# Blogging Website Design Document

## Project Overview

* **EchoExpress** is a web platform designed for users to create, edit, and share blog posts. It provides functionalities such as user authentication, profile management, and image uploading..

## Goals

* The primary goal of the project is to create a user-friendly and efficient platform for users to share their thoughts, ideas, and projects through blog posts. Key features include:

## Architecture Overview

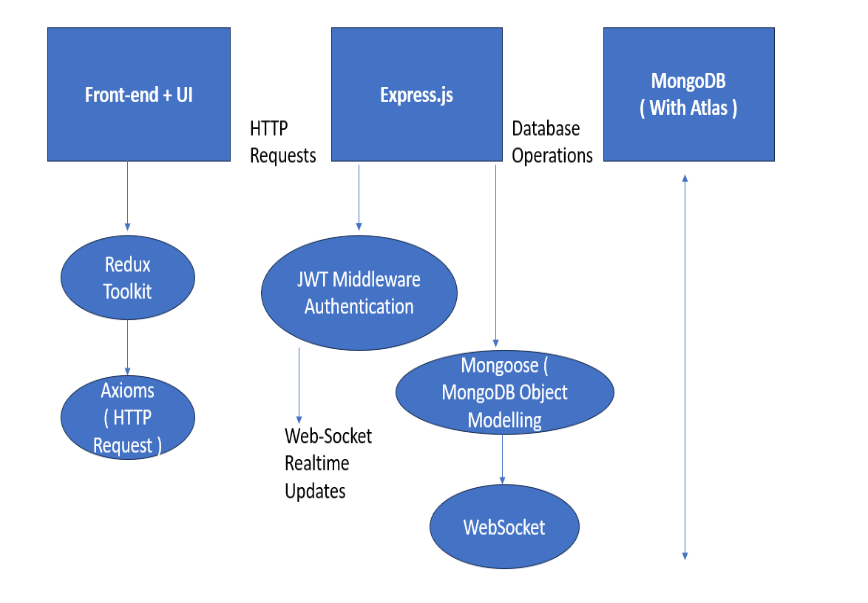
* EchoExpress adopts a scalable and maintainable architecture, utilizing modern technologies to ensure a seamless blogging experience for users. The system architecture consists of frontend and backend components, with Redux managing state for authentication and authorization processes.

## Frontend Architecture

* The frontend of EchoExpress is developed using HTML, CSS, and JavaScript, with Redux integrated to manage application state, including user authentication and authorization. React.js, a popular frontend library, is employed to build dynamic user interfaces, facilitating smooth interaction and real-time updates.

## Backend Architecture

* Powered by Node.js and Express.js, the backend of EchoExpress provides RESTful API endpoints to handle client requests. MongoDB serves as the database, chosen for its scalability and flexibility in managing user accounts, blog posts, and profile information. Authentication and authorization processes are handled using JSON Web Tokens (JWT) for secure access to resources.



## Key Features

### User Authentication and Authorization

* Users can register for an account, providing a unique username and password.
* Users can log in and log out of their accounts securely.
* Passwords will be securely hashed and stored in the database.

### Blog Posting

* Registered users can create new blog posts.
* Each blog post can have a title, content, and optional image attachment.
* Users can publish, save drafts, or schedule posts for future publishing.

### Blog Editing

* Users can edit their existing blog posts.
* Editing capabilities include modifying the title, content, and attached images.
* Users can delete their own blog posts.

### Profile Management

* Users can update their profile information, including their name and profile photo.
* Profile photos can be uploaded from the user's device.
* Users can view their own profile and potentially the profiles of other users.

## Database Schema

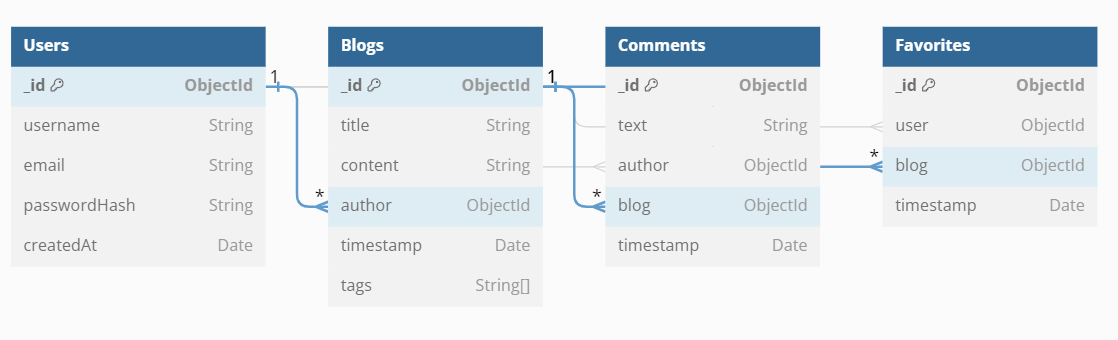
The MongoDB database will consist of the following collections:

1. Users:

* \_id: ObjectId (unique identifier)
* username: String (username of the user)
* email: String (email address of the user)
* passwordHash: String (hashed password of the user)
* createdAt: Date (timestamp of user registration)
* profile\_pic: String(Profile picture of the user)

1. Blog Posts

* \_id: ObjectId (unique identifier)
* title: String (title of the blog post)
* desc: String (content of the blog post)
* username: ObjectId (reference to the user who authored the blog)
* timestamp: Date (timestamp of when the blog post was created)
* photo: String(picture of the blog)



## Technical Requirements

* Frontend: React.js with Redux for state management, styled-components for styling.
* Backend: Node.js with Express.js, JWT for authentication.
* Database: MongoDB with Mongoose for ODM.
* Deployment: Deployment to a cloud platform like render and verceil.

## User Interface Design

* The user interface will be designed to be intuitive and visually appealing, with a focus on simplicity and ease of use. Key pages/screens include:

1. Landing page
2. Sign up/login forms
3. User dashboard
4. Blog creation/editing/deletion forms
5. Blog post display page

## Development Roadmap

* Set up the project structure and initialize the Git repository.
* Implement user authentication and authorization.
* Create CRUD functionality for blogs..
* Implement user following system.
* Design and implement the user interface.
* Test the application thoroughly and fix any bugs.
* Deploy the application to a cloud platform.

## 

ENDPOINTS

## This table outlines the various API endpoints available for user authentication, blog management, comment functionality, user following, and favoriting blog posts. Each endpoint is accompanied by the HTTP method it supports and a brief description of its functionality.

| Endpoint | Method | Description |
| --- | --- | --- |
| /api/auth/register | POST | User registration |
| /api/auth/login | POST | User login |
| api/auth/logout | POST | User logout |
| /api/blogs | GET | Retrieve all blog posts |
| /api/blogs/:id | GET | Retrieve a specific blog post by ID |
| /api/blogs | POST | Create a new blog post |
| /api/blogs/:id | PUT | Update a blog post by ID |
| /api/blogs/:id | DELETE | Delete a blog post by ID |

## 

## Future Enhancements

* Implementing comments section for each blog post.
* Adding categories or tags for organizing blog posts.
* Integrating social media sharing features.
* Implementing search functionality for finding specific blog posts..

## Deployment

Backend Deployment (Render)

The backend of EchoExpress is deployed on Render, a cloud platform for hosting web applications. Render provides a simple and scalable solution for deploying backend services, ensuring high availability and reliability.

**LINK:** <https://echoexpress-1.onrender.com>

Frontend Deployment (Vercel)

The frontend of EchoExpress is deployed on Vercel, a cloud platform for hosting frontend applications. Vercel offers seamless deployment workflows and provides features such as automatic deployments and custom domains.

**LINK:** <https://echo-express.vercel.app/>

## Conclusion

**EchoExpress aims to provide a user-friendly platform for blogging and content creation, with robust features for user management and profile customization. With a well-designed architecture and intuitive user interface, EchoExpress can empower users to share their thoughts and ideas effectively through blog posts.**