**System Design Document**

**Project Name**

**Team Name: Admirable Blue Beatles**

**ARCHITECTURE**

MongoDB is a no-SQL database. Instead of tables and rows as in a relational database, MongoDB is made of collections and documents.

Documents are accessed through key-value pairs and are stored as Binary JSON (BSON). Each document has different fields, and the value of a field can be any of the BSON data type. Each document will also have a primary key, which must be unique and immutable.

Documents are then organized ins sets called collections. Collections are analogous to tables in relational databases.

MongoDB has an auto-sharding feature to horizontally shard databases to distribute data in a way that is efficient.

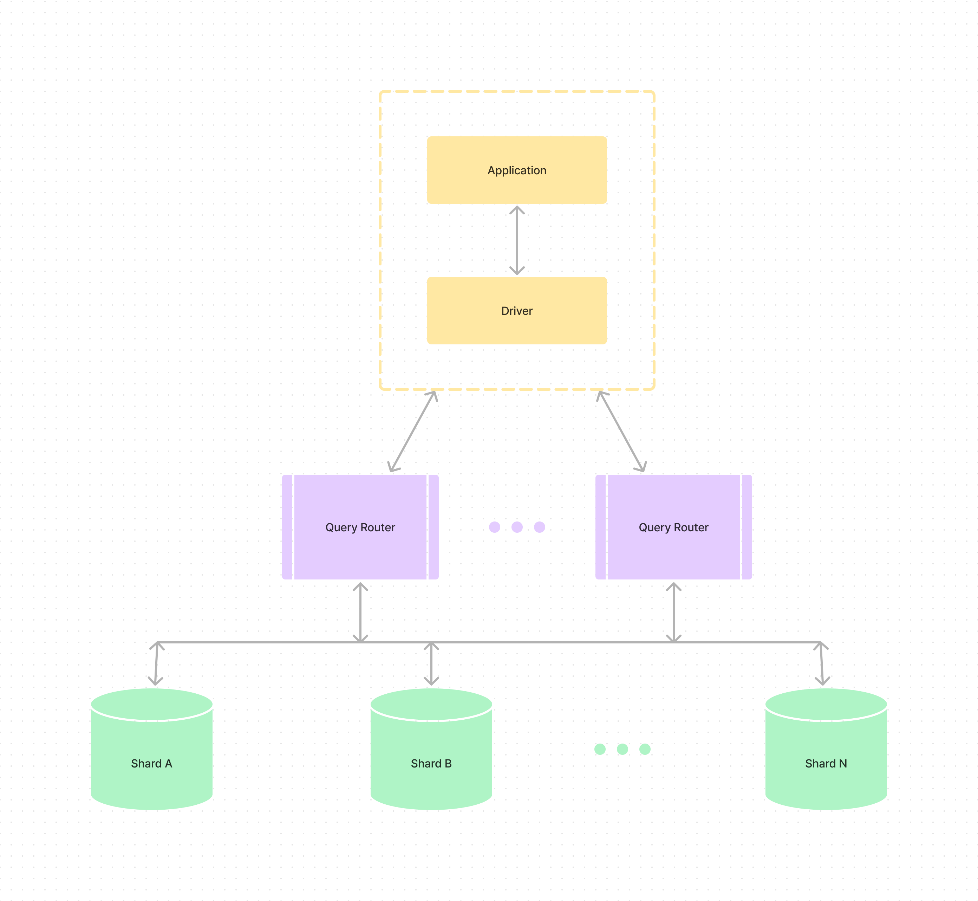


Figure : Diagram of MongoDB Arcitecture

**WEB FRAMEWORK**

#### **User Interface**

This is the front end of the web application, where users access the database, query the database and sign in.

#### **Client-side components**

* User Authentication
  + Where users can sign into the database.
* Search Interface
  + Where users can query the database
* Document Interface
  + Where users can read the document they queried.

#### **Serve side components**

* Server
  + Handles user request and communicates with the database, where the MongoDB database is hosted.
* Database
  + Stores the data in document collections.

**DATA ACCESS DESIGN**

**MODEL DESIGN (DATA MODEL)**

In MongoDB, the data model revolves around collections and documents. For our educational platform:

**Collections:**

* 1. **Users:**
  + UserID (Primary Key)
  + Username
  + Password
  + First\_name
  + Last\_name
  + Age
  + Gender
  + Email
  + Id (for student is roll number and for staff and admin it’s their official id )
  + UserType (Student, Tutor, Administrator)
  1. **Articles:**
  + ArticleID (Primary Key)
  + Title
  + Content
  + CategoryID (Foreign Key referencing Categories)
  + UserID (Foreign Key referencing Users)
  1. **Categories:**
     + CategoryID (Primary Key)
* CategoryName (Art, Mathematics, Technology)

**Relationships:**

* Users and Articles have a one-to-many relationship (One user can contribute multiple articles).
* Articles and Categories have a many-to-one relationship (An article can belong to one category).

**FUNCTIONAL DECOMPOSITION**

**User Authentication:**

* Implement user authentication mechanisms, including sign-in and sign-out functionalities.

**Article Management:**

* Enable users to browse articles by category and keyword in the title.
* Allow tutors to add or modify articles.
* Empower administrators to add, modify, or remove articles.

**Database Interaction:**

* Develop server-side components to handle user requests and communicate with the MongoDB database.
* Implement functions for storing, retrieving, and modifying data in the database.

**FOLDER STRUCTURE**

* admirable blue beatles
  + controllers
    - article.js
    - auth.js
  + db
    - connectdb.js
  + middleware
    - auth.js
    - permission.js
  + models
    - schems.js
    - user.js
  + routes
    - routes.js
  + scripts
    - scripts.js
  + utils
    - sampleData.js
  + views
    - partials
      * header.ejs
      * footer.ejs
    - index.ejs
    - login.ejs
    - signup.ejs

**SECURITY DESIGN**

**FRAMEWORK SECURITY**

The security features that MongoDB comes with includes:

* Authentication
* Authorization
* Auditing
* Data Encryption
* Network Security
* Data Sovereignty

**SECURITY MECHANISMS**

**Authentication and Authorization**

* In Atlas, you configure database users to access your deployments. Atlas provides various ways to perform user authentication and authorization, including LDAP, OIDC, and X.509.

**Encryption**

* Atlas encrypts all data stored in your deployments and uses TLS/SSL to encrypt the connections to your databases.

**IP Access List**

* Atlas allows connections only from addresses specified in the IP access list.

**Cloud Provider Support**

* Atlas supports network peering connections and private endpoints to secure your deployments hosted on AWS, Azure, and Google Cloud.

**USER ROLES AND PERMISSIONS**

**Student User**

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

**Teacher User**

* Add articles
* Modify articles

**Administrator User**

* Add articles
* Modify articles
* Delete articles

**USER INTERFACE DESIGN**

**VIEW DESIGN**



**User Stories**

*As a student, I want to be user-friendly and engaging, so that I can easily navigate and access information related to art, mathematics and technology for my learning.*

*As a student, I want to be able to search for information in order, so that I can quickly find information on a particular topic.*

*As a student, I want a functional and mobile-friendly system, so I can access educational materials on the go from my smartphone or tablet.*

*As a student, I would like to have personalized suggestions for content based on my interests and past interactions with the platform, so that I can easily find context.*

*As a tutor, I want to create an easy-to-use interface to add and edit text, so that I can efficiently populate the database with relevant educational content.*

*As a tutor, I want a flexible exchange system, including the ability to add graphics and multimedia materials, so that I can enhance the teaching content for students.*

*As a tutor, I want to be able to review and approve student submissions before they are published, so that I can ensure the accuracy and appropriateness of the content.*

*As an administrator, I want to have a secure and efficient user management system, so that I can easily add, change, or remove users and control access to the application.*

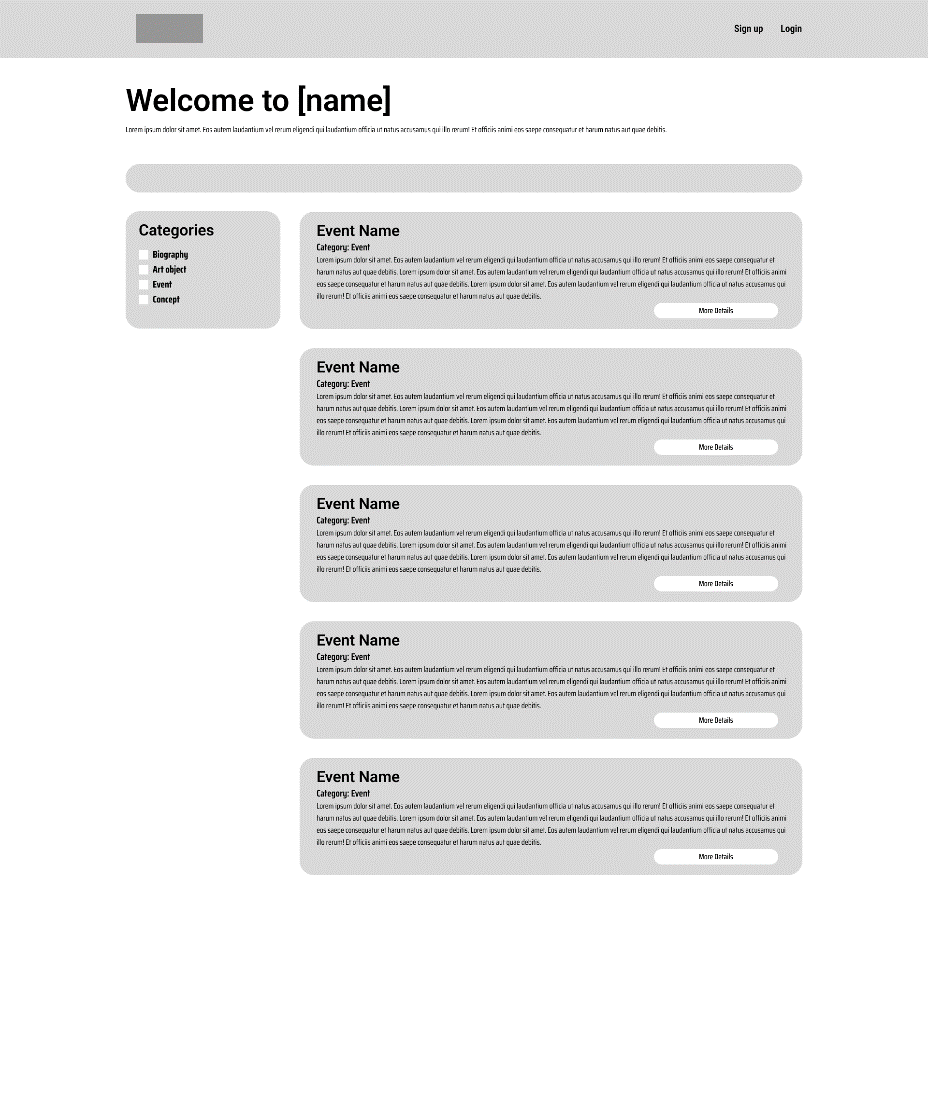
*As an administrator, I want a comprehensive dashboard of system analytics and usage metrics, so I can monitor application performance and user engagement.*

*As an administrator, I want an easy and efficient way to back up and restore a database, so that I can protect the integrity of the data and retrieve information when needed.*

**ELEMENTS**



**WIREFRAMES**



**MOCKUPS**

