**Solution Design Document**

IT6037 Data Access and Management

Team: SWESLBS

Architecture:

* *Investigate possible database management systems appropriate to meet the requirements and select one. Justify your selection.*

Possible database management systems

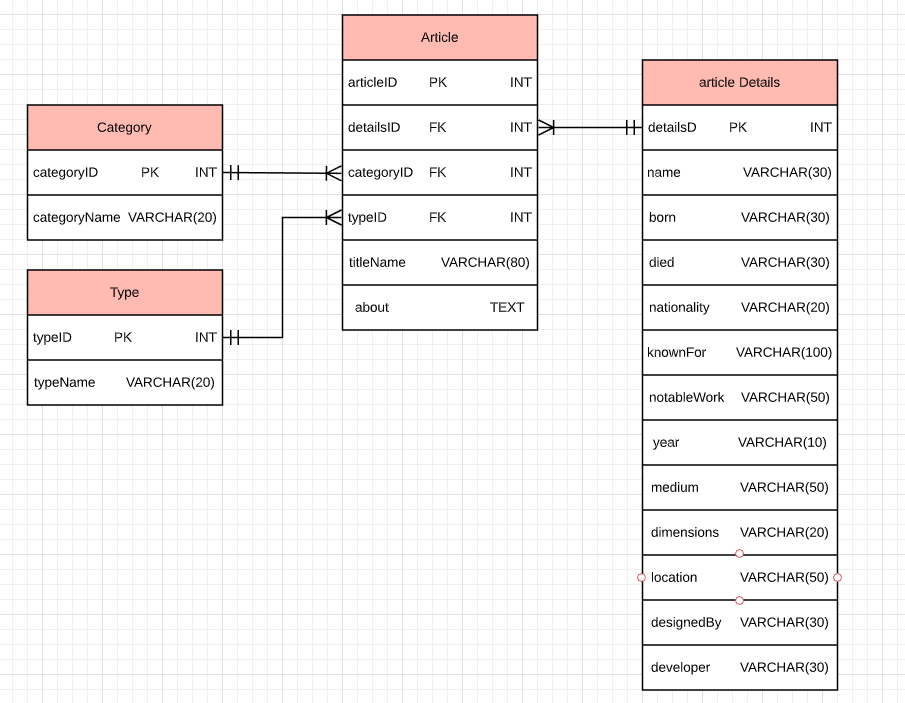
* MySQL
* MongoDB

Chosen database management system: MySQL

We have chosen to go with the MySQL database management system, as it is far more efficient and easier compared to MongoDB. Since the requirements for this project is for a school, MySQL would be easier as it is an open source, meaning the school doesn’t have to pay a lot of money to regulate it. MongoDB is also open source. However, it is quite limited, and you’d have to pay for those extra features, the default setting for MongoDB is unsecure and it not changed (set up correctly) could be easily hacked, MySQL is popular and well-established, almost any problem that you encounter will be answered online, often by multiple people and in detail. Since this database is used for searching/modifying and deleting articles, MySQL would be better as it is a relational database, meaning you can use queries to find what you are looking for fast and easy.

Database Design:

* *Design data model. Identify main entities, attributes, data types and relationships.*



Many categories are in a single article

Many types are in a single article

A single article, can contains a single or many article detail

* *List appropriate queries as per the requirements and planning documents.*

1. Browse articles by category

Example:  
CALL searchArticleByCategory(‘Arts’);

DELIMITER //  
CREATE PROCEDURE ‘searchArticleByCategory’(cat VARCHAR(20))  
BEGIN  
SELECT c.categoryName, a.titleName, a.about  
FROM category c  
INNER JOIN article a  
ON C.categoryID = a.categoryID  
WHERE categoryName LIKE cat;  
END  
  
DELIMITER;

1. Browse articles by a keyword in the title

Example:  
CALL searchArticleByKeyword(‘%....%’);

DELIMITER //  
CREATE PROCEDURE ‘searchArticleByKeyword’(keyw varchar(80))  
BEGIN  
SELECT d.titleName, d.about  
FROM articleDetails d  
WHERE d.titleName LIKE keyw;  
END  
  
DELIMITER;

1. Add article. To add an article, use this format:

INSERT INTO Article Values (articleID, detailsID, categoryID, typeID, titleName, about);

Example:

INSERT INTO Article VALUES (1, 1, 1, 1, ‘Biography of Claude Monet’, ‘Oscar-Claude Monet was a French painter, a founder of French Impressionist painting and the most consistent and prolific practitioner of the movement\'s philosophy of expressing one\'s perceptions before nature, especially as applied to plein air landscape painting. The term "Impressionism" is derived from the title of his painting Impression, soleil levant (Impression, Sunrise), which was exhibited in 1874 in the first of the independent exhibitions mounted by Monet and his associates as an alternative to the Salon de Paris. Monet\'s ambition of documenting the French countryside led him to adopt a method of painting the same scene many times in order to capture the changing of light and the passing of the seasons. From 1883, Monet lived in Giverny, where he purchased a house and property and began a vast landscaping project which included lily ponds that would become the subjects of his best-known works. In 1899, he began painting the water lilies, first in vertical views with a Japanese bridge as a central feature and later in the series of large-scale paintings that was to occupy him continuously for the next 20 years of his life.’);

1. Modify articles. This depends on what information you want to modify. Add what column you want to modify in the SET part and add a specific value that the row has in the WHERE part.

Example:

UPDATE Article

SET titleName = ‘Biography of Claude Monet’

WHERE articleID = ‘1’;

1. Remove article

DELETE FROM Article  
WHERE articleID = ‘1’;

Link to all queries in text file:

<https://docs.google.com/document/d/1B6xL4RKXHa7gICYACmBsSv2Yw9O08Kd15DDm5pz2OVc/edit?usp=sharing>

User Access Design:

* *Advise on how the user security requirements should be implemented regarding user authentication and authorization. Use security compliance and secure coding checklists as guidance (“security by design”).*
* All access to the database should be properly authenticated. (Use the DBMS authentication functions.)
* Limit access to data using the most restrictive settings, default access is no access, clients who need to see the data should have read-only access, only give write access to those clients who really need it. (Use the DBMS access control or permissions function, different setting for different functions.)
* The application should use the lowest possible level of privilege when accessing the database.
* Use stored procedures to abstract data access and allow for the removal of permissions to the base tables in the database.
* Remove or change all default administrative passwords. Utilize strong passwords/phrases or implement multi-factor authentication.
* The application should connect to the database with different credentials for every trust distinction (users, read-only users, administrators).
* Use only trusted system objects, sever side session objects, for making access authorization decisions.
* Use a single site-wide component to check access authorization.
* Enforce authorization controls on every request.
* Restrict access to protected functions to only authorized users.
* Restrict direct object references to only authorized users.
* Restrict access to services to only authorized users.
* Restrict access security-relevant configuration information to only authorized users.
* If long authenticated sessions are allowed, periodically re-validate a user’s authorization to ensure that their privileges have not changed and if they have, log the user out and force them to re-authenticate.
* The application must support disabling of accounts and terminating sessions when authorization ceases.
* All authentication controls should fail securely.
* Enforce password complexity requirements. Authentication credentials should be able to withstand attacks that are typical of the threats in the deployed environment.  
  Re-authenticate users prior to performing critical operations

*Define user roles and permissions.*

Students (read only)

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

Tutors (add/update)

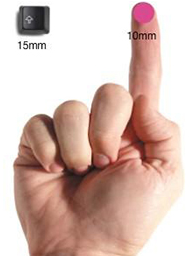
* Add or modify articles

Administrators (add/update/delete)

* Add, modify or remove articles

User Experience Design:

* *Suggest views required for the overall mobile application.*

1. Prioritize features: Focus on refining the experience around the core objectives, trying to add as many features as possible rarely results in better user experience.
2. Cut out the clutter: Cluttering a user interface overloads your user with too much information.  
   Strive for minimalism, focus on the content that is valuable for your users.  
   Prioritize one primary action per screen, try to design each screen for one thing and one thing only.
3. Make navigation self-evident: Helping users navigate should be a top priority for mobile applications.  
   Make sure the navigation feels familiar to users.  
   Design good information architecture.  
   Navigation shouldn’t draw focus away for the content.  
   Communicate user's current location on the application.  
   Strive for consistency.  
   Provide a clear path
4. Design finger-friendly tap-targets: Create controls that measure at least 7-10 mm so they can be tapped easily with a finger.  
   
5. Make text legible: Choose a typeface that works well in multiple sizes and weights to maintain readability and usability in every size.
6. Use legible font size, text should be at least 11 points so user can read it without zooming.
7. Use appropriate colour for text
8. Provide feedback on interactions: Feedback acknowledges actions and help users understand the results of operations.  
    Feedback helps users know what the application is doing now.
9. Minimize the need for typing: Ask only what you really need to know.
10. Make data entry as efficient as possible.