GAMEON GAME HUB

ANALYSIS AND DESIGN DOCUMENT

**Team 4**

**Zainab Akubat**

**Richard Hall**

**Muhammed Mozzam Khan**

# Contents

[Contents 2](#_Toc412187801)

[Introduction 4](#_Toc412187802)

[System Architecture 4](#_Toc412187803)

[System Overview 5](#_Toc412187804)

[Architectural Design 5](#_Toc412187805)

[Decomposition Description 5](#_Toc412187806)

[Design Rationale 5](#_Toc412187807)

[Human Interface Design 6](#_Toc412187808)

[registration **Error! Bookmark not defined.**](#_Toc412187809)

[INPUT & OUTPUT 8](#_Toc412187810)

[Actions 8](#_Toc412187811)

[Pre and Post Conditions 8](#_Toc412187812)

[Validation 8](#_Toc412187813)

[LOGIN **Error! Bookmark not defined.**](#_Toc412187814)

[INPUT & OUTPUT **Error! Bookmark not defined.**](#_Toc412187815)

[Actions **Error! Bookmark not defined.**](#_Toc412187816)

[Pre and Post Conditions **Error! Bookmark not defined.**](#_Toc412187817)

[Validation **Error! Bookmark not defined.**](#_Toc412187818)

[CHANGE PASSWORD **Error! Bookmark not defined.**](#_Toc412187819)

[INPUT & OUTPUT **Error! Bookmark not defined.**](#_Toc412187820)

[Actions **Error! Bookmark not defined.**](#_Toc412187821)

[Pre and Post Conditions **Error! Bookmark not defined.**](#_Toc412187822)

[Validation **Error! Bookmark not defined.**](#_Toc412187823)

[Component Design 9](#_Toc412187824)

[Entities 9](#_Toc412187825)

[Data Design 9](#_Toc412187826)

[Data Description 9](#_Toc412187827)

[Registration / Authentication 9](#_Toc412187828)

[DATABASE Dictionary / SCHEMA 11](#_Toc412187829)

[Registration / Authentication **Error! Bookmark not defined.**](#_Toc412187830)

[Full DAtabase Model **Error! Bookmark not defined.**](#_Toc412187831)

[Requirements Matrix **Error! Bookmark not defined.**](#_Toc412187832)

[Appendices **Error! Bookmark not defined.**](#_Toc412187833)

# Introduction

* ***Purpose:***
  + *Currently, a gamer has no centralized location to get all of their gaming needs.*
  + *Our software will provide a “one stop shop.” Gamers will no longer need to roam the internet hoping to find the relevant information they require for their game of choice.*
  + *Our objective is to offer competition to sites such as Steam and Origin. We plan on optimizing our loading times to less than one second per page load. We plan on accomplishing this goal by providing simple, yet flamboyant layouts, based on PHP, that quickly inform our customers of exactly what they need to know, whenever they need to know it.*
* ***Scope:***
  + *Our site will generate a custom web experience based on the database management software PHP. A user will request information that our database will return, tailored to the user.*
  + *Our software will allow users to search for games, read reviews, sort games by popular headings such as ratings, genre, platform, top selling, and top fan favorites.*
  + *Our software will provide a closed community of gamers without the fear of flaming or spam. This will be moderated by our community log in membership requirement.*
  + *A benefit of our software is that we strive to build a community of gamers who share in the gaming experience, building a network of trusted peers within the gaming community.*
* ***Definitions, acronyms and abbreviations****:*

***Admin:*** *An elevated system user who has extended privileges including viewing question metadata, creating other admins, answering forum questions, adding definitions to games in the database, and editing customer questions in the forum.*

***Customer****: any individual who registers on “GameOn” is considered a customer.*

***Database****: our continuously updated reservoir of knowledge, in digital form, that a user may access.*

***Gamer****: an all-inclusive term that simply means any individual interested in learning more about popular video games.*

***PHP****: is a server-side scripting language designed for web development but also used as a general-purpose programming language.*

# System Architecture

*(This section is only one per group) High level description of the system involving modules and components. Include Context diagrams and a description of the architecture planned.*

## System Overview

*Give a general description of the functionality, context and design of your project. Provide any background information if necessary.*

## Architectural Design

*Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high level overview of how responsibilities of the system were partitioned and then assigned to subsystems. Identify each high level subsystem and the roles or responsibilities assigned to it. Describe how these subsystems collaborate with each other in order to achieve the desired functionality. Don’t go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together. Provide a diagram showing the major subsystems and data repositories and their interconnections. Describe the diagram if required.*

## Decomposition Description

*Provide a decomposition of the subsystems in the architectural design. Supplement with text as needed. You may choose to give a functional description or an object­oriented description. For a functional description, put top­level data flow diagram (DFD) and structural decomposition diagrams. For an OO .description, put subsystem model, object diagrams, generalization hierarchy diagram(s) (if any), aggregation hierarchy diagram(s) (if any), interface specifications, and sequence diagrams here.*

## Design Rationale

*Discuss the rationale for selecting the architecture described in 3.1 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.*

# Human Interface Design

## registration



The registration page will allow new users to register with the system as well as having the option to subscribe to Newsletters from GameOn. The user must enter all fields before choosing the register button.

### INPUT & OUTPUT

#### Input:

* **First Name**: String representing the first name of user. Minimum of 1 and maximum of 30 characters.
* **Last Name**: String representing last name of user. Min 1 and maximum 30 characters.
* **Email**: User’s email address used for communication and receiving Newsletters if chosen. Should match standard email pattern “[name@server.com](mailto:name@server.com)”. Minimum of 10 characters, maximum of 30 characters.
* **Username:** string representing the username can be a minimum of 3 characters and a maximum length of 20.
* **Password:** Also a string. Must be a minimum of 10 characters and a maximum of 20. Must contain one capital letter.
* **Check box:** The user will leave the check box marked if they want to receive Newsletters. If the user does not want to receive Newsletters they will uncheck the check box.

#### Output:

Upon successful creation of user, the system will return success and display a message saying, “Thank You”. In addition, if the data entered is not valid, the system will be failed and will notify the user. The user will correct the errors and click Register once again.

**Possible messages:**

Successful: “Thank You”

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.

## create newsletter



The Newsletter page will allow the administrator to create a new Newsletter and save it, so the administrator can send out emails to registered users.

### INPUT & OUTPUT

#### Input:

* **Title:** The title of the Newsletter or title of the game.
* **Text:** Information about the Newsletter and about the game that is being talked about.

#### Output:

The Newsletter will be saved to the database as well as where the admin wants to specifically save it.

### Actions

Once the admin saves the Newsletter, it will be saved to the database as well as wherever they wish to save it. Once the Newsletter is saved, the admin will be able to send out emails to the users that are registered to receive Newsletters.

### Pre and Post Conditions

**Pre-conditions:** The admin must have access to create Newsletters, as well as be able to look at the database where the users and their names are listed to receive Newsletters.

**Post-conditions:** If the admin correctly makes the Newsletter, they will be able to send out emails to registered users.

### Validation

The ID of the Newsletter must be a new number in order to have access to sending out emails to registered users.

## EDIT NEWSLETTER



The Newsletter page will allow the administrator to edit the Newsletter of their choice as well as save and update it.

### INPUT & OUTPUT

#### Input:

* **Title:** Select the Newsletter from the drop down arrow, and select the Newsletter you want to edit.

#### Output:

The system will place the original Newsletter with the new Newsletter and save it to the database.

### Actions

### Once the admin saves and updates the Newsletter, it will be saved to the database. Once the Newsletter is saved, the admin will be able to send out emails to the users that are registered to receive Newsletters.

### Pre and Post Conditions

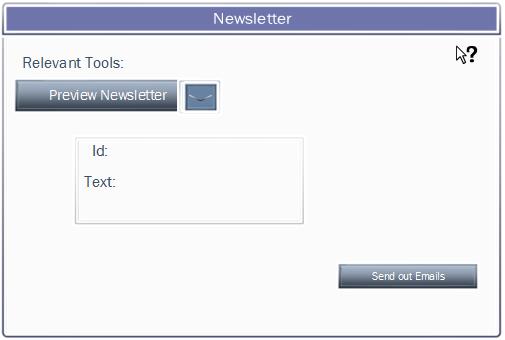
**Pre-conditions:** Admin has access to edit Newsletters.

**Post-conditions:** Successful edit, will replace the old Newsletter with the new Newsletter and save it to the database. Registered users will receive the edited version of the Newsletter.

### Validation

Based on successful completion of editing Newsletter, the old Newsletter will be replaced with the new Newsletter, as well as users will be notified of the edited version.

## PREVIEW NEWSLETTER



The Newsletter page will allow the administrator to preview the Newsletter of their choice before sending out emails.

### INPUT & OUTPUT

#### Input:

* **Title:** Select the Newsletter from the drop down arrow, and select the Newsletter you want to preview.

#### Output:

The admin will be able to select the email list and send out emails.

### Actions

### Once the admin selects a Newsletter from the drop down list, the database will be searched and the Newsletter will be displayed. Once the admin previews the Newsletter, they will select the send out emails tab, and send out emails to registered users.

### Pre and Post Conditions

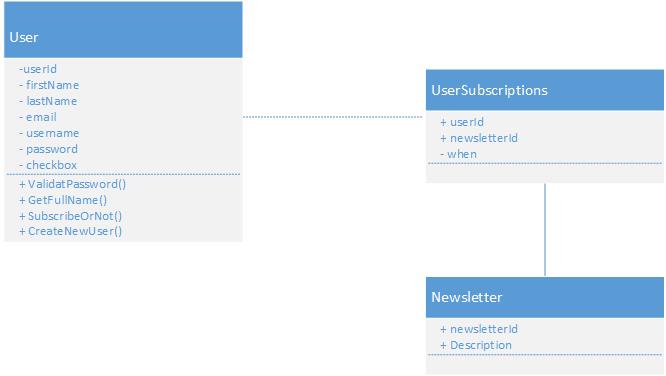
**Pre-conditions:** The admin has access to preview the Newsletters.

**Post-conditions:** Successful submission of Newsletter will allow the admin to preview the final product before sending out emails to registered users.

### Validation

Based on successful completion of previewing Newsletter, the admin will be able to send out emails to the registered users.

# Component Design



## Entities

**User**: The User class represents both user and administrator. It is the base class for all users as well as a base for administrators to receive information and store it in the database. This class is responsible in managing the user personal information.

**UserSubscription**: The UserSubscription class will only have the responsibility to let the admin know which users want Newsletter subscriptions.

**Newsletter**: This class is a database entry with two field, newsletterId and the description of the Newsletter.

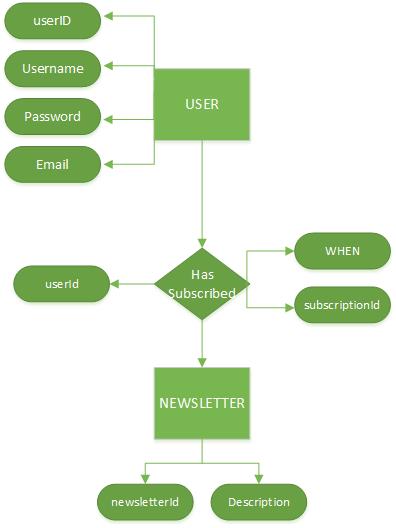
# Data Design

One per module of the following:

## Data Description

## newsletter

From the above diagram, we identified a relationship between the user and their status on being subscribed. The possible types will be user and Administrator. The relationship “hasSubscribed” will also include an attribute to determine when the user subscribed, their subscriptionId, and their userId. The Newsletter will have attributes such as the newsletterId and the description of the Newsletter.



# DATABASE Dictionary / SCHEMA

## Newsletter

The following tables are defined for the News module, Newsletter module, and

|  |  |  |  |
| --- | --- | --- | --- |
| **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(20) | Not null | User’s password stored |
| **first\_name** | varchar(30) | Not null | User’s first name |
| **last\_name** | varchar(30) | Not null | User’s last name |
| **Email** | varchar(30) | Not null | User’s E-mail address |

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_Types Table** |  |  |  |
| **Entity** | **Type** | **Value** | **Description** |
| **user\_type\_id** | INT | Not Null | PK. Unique identifier of a user. |
| **newsletter\_type\_id** | INT | Not null | Newsletter Id |
| **subscribed\_type\_when** | INT | Not null | Date of subscription |
| **type\_desctiption** | varchar(2000) | Not null | Newsletter description |

# Full DAtabase Model

*Full entity relationship for ALL MODULES assigned to your group. No need to include dictionary. This entity relationship diagram should include tables for ALL the use cases presented in the SRD. This is the only section of the document where you will be using info from the fall your use cases.*

***Guidelines:   
Include ALL assigned modules***

- *Include in your diagram*

*- Relationships are well defined*

*- Table names*

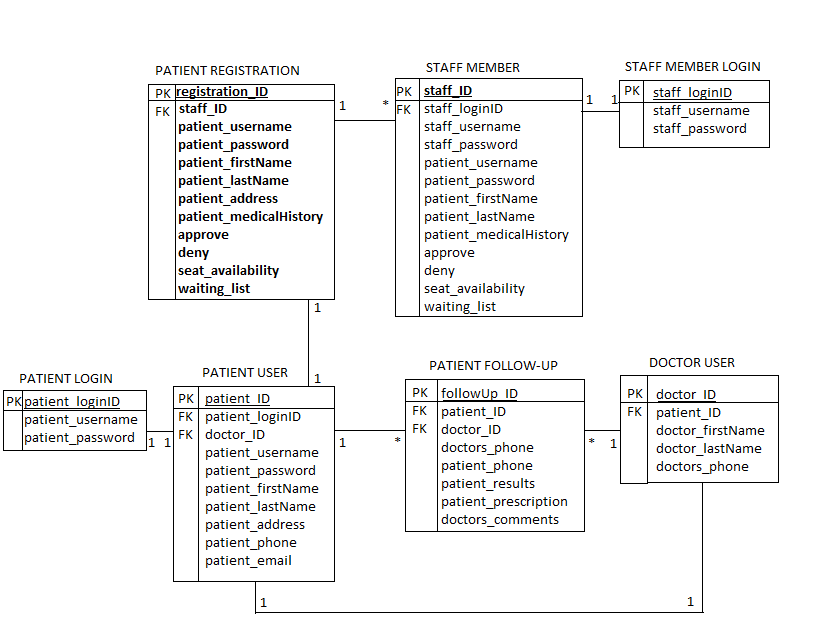
*- Fields*

*- Primary keys (PK)*

*- Foreign keys (FK)*

*- Relationships*

*SAMPLE DIAGRAM(NOT VALID FOR THIS PROJECT)*



# Requirements Matrix

*For this section, simply provide a cross-reference that traces components and tables to your original requirements from your SRS document.*

*Use a structured or table format and identify each requirement with a design component.*

*You can refer to the requirements by Module/Use case or if you use a numeric format in your SRS, simply reference the requirement number.*

*For the component matching the requirement, you can use a page number or section number. For example:*

|  |  |  |
| --- | --- | --- |
| ***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* |

# Appendices

*Include in this section any relevant information to the design of the project that you think will be necessary to the development of the modules.*