1. **Introduction**
   1. Purpose

The purpose of this document is to provide a more in-depth analysis of the UPOD project’s architecture. This will be achieved through the use of UML diagrams and use case diagrams, as well information on user interfaces, models, classes and the connection between them.

Undergraduate Physics Online Database is intended to provide students with a wide variety of knowledge and resources to assist in the learning of physics. The website will provide the essential information for learning physics ranging from kinematics to quantum mechanics.

* 1. Scope

The scope of this project is to design, create, test and maintain the online physics database. This will be accomplished by a variety of teams such as front-end, back-end, SQA and requirements all working together to build a final product.

* 1. Definitions, Acronyms and Abbreviations
* UPOD - Undergraduate Physics Online Database
* WYSIWYG - What You See Is What You Get
* CP317- Software Engineering Course
* HTML - Hyper Text Markup Language  
  1. References

The analysis template is based on the following software specifications template:  
 <http://users.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc>

1. **Functional Requirements**

*This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section*.

* 1. Editor
     1. Introduction

The Editor will allow the user to create posts through a drag-n-drop interface. The user will be expected to drag a block from a sidebar onto the main page and will be able to edit it there. The purpose of this structure is mainly to simplify the process of storing the articles in a normalized manner. That is, instead of storing the articles as one large text blob; the article will be split into different blocks in the database - all images will be stored together, all equations will be stored together, etc.

This structure was selected to emphasize performance, maintainability and ease of use. Performance will be improved as the most intensive process would be extracting the individual elements out of the large string. The block approach allows the application to receive the individual elements and thus, no extraction is required. Maintainability is emphasized as the large string approach would be mainly dependent on a 3rd party WYSIWYG editor where the output could change without notice. If the string approach was selected, an update to that editor could break the entire application. Additionally, future developers succeeding the current CP317 class would be able to extend the database and functionality more simply. Lastly, the Ease of Use would be simplified as the users will not be expected to write HTML or Markdown syntax.

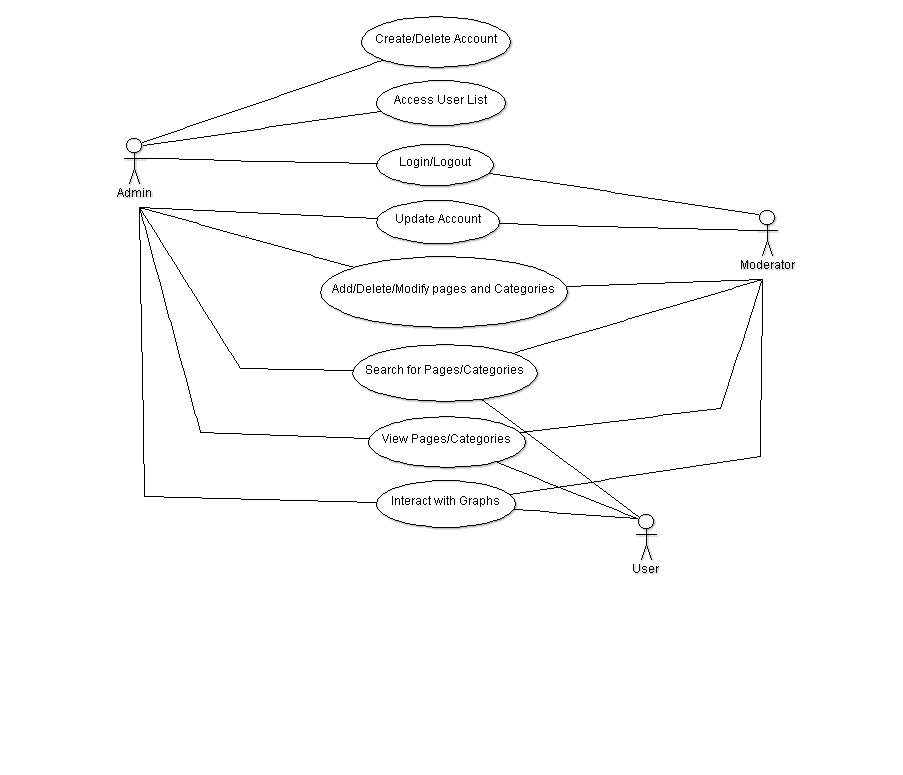
* + 1. Inputs
    2. Processing
    3. Outputs
  1. <Functional Requirement or Feature #2>
     1. Introductions
     2. Inputs
     3. Processing
     4. Outputs

1. **Non-Functional Requirements**

*Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms. (****Here are some Examples)***

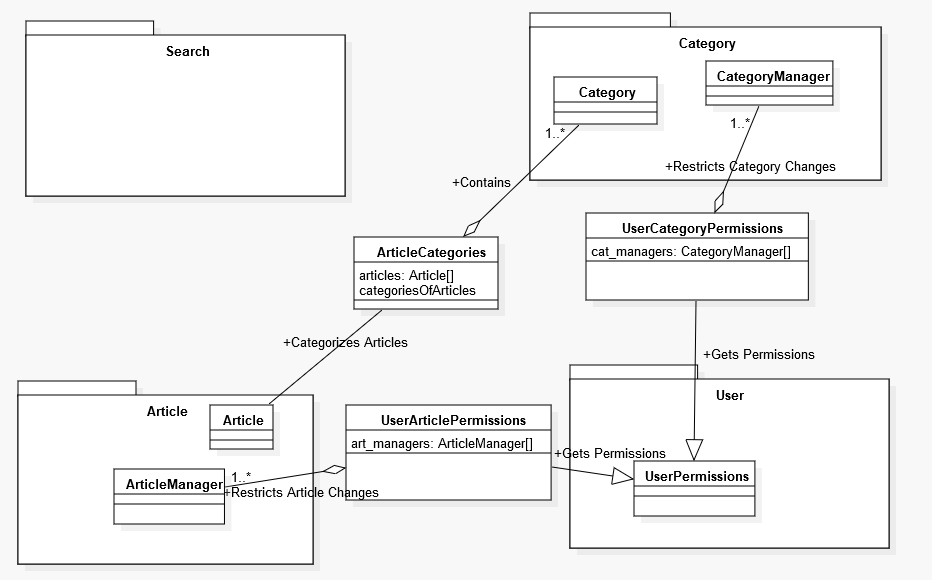
* 1. Performance
  2. Reliability
* Database provides up-to-date, accurate information.
* No duplicate information is stored in the database.
* The search engine should provide a list of articles that are contains the keywords or are related to the key words.
  1. Availability
  2. Portability
  3. Security
* Account passwords of administrators and moderators accounts should be stored securely.
* Without appropriate permissions, no articles should be added, removed, or edited.
  1. Maintainability
* Additional features may be added to the website in the future, so the whole software should be well-documented and easy to maintain.

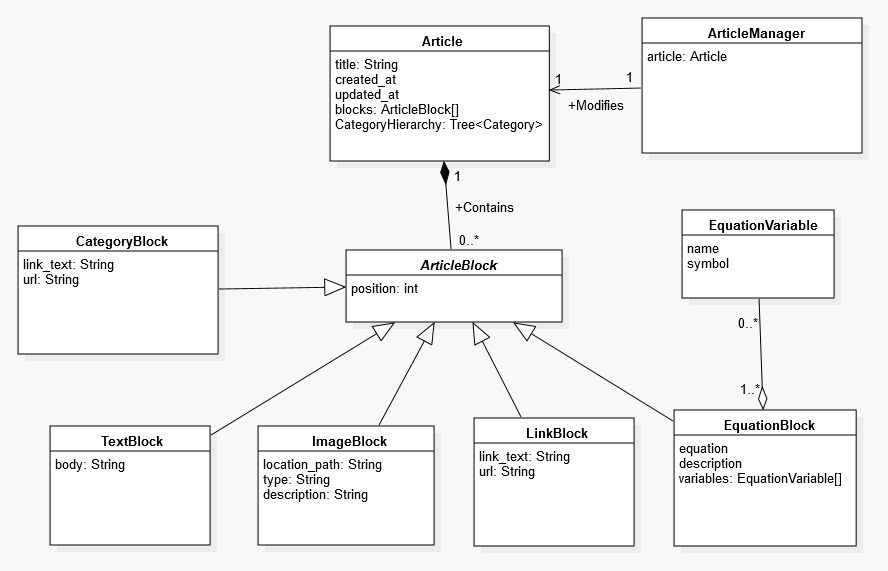
1. **Use Cases**
   1. Front End Use Cases
   2. Back End Use Cases



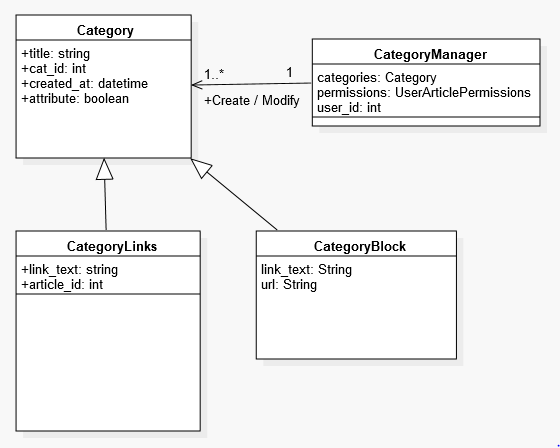
* 1. SVG Diagram Use Cases

1. **UML Diagrams**
   1. Front End UML Diagrams
   2. Back End UML Diagrams

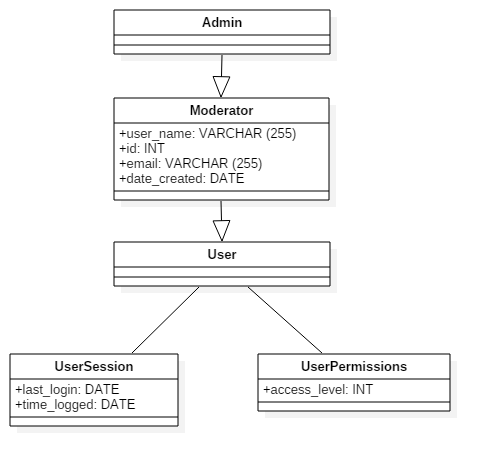
**Relationship Between Back-end Functions**

**Article Class**

**Category Class**



**User Class**



* 1. SVG UML Diagrams

1. **Appendices** (If Required)