# **Technical Design Document**

## **Table of Contents**

1. Introduction
2. System Overview
3. Architecture
   1. High Level Architecture
   2. Component Interactions
4. Backend Design
   1. Technologies Used
   2. Project Structure
   3. API Design
   4. Database Design
5. Frontend Design
   1. Technologies Used
   2. Project Structure
   3. API Design
   4. State Management
   5. Key Components
6. Security Considerations
7. Deployment Plans
8. Future Scope
9. Conclusion

# **Introduction and System Overview**

## **Introduction**

This project is a **modern code-sharing platform** designed specifically for developers, enabling seamless collaboration and efficient sharing of code snippets. With built-in syntax highlighting, adaptive design, and robust security features, the platform provides a streamlined and user-friendly solution for sharing and showcasing code.

## **System Overview**

This project is built using **MERN Stack** which incorporates the following technologies:

1. MongoDB
2. Express.js
3. React.js
4. Node.js

This application is catered to all the online communities and forums where code needs to be shared in a readable fashion. It is much superior to the native solutions provided by most internet forums such as Reddit for sharing code snippets.

# **High Level Architecture**

# **Backend Design**

## **Technologies Used**

* **Node.js:** Javascript runtime.
* **Express.js:** JS Framework for building the API.
* **MongoDB:** A document oriented, noSQL database.
* **Mongoose:** An Object Data Modelling library for MongoDB and Node.js.
* **JWT:** JSON Web Token for authentication.

## **Project Structure:**

* **/backend/index.js:** Entry point for the server.
* **/backed/db.js:** Code that connects to the database.
* **/backend/models:** Definitions for all schemas and exports them as models.
* **/backend/controllers:** Controllers declared on the imported models.
* **/backend/routes:** Handler functions for all the routes supported by the API.

# **API Design**

## **Endpoints**

1. **User Signup**

* **URL:** /auth/signup
* **Method:** POST
* **Description:** Registers a new user

1. **User Sign in:**

* **URL:** /auth/singin
* **Method:** POST
* **Description:** Authenticates an existing user and returns a JWT Token

1. **Get All Entities:**

* **URL:** /pastes
* **Method:** GET
* **Description:** Get all pastes that the (authenticated) user has created.

1. **Edit an Entity:**
   * **URL:** /:id
   * **Method:** PUT
   * **Description:** Updates the entity if the (authenticated) user is the creator.
2. **Get and Entity:**
   * **URL:** /id
   * **METHOD:** GET
   * **Description:** Gets the particular entity based on a valid uid.

## **Authentication and Authorization**

* **Token Based Authentication:** All protected routes require a JSON Web Token (JWT) passed in the Authorization Header.

# **Database Design**

## **User Schema**

**Collection Name:** users

**Purpose:** Stores user information for authentication and authorization

**Fields:**

| **Field Name** | **Type** | **Required** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| \_\_id | ObjectId | Yes | Auto | Unique identifier for the user. |
| username | String | Yes | None | Unique username chosen by the user. |
| email | String | Yes | None | User’s email address. |
| password | String | Yes | None | Hashed password for authentication. |
| createdAt | Date | Yes | Auto | Timestamp when the user was created. |
| pastes | [entitySchema] | No | [] | Array of entities created by the user. |

## **Paste Schema**

**Collection Name:** entities

**Purpose:** Stores paste content created by users.

**Fields:**

| **Field Name** | **Type** | **Required** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| \_\_id | ObjectId | Yes | Auto | Unique identifier for the entity. |
| title | String | Yes | None | Title of the text snippet saved. |
| content | String | Yes | None | The actual content (text/code). |
| updatedAt | Date | Yes | None | Timestamp when the entity was last updated. |
| createdAt | Date | Yes | Auto | Timestamp when the entity was created. |
| burnAfterRead | Boolean | No | False | Whether or not the entity will be marked for deletion after first read. |
| expiresAfter | Date | No | None | The timestamp after which the entity will be marked for deletion. |

# **Frontend Design**



### **Technologies Used**

* + **React.js**: JavaScript library for building user interfaces.
  + **React Router DOM**: Handling client-side routing.
  + **Bootstrap:** A CSS framework for designing responsive and mobile-first web applications, used to style and layout the user interface.
  + **ESLint**: Linting utility to maintain code quality.

### **Project Structure**

* + **main.jsx**: Entry point of the React application.
  + **App.jsx**: Main application component.
  + **App.css**: Global CSS and Tailwind directives.

# **Frontend Design - Routing, State Management and Key Components**



### **Routing**

Implemented using React Router:

* + /: Landing Page
  + /<post URL>: Link to the shareable text snippet

### **State Management**

#### **Coding Space**

* + - Has “placeholder” on the main page (when “value” is null).
    - Has the content of the paste on view page (“value” is passed).
    - Syntax highlighting used for the “language” passed.
  + **Drop down menu**
    - Contains the list of items specified in “items”.
    - Labelled with “heading” passed.

### **Key Components**

### **Landing Page (Home Page)**:

### A welcoming page with options to paste text and share it via a generated link.

### Includes a form to input text, validate its content, and generate a shareable URL.

* Includes options to input title and set the expiration time.
* Includes a dropdown to select the language to use syntax highlighting.

### The layout is responsive, using Bootstrap.

**Paste Viewing Component**:

* Displays the shared text snippet when the user navigates to a unique post URL (e.g., /a1b2c3).
* Dynamically fetches and renders the content based on the URL parameters.
* Includes additional functionality to edit the content of the paste.

**Error Handling**:

* A generic error page is displayed for invalid or non-existing post URLs.
* A user-friendly message is shown if the content does not exist or there is a server error.