***MyPrivNote Documentation***

1. Introduction:

MyPrivNote is a web-based application designed to enable users to create and send self-destructing, secure notes. The notes are deleted automatically after they are read, ensuring that sensitive information does not persist beyond its intended use. The application is developed using Node.js and MongoDB, and the initial focus is on building a secure and scalable backend that supports REST API operations for note management.

1. What?

This project will replicate the core functionality of PrivNote.com, offering users the ability to create and share notes that self-destruct after being read. Users can interact with the application via a web-based user interface or some API’s.

1. Why?

The need for privacy and security in online communication is increasingly important. MyPrivNote addresses this by offering a simple and secure platform for sharing sensitive information that is automatically deleted after being accessed.

1. List of Supported Features

* Creation of secure, self-destructing notes via API’s.
* Retrieval of notes by their unique ID.
* HTTPS support for secure communication (planned for future weeks).
* MongoDB as the database for note storage.

1. List of Not Supported Features

* User authentication.
* Full CRUD operations.
* Logging and auditing features.

1. List of Future Planned Features
   * User authentication and authorization (JWT-based).
   * Full CRUD API (create, read, update, delete).
   * Data encryption before storage.
   * HTTPS configuration.
   * Integration with mobile apps (iOS/Android).
   * Scalability to support millions of users.
2. How?
3. High-Level Diagram

* A high-level diagram shows the flow of data between:
* Frontend: HTML/JavaScript-based UI (to be developed).
* Backend: Node.js server with Express.
* Database: MongoDB for storing notes.

1. Components/Modules

* server.js: Core Express server that handles API routes.
* db.js: Manages MongoDB connection.
* models/Note.js: Mongoose schema for managing notes in the database.

1. Languages and Technologies

* Backend: Node.js, Express.js
* Database: MongoDB
* Schema Management: Mongoose
* Environment Variables: dotenv

1. 3rd Party/Open Source Modules

|  |  |
| --- | --- |
| Library | License |
| Express | MIT |
| Mangoose | MIT |
| dotenv | MIT |
| Node js | MIT |

* Mongoose: ODM for MongoDB.
* Express: Web framework for building APIs.
* dotenv: For managing environment variables.

1. **List of Any 3rd-Party Services/APIs:**

**MongoDB Atlas** (or local MongoDB instance) for database storage.

1. REST API Endpoints

* POST

Description: Creates a new note with unique content and URL(Response , Request).

* GET

Description: Retrieves a note by its unique URL(Response).

1. Build Steps/Scripts

* Create the Project Files
* Navigate to the Project Directory.
* Initialize the Node.js Project
* Install Dependencies
* Create the .env File
* Run the Application.

1. Github information:
2. References

**MongoDB Documentation**: <https://docs.mongodb.com/>

**Express.**js Documentation : <https://expressjs.com/>

**Node.js Documentation**: <https://nodejs.org/en/docs/>

**Mongoose Documentation**: https://mongoosejs.com/docs/