Message Signature App – Documentation

1. Introduction

The **Message Signature App** is a secure communication platform where users can send, save, and verify messages using digital signatures. This application ensures the integrity of messages by generating a digital signature for each message sent. The goal is to protect data from tampering and provide a way to verify whether the saved message and fetched message remain identical.

It demonstrates effective frontend-backend communication and emphasizes **security** through hashing algorithms.

2. Technologies Used

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js, Express
- **Security:** SHA-256 hashing for message signatures
- **Dependencies:**
  - `Express` for routing
  - `Body-Parser` for JSON parsing
  - `CORS` for handling cross-origin requests
  - `crypto` for generating secure hashes

3. Features

1. **Send Message:**
   Users can send messages to be stored with a digital signature.

2. **Save Message:**
   Messages are saved in memory with an associated digital signature.

3. **Fetch Saved Message:**
   Retrieve previously saved messages from the vault.

4. **Verify Message:**
   Verify if the saved and fetched messages are identical by comparing their digital signatures.

5. **Clean UI:**
   User-friendly design with aligned buttons and input fields

4. Installation & Setup

Prerequisites:

- Node.js installed on your machine.
- Code editor (e.g., Visual Studio Code).
- Web browser (e.g., Chrome).

Steps:

1. **Clone the Repository:**

   ```
   git clone <repository-url>
   ```

2. **Navigate to Backend Directory:**

   ```
   cd backend
   ```

3. **Install Dependencies:**

   ```
   npm install
   ```

4. **Start the Backend Server:**

   ```
   node index.js
   ```

   The backend will be running on **http://localhost:5000**.

5. **Open the Frontend:**
   Navigate to the `index.html` file inside the frontend folder and open it in your browser, or run:

   ```
   npx http-server ./frontend
   ```

5. How It Works

1. **Send Message:**
   - The user inputs a **message** in the sender's section and clicks **Send**.
   - The message, along with a digital signature, is sent to the backend via a POST request.

2. **Save Message:**
   - The message and its **digital signature** are stored in memory.
   - The digital signature ensures that the integrity of the message is maintained

3. **Fetch Message:**
   - On clicking the **Fetch** button, the saved message is retrieved from the vault.
   - The saved message is displayed along with its corresponding digital signature.

4. **Verify Message:**
   - Clicking the **Verify** button compares the original saved message with the fetched message using their digital signatures.
   - A success message appears if the signatures match; otherwise, it shows an error.

6. Endpoints & API Documentation

POST `/save`

Saves a message along with its digital signature.

- **Request Body:**

  {

  "message": "Hello, this is a secret message!"

  }

- **Response:**

  {

  "signature": "3f47e29dfbf32c..."

  }

GET `/fetch`

Fetches the saved message from the vault.

- **Response:**

  {

  "message": "Hello, this is a secret message!"

  }

POST `/verify`

Verifies if the fetched message matches the original saved message.

- **Request Body:**

  {

  "message": "Hello, this is a secret message!",

  "signature": "3f47e29dfbf32c..."

  }

- **Response:**

```
{

  "isValid": true

}
```


7. Usage Instructions

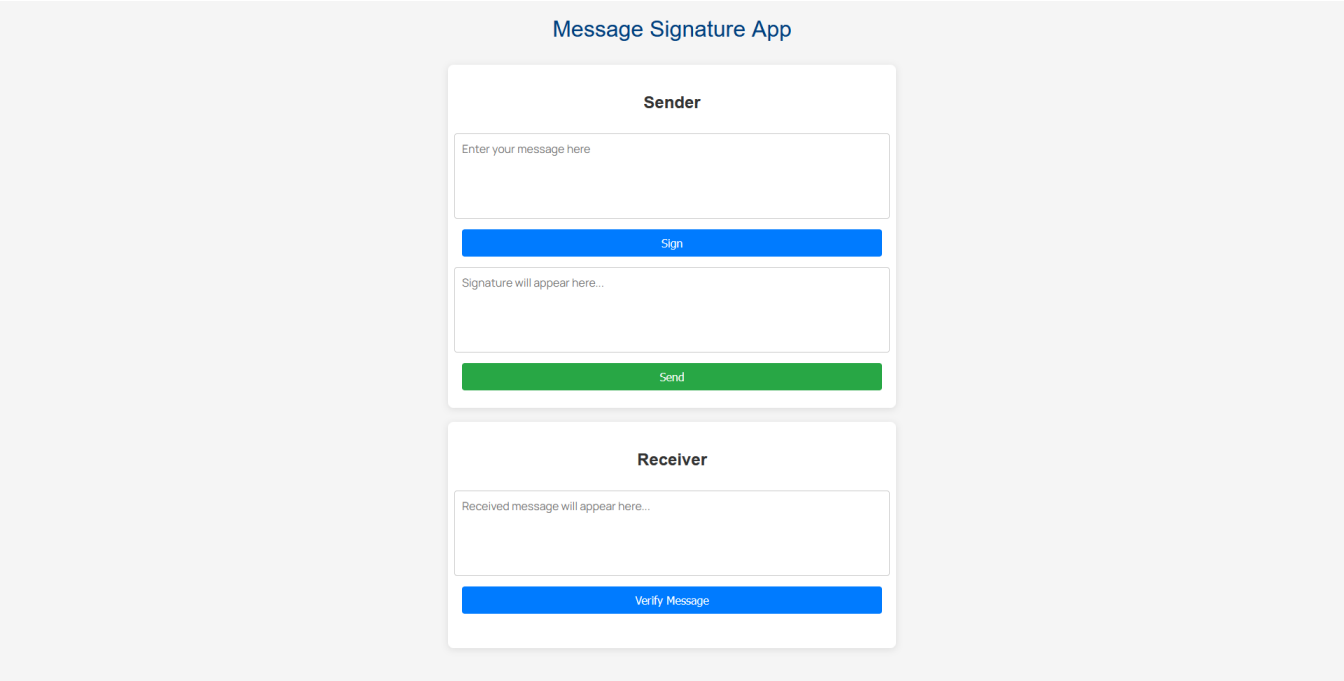
1. **Launch the Application:**
   Open the frontend in your web browser.

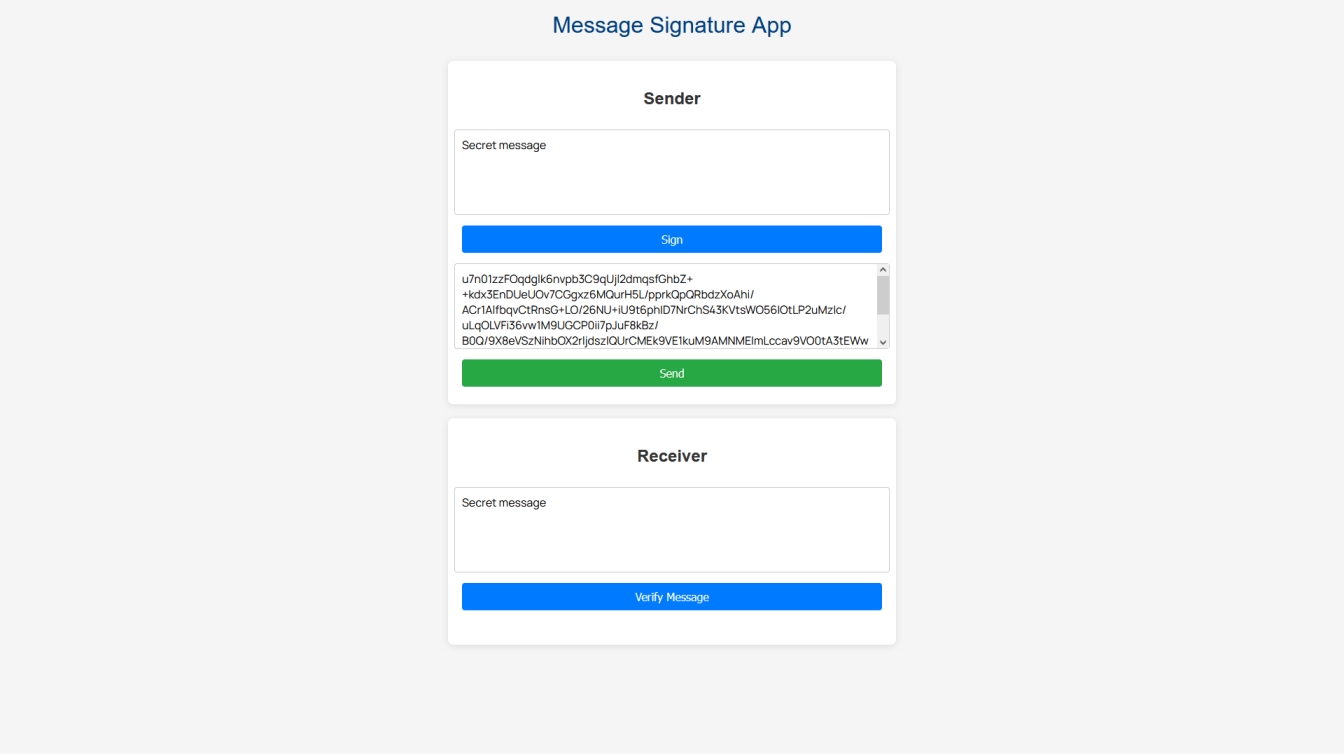
2. **Send a Message:**
   Type a message in the **sender** field and click **Send**.

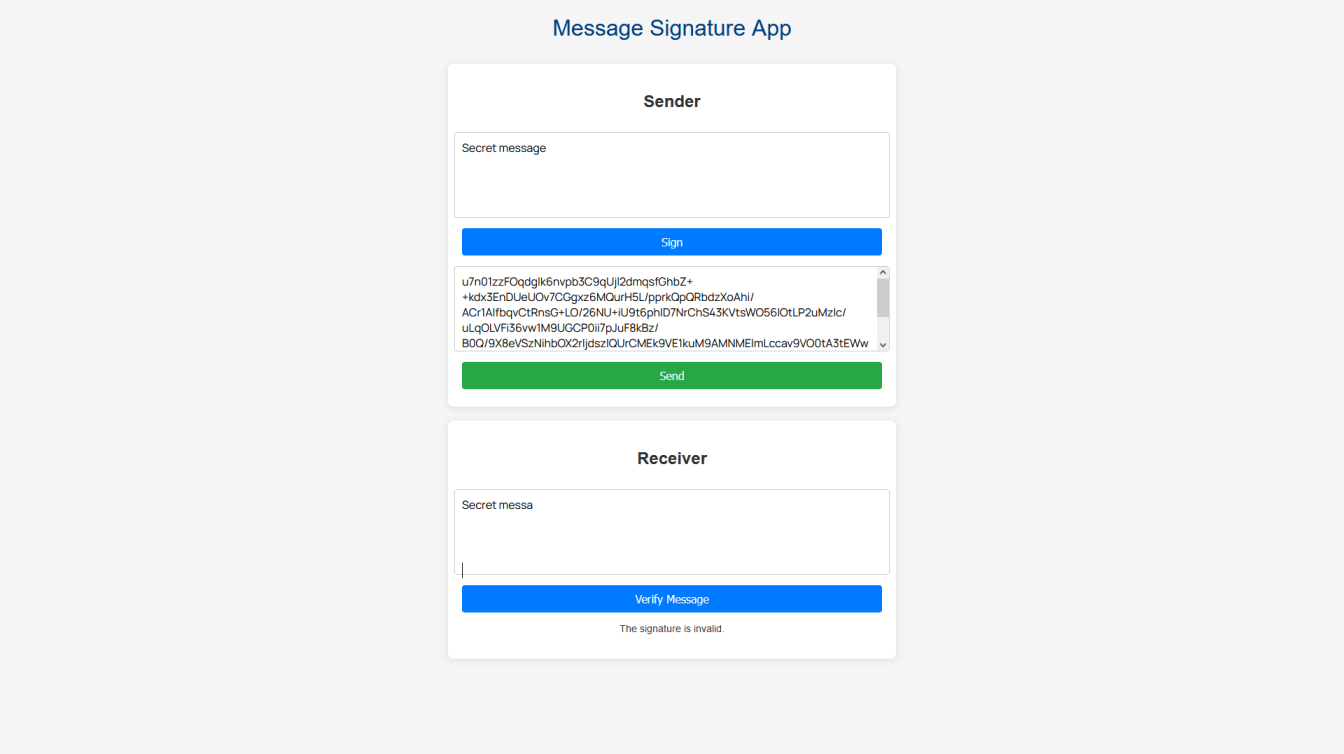
3. **Fetch Saved Message:**
   Click the **Fetch** button to display the message in the vault.

4. **Verify Message:**
   Click **Verify** to check whether the saved and fetched messages match using the digital signature.

8. Screenshots







9. Conclusion

The **Message Signature App** provides a secure and simple way to send, save, fetch, and verify messages using digital signatures. It highlights the importance of data integrity and can serve as a foundation for more complex, secure messaging applications. With further improvements, the app can be expanded into a fully-featured secure messaging system.