# **My Research and development ( R & D ) to website creation**

Phase - 01

Database technology used

1. MongoDB

Storage technology used

1. AWS , Hostinger

**Summary of findings. :-**

**Summary of Findings: MongoDB with AWS and Hostinger**

**1. Database Technology: MongoDB**

**Advantages:** MongoDB is a NoSQL database known for its flexibility in handling unstructured and semi-structured data, making it suitable for modern applications requiring dynamic schemas and scalable architecture.

**Features:** Offers horizontal scalability through sharding, supports high availability with replica sets, and provides flexible querying using JSON-like documents (BSON format).

**Use Cases:** Ideal for applications such as content management systems, real-time analytics, data management, and mobile app backends where flexible data models and scalability are critical.

**2. AWS (Amazon Web Services) - Storage technology :**

**Advantages:** AWS provides a robust cloud infrastructure supporting MongoDB deployments through services like Amazon EC2 (for virtual servers), Amazon S3 (for object storage), and Amazon EBS (for block storage).

**Managed Services:** AWS offers MongoDB-compatible managed database services such as Amazon DocumentDB (compatible with MongoDB) and MongoDB Atlas (offered by MongoDB Inc. as a fully managed service on AWS).

**Scalability and Flexibility:** AWS supports elastic scaling of MongoDB clusters based on workload demands, ensures high availability, and integrates with other AWS services for comprehensive application architecture.

**3. Hostinger Hosting Services:**

**Advantages:** Hostinger provides affordable and scalable hosting solutions suitable for hosting MongoDB applications, offering various plans with SSD storage, dedicated resources, and easy scalability.

**Infrastructure Support:** Hostinger supports MongoDB deployments through VPS (Virtual Private Server) and cloud hosting options, ensuring reliable performance and uptime for applications.

**Ease of Deployment:** Hostinger's control panel and management tools simplify the deployment and management of MongoDB instances, catering to developers and businesses of different sizes.

**4. Performance and Cost Considerations:**

**Performance:** MongoDB on AWS with Hostinger benefits from high-performance infrastructure, low-latency network connectivity, and optimized storage options (e.g., SSDs) for efficient data operations.

**Cost Efficiency:** Both AWS and Hostinger offer cost-effective pricing models, allowing businesses to scale MongoDB deployments based on actual usage and avoid upfront infrastructure investments.

**Operational Efficiency:** Integration with AWS services and Hostinger’s management tools enhances operational efficiency, automates backups, and provides monitoring capabilities to ensure smooth MongoDB operations.

Here's a comparison of AWS, Hostinger, and DigitalOcean in table format:

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **AWS** | **Hostinger** | **Digital Ocean** |
| **Service Type** | Cloud computing platform | Web hosting and domain services | Cloud infrastructure and hosting |
| **Pricing Model** | Pay-as-you-go (varies by service and usage) | Fixed monthly plans | Pay-as-you-go (based on usage and resources) |
| **Free Tier** | 12-month free tier with limited resources | Limited free trial period for certain plans | 60-day free trial with $200 credits for new users |
| **Compute Options** | EC2 (Elastic Compute Cloud) with extensive configurations | Shared, VPS, and cloud hosting options | Droplets (VPS) with customizable configurations |
| **Storage Options** | S3 (Simple Storage Service), EBS (Elastic Block Store) | Shared storage with hosting plans | Block Storage and Spaces (object storage) |
| **Database Options** | RDS (Relational Database Service), DynamoDB, Aurora | Managed databases (MySQL, PostgreSQL) | Managed Databases (PostgreSQL, MySQL, Redis) |
| **Support** | 24/7 support with various plans, extensive documentation | 24/7 support with various plans | 24/7 support with various plans |
| **Security Features** | Advanced security features including IAM, encryption, etc. | Basic security features (firewall, SSL, etc.) | Basic security features with advanced options available |
| **User Interface** | AWS Management Console (complex) | User-friendly control panel | User-friendly control panel |

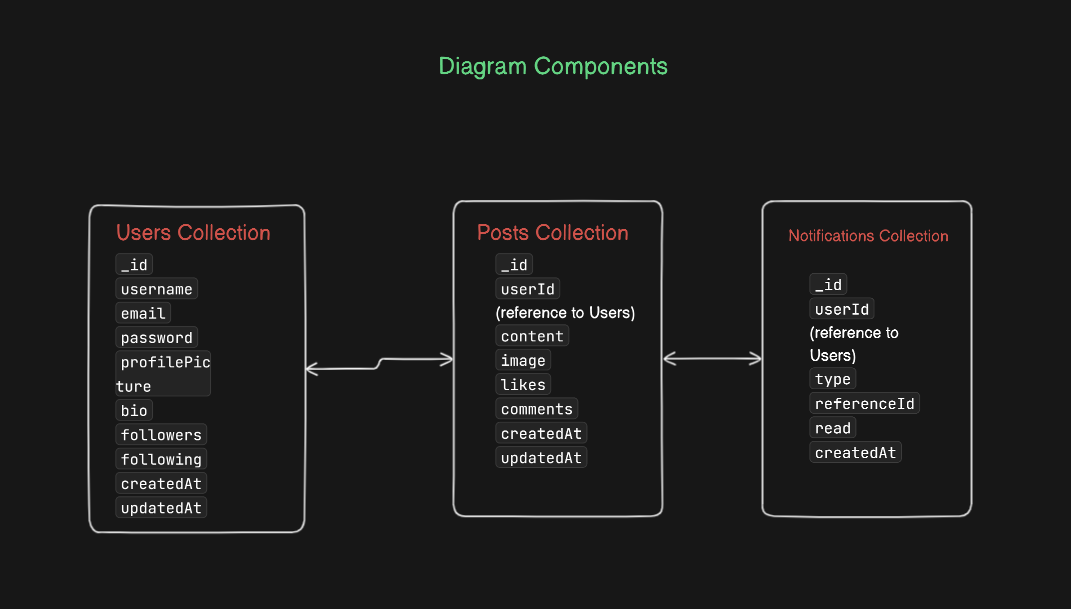
**conclusion**

In conclusion, leveraging MongoDB with AWS and Hostinger hosting services provides a robust solution for building scalable, flexible, and cost-effective applications. This setup combines MongoDB's strengths in handling diverse data types and workloads with AWS’s cloud infrastructure and Hostinger’s reliable hosting services, making it suitable for a wide range of modern application scenarios.

# **My Research and development ( R & D ) to website creation**

Phase - 02

1.Design the database schema and back-end architecture.



**Description: -**

**Users**: Contains user information such as username, email, password, profile picture, bio, followers, and following.

**Posts**: Contains posts made by users, including the content, image, likes, and comments associated with each post.

**Comments**: Contains comments made on posts, including the content, likes, and timestamps.

**Notifications**: Contains notifications for users, including the type of notification (like, comment, follow) and a reference to the relevant post, comment, or user.

**Relationships**: Handles follower-following relationships, where each document represents a follower-following connection.

**Note:-** This diagram gives a high-level view of the entities and their relationships. Each entity is represented as a table with its attributes, and the relationships between entities are indicated by arrows.

2.Plan integration with the front-end.

To integrate the backend with the front-end for a social media app, you need a clear plan to ensure smooth communication between the client and the server. Here is a step-by-step plan:

1. **Define API Endpoints : -**

Identify the necessary API endpoints to handle all CRUD operations for your entities. Here are some examples:

**Users**

· **GET** /api/users: Get all users.

· **GET** /api/users/:id: Get a single user by ID.

· **POST** /api/users: Create a new user.

· **PUT** /api/users/:id: Update a user by ID.

· **DELETE** /api/users/:id: Delete a user by ID.

**Posts**

· **GET** /api/posts: Get all posts.

· **GET** /api/posts/:id: Get a single post by ID.

· **POST** /api/posts: Create a new post.

· **PUT** /api/posts/:id: Update a post by ID.

· **DELETE** /api/posts/:id: Delete a post by ID.

**Comments**

· **GET** /api/comments: Get all comments.

· **GET** /api/comments/:id: Get a single comment by ID.

· **POST** /api/comments: Create a new comment.

· **PUT** /api/comments/:id: Update a comment by ID.

· **DELETE** /api/comments/:id: Delete a comment by ID.

**Notifications**

· **GET** /api/notifications: Get all notifications.

· **GET** /api/notifications/:id: Get a single notification by ID.

· **POST** /api/notifications: Create a new notification.

· **PUT** /api/notifications/:id: Update a notification by ID.

· **DELETE** /api/notifications/:id: Delete a notification by ID.

1. **Set Up Backend Routes:-**

Implement the defined API endpoints using Node.js and Express.

### 3. Implement Controllers

Create controller functions to handle requests and interact with the MongoDB database.

### 4. Set Up MongoDB Models

Define MongoDB models using Mongoose.

### 5. Front-end Integration

Using React for the front-end, you can integrate the backend by making API calls using fetch or a library like axios.