## **IT6037 Project - System Design**

Group: Dominic Martindale & Matthew Dods

## Architecture:

### Possible Database Management Systems

**Name: MySQL**

**About:** MySQL is the world's most popular open source database, it is maintained by Oracle, initially released in 1995 and is a RDBMS. Quick Tip; Did you know that “*If a PRIMARY KEY or UNIQUE index consists of only one column that has an integer type, you can also refer to the column as “\_rowid” in SELECT statements.*” - bigdata.com (guest, circa. 2015)

**Pros:** Its free, can work easily with other databases.

**Cons:** No automatic incremental backups (you have to code that yourself), no built in XML support.

**Name: MongoDB**

**About:** Made by the organisation called MongoDB Inc., MongoDB was initially released in 2009, with its key feature being JSON-like schemas, it is licensed under the Server Side Public Licence (SSPL).

**Pros:** NoSQL, easy to manipulate/make your own schemas, direct integration with AWS, Azure and google cloud

**Cons:** NoSQL(really depends on your perspective!), *“Data consumption is generally high due to de-normalization.” - Scott McNulty (*[*quora*](https://www.quora.com/What-are-the-disadvantages-of-MongoDB)*, circa. 2019)*

**Name: MicrosoftSQLserver**

**About:** Made and maintained by the one and only microsoft, first version was released in 1989, it is proprietary software, meaning you have to buy it.

**Pros:** Fast and stable, built in tools to help reduce resource use, part of the microsoft ecosystem. Has been around for a long time, plenty of information on it out there

**Cons:** (the enterprise version is) **Really, really expensive** ([see here](https://redmondmag.com/articles/2019/11/08/sql-server-2019-licensing.aspx), $7128 USD/core)

**Conclusion with reasoning:**

Due to our group’s collective expertise and knowledge and due to MySQL being open source, free and based on relationships with entities, we have decided that it would be the most suitable DMS(database management system) to use in the project for our clients needs.

Sources:

<https://en.wikipedia.org/wiki/MySQL>

<https://www.mysql.com/about/>

<https://bigdata-madesimple.com/10-interesting-facts-and-tips-about-mysql/>

<https://www.keycdn.com/blog/popular-databases>

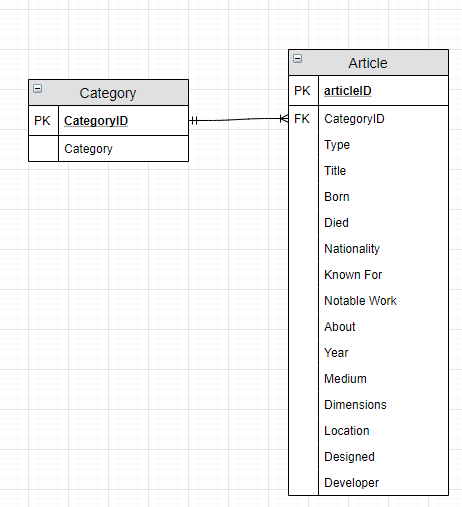
<https://en.wikipedia.org/wiki/Microsoft_SQL_Server>

<https://redmondmag.com/articles/2019/11/08/sql-server-2019-licensing.aspx>

<https://www.quora.com/What-are-the-disadvantages-of-MongoDB>

## Database Design:

**Data Model:**



## Queries as per the requirements and planning documents:

* **Browse articles by category:**

**SELECT** Article.articleID, Article.Title, Category.Category

**FROM** Category

**INNER JOIN** Article

**ON** Category.CategoryID = Article.CategoryID

**WHERE** Category.Category = ‘Wow amazing category name’

* **Browse articles by a keyword in the title:**

**SELECT** \*

**FROM** Article

**Where** Article.Title **LIKE** ‘%keyword%’

Or

**SELECT** \*

**FROM** Article

**WHERE LOCATE**(Article.Title, ‘keyword’) != 0

Note: above queries need testing before use.

## User Access Design:

**Advice on how user access design should be handled:**

* **Passwords for users should be enforced in a way so that they are (reasonably) secure.**
  + E.g. When users are creating their passwords, it is required through checking via an algorithm that it at least contains at least one of each of the following: an upper-case letter (A-Z), a lower-case letter (a-z), a number (0-9) and a special character (E.g. ; , . ‘ “ [ ] … ).
* **Passwords are not stored in cleartext in the database they are stored on.**
  + E.g. Appropriately complex encryption is used on the stored passwords (Currently used standard as of 2020: AES)
* **Sanitize data on all user inputs**
  + E.g. Wherever there is an input that the user can use, filter the input so that it cannot be used in a potentially malicious manner (for text inputs, this would be restricting which special characters can be used; no ‘,”,; … which could be used for SQLinjection attacks)

List of users and their permissions:

|  |  |
| --- | --- |
| **Role** | **Permissions** |
| Student | View articles and browse them using keywords or category. |
| Tutor | Add or modify existing articles. |
| Admin | Add, modify, and delete existing articles. |

## User Experience Design:

Some views to implement on the overall mobile application are:

* Login View (difference in access between Students, Tutors, and Admins.)
* View to manage articles (add, modify, delete depending on permissions.)
* View browse articles by category or name/keywords.

## Implementation and Testing:

To implement user access into MySQL Workbench follow the steps below:

* Go into your open database and click on the server dropdown button on the bar at the top.
* Click Users and Privileges.
* Click Add Account.
* Enter the User Data (login and password) as you please.
* Go to Administrative Roles.
* Apply the permissions you want this user to have (either using one of the roles on the left or you can apply a specific set of permissions on the right).
* Click apply in the bottom right.
* Done!