*Amazon AWS cloud service will store all our data to help manage memory restrictions and serve as a back up server is in house server fails.*

*Were are able to communicate with the customer on frequent matter on bi-weekly bases to maintain a good standing relationship with client.*

*We are to be made available at all times and if not you are to response back within 24 hrs.*