**System Design Document**

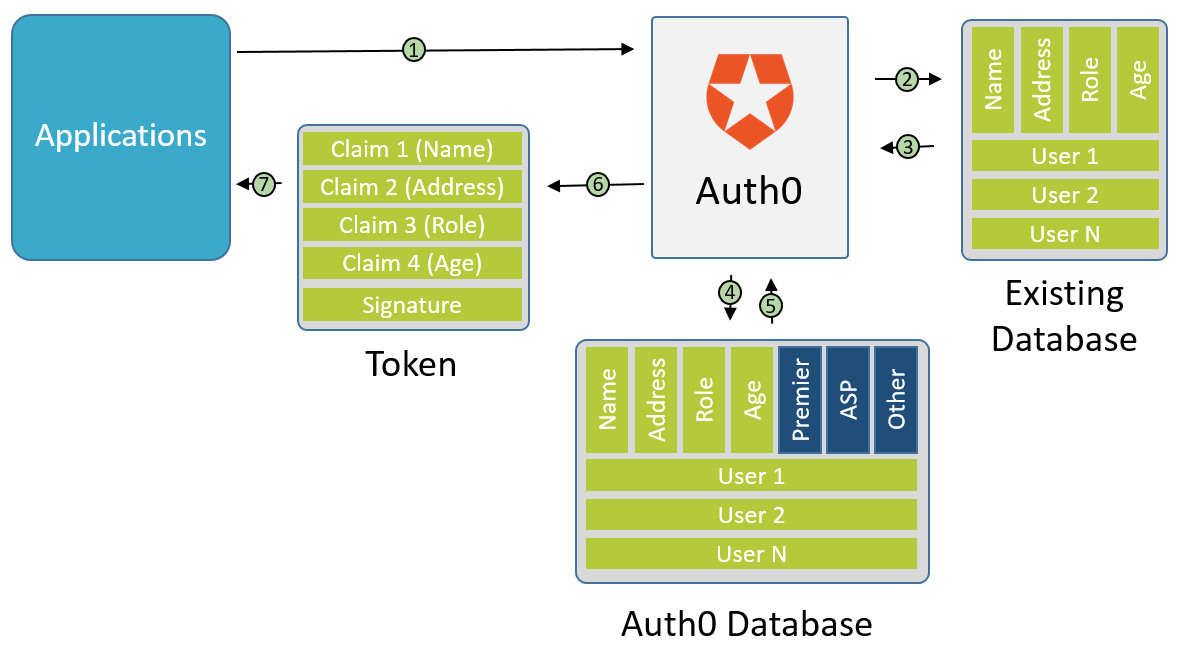
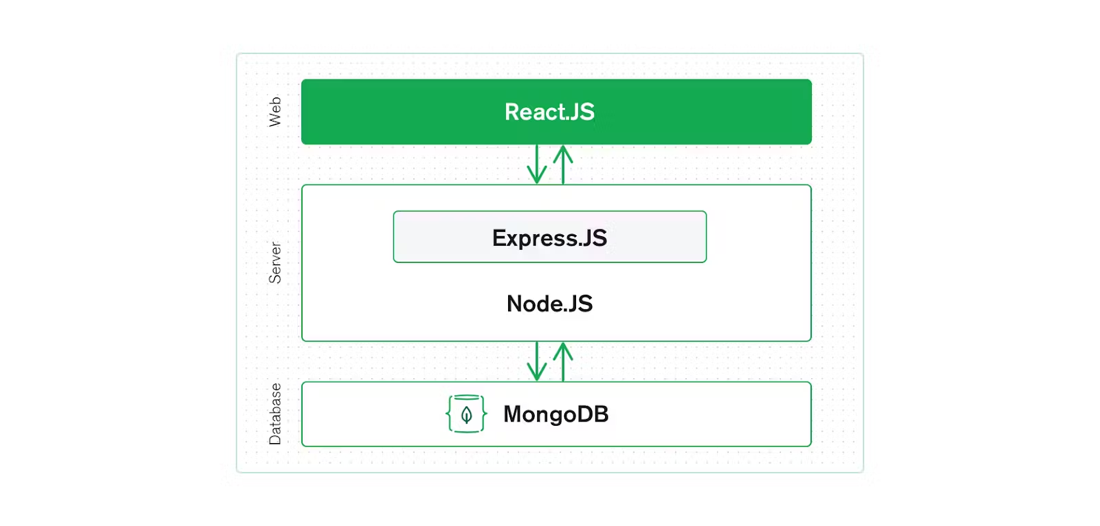
EduArchive

**Team Data Access Geeks**

# Architecture Overview

For our development framework, we will be adopting the MERN tech stack. This includes the following elements, with the inclusion of some additional features:

* **Backend:** 
  + **MongoDB Atlas** is a cloud based, NoSQL document-oriented database that utilises a JSON format to store its data. It is well suited for an educational archive web application given its reliability in storing unstructured and adaptable data, and its horizontal scalability ensures that it can keep up with the growing user traffic and database collection as more articles are created and stored.
* **Server:**
  + **Node** environment will be utilised as the core of our data structure, providing our developers with the tools needed to create a JavaScript
  + **Express** is a Node JavaScript framework that will provide us with a broad set of tools and features when building our MERN application.
  + **Mongoose** is an object-data modelling library for MongoDB that both facilitates developer interaction with their database and creates a connection between the Node application and the MongoDB database.
* **Frontend:** 
  + **React** will be used for building the user interface and client-side functionality of the web application. This should allow us to take a more rapid approach to the development of the user interface and allows us to easily create and reuse modules/components throughout the application.
  + **Vite** is a more efficient than the standard react application, as it will allow a faster development process, with an optimised build that will decrease the size of the application and enhance its performance.
  + **Tailwind** is a CSS styling approach that works well with React applications.
* **Authentication and Authorisation:** 
  + **Auth0:** we will utilise a third-party server that will open a new window when users interact with the application. User accounts and data will be stored within an auth0 database and user roles/permissions will be set using this application.

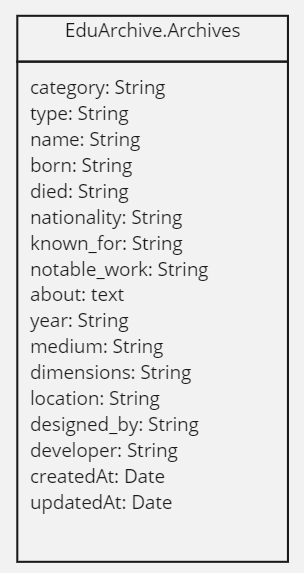


Database Design

**Data Model / Entity Relationship Diagram**

We have decided to adopt a single collection approach to this application, as the current iteration of the application has a simple and limited scope. If the client decides to grow its database or functionality in the future, we can update this into separate collections to enable more flexibility in a later iteration. User data will be handled and stored by Auth0, and thus there will be no relationship between article attributes and user data, as all attributes can be viewed by all user types.

MongoDB Collection:



**Queries**

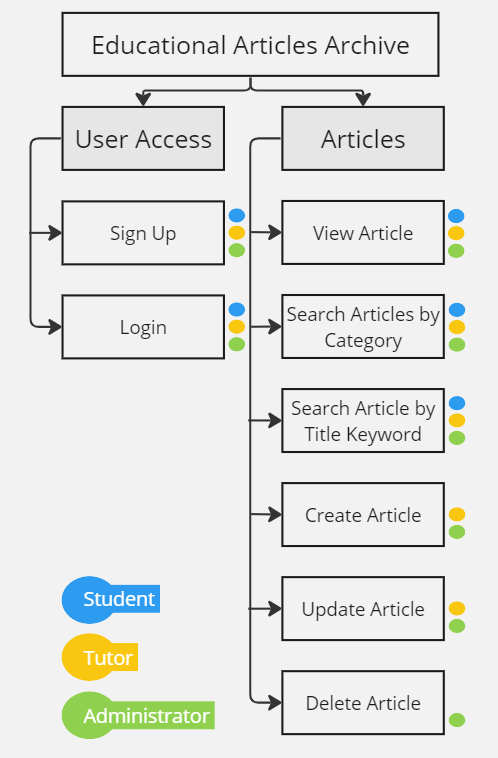
* Create USER ACCOUNT
  + Student
  + Tutor
  + Administrator
* Log in USER
  + Student
  + Tutor
  + Administrator
* Create new ARTICLE
* Select / view ARTICLE
* Update ARTICLE
* Delete ARTICLE
* Filter results by category: ARTICLES
  + Art
  + Technology
  + Mathematics
* Filter results by keyword input within the title/name parameter: ARTICLES

**Security Design considerations**

1. **Authentication**
   * MongoDB requires a username and password to authenticate any user accounts in their database, so we will include a system where both parameters are required to log in.
   * MongoDB has no built-in password complexity requirements, so we will include our own password requirements that users will need to consider when creating their passwords. This will most likely specify a minimum character requirement, and the use of numerals, special characters, and/or variety of lower and upper letter characters.
2. **Authorisation**
   * Our application will require diverse user permissions depending on the user account type that has been logged in. We will need to create separate user creation forms or include a feature within the user creation form to grant an account certain a user role with associated permissions.
3. **Error Handling**
   * We will follow standard error handling conventions and ensure that the correct error code is displayed to the user when an issue arises.

# User Access Design

**Functional Decomposition diagram**



**User Interface List / View Design:**

GUEST:

1. Guest user selects ‘sign up’ to create a user account.

STUDENT:

1. Student selects ‘log in’ to access articles in database with restricted permissions.
2. Student selects an article title to open and read the corresponding article.
3. Student selects category filter option (art, technology, mathematics) to show articles contained within that category.
4. Student clicks on search bar and inputs a keyword(s) to find articles with that keyword(s) included.

TUTOR:

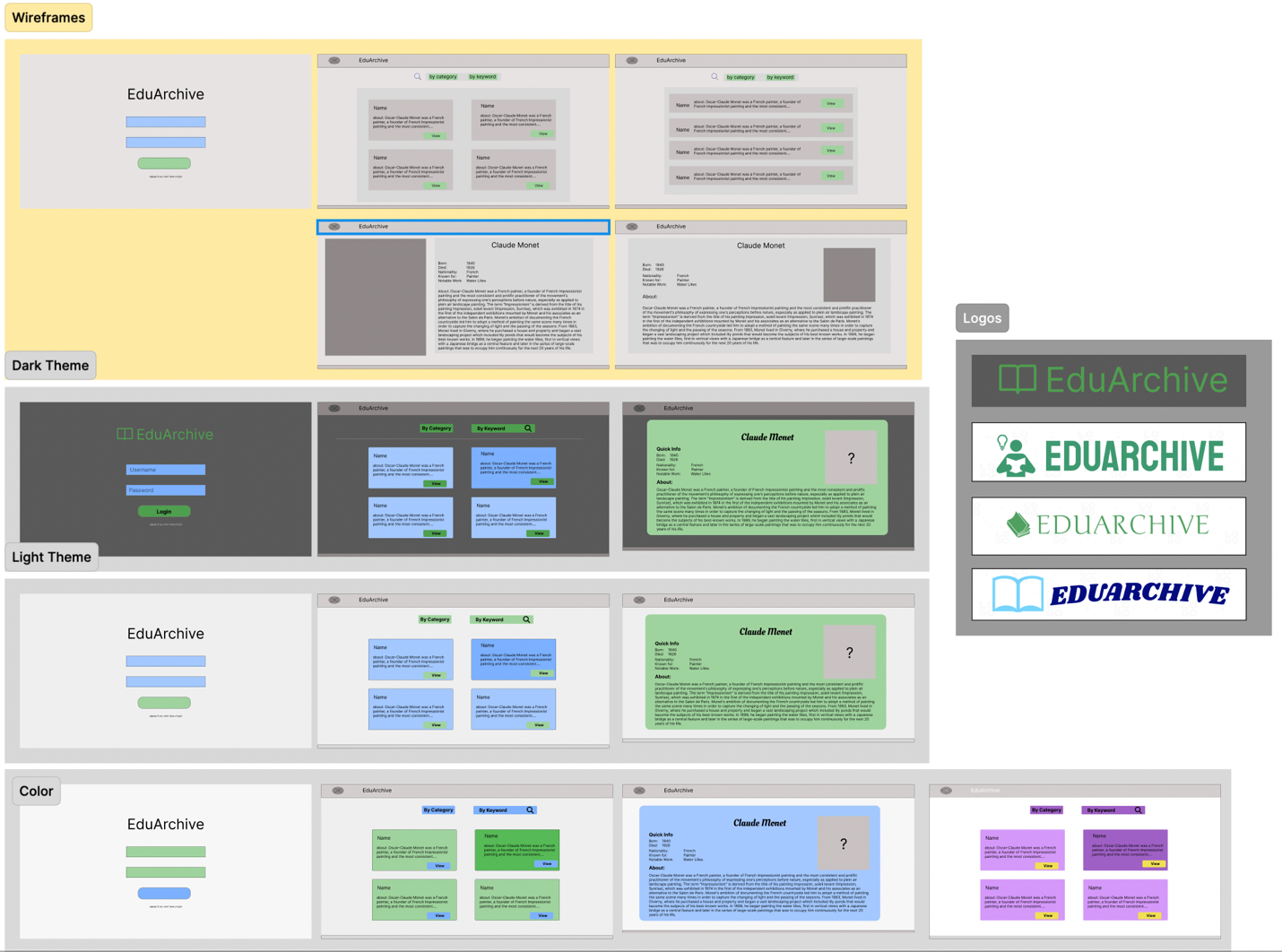
1. Tutor selects ‘log in’ to access articles in database with restricted permissions.
2. Tutor selects an article title to open and read the corresponding article.
3. Tutor selects category filter option (art, technology, mathematics) to show articles contained within that category.
4. Tutor clicks on search bar and inputs a keyword(s) to find articles with that keyword(s) included.
5. Tutor selects ‘create article’ to create a new article and input details.
6. Tutor selects ‘update article’ to update/edit the details within a preexisting article.

ADMINISTRATOR:

1. Administrator selects ‘log in’ to access articles in database with full permissions.
2. Administrator selects an article title to open and read the corresponding article.
3. Administrator selects category filter option (art, technology, mathematics) to show articles contained within that category.
4. Administrator clicks on search bar and inputs a keyword(s) to find articles with that keyword(s) included.
5. Administrator selects ‘create article’ to create a new article and input its information.
6. Administrator selects ‘update article’ to update/edit the details within a pre-existing article.
7. Administrator selects ‘delete article’ to delete a pre-existing article.

# User Experience Design

**Wireframes**

****

Above is the different wireframes, different color themes and logos for the front end of the application.

**Final Design:**

