Cipher Vault — Server Test

Goal

Build a Node.js + Express server that:

- stores **users in MongoDB**
- stores **encrypted messages in MySQL**
- allows users to encrypt and decrypt messages using simple ciphers after authenticating

MongoDB and MySQL are started manually using Docker/using managed services like atlas or supabase. Describe in README what and why your selection is.

Tech Rules

- Node.js + Express
- MongoDB (Docker/atlas, `mongodb` or `mongoose`)
- MySQL (Docker/supabase, `mysql2`)
- Authentication: **HTTP Basic Auth**
- Passwords are stored as **plain text**
- No file storage

Data

MongoDB — `users collection`

- `_id`
- `username` (unique)
- `password` (plain text)
- `encryptedMessagesCount`
- `createdAt`

MySQL — `messages table`

- `id`
- `username`
- `cipher_type`
- `encrypted_text`
- `inserted_at`

Supported Ciphers

- `REVERSE` — reverse the string and uppercase it, reversible (can be decrypted)
- `ATBASH` — reversible (can be decrypted) - BONUS
- `RANDOM_SHUFFLE` — not reversible (cannot be decrypted) - Bonus

Endpoints

1) Register User

POST `/api/auth/register`

Body

{ "username": "dana", "password": "1234" }

Response 201

{ "id": "mongoUserId", "username": "dana" }

Description
Create a new user in MongoDB.

---

2) Encrypt Message

POST `/api/messages/encrypt`
Basic Auth required - user needs to provide username and password, and must be authenticated before allowed to the actual endpoint - you can pass username and password in body/headers/query params

Body

{ "message": "attack at dawn", "cipherType": "reverse" }

Response 201

```
{
  "id": 12,
  "cipherType": "reverse",
  "encryptedText": "NWAD TA KCATTA”
}
```

Description
Encrypt a message, save it in MySQL, and increase the user counter (`encryptedMessagesCount)`

---

3) Decrypt Message

POST `/api/messages/decrypt`
Basic Auth required - user needs to provide username and password, and must be authenticated before allowed to the actual endpoint - you can pass username and password in body/headers/query params

Body

```
{ "messageId": 12 }
```

Response 200 (reversible)

```
{ "id": 12, "decryptedText": "attack at dawn" }
```

Response 200 (not reversible)

```
{ "id": 12, "decryptedText": null, "error": "CANNOT_DECRYPT" }
```

Description
Load message using id. Decrypt a message if possible.

4) My Profile

GET `/api/users/me`
Basic Auth required - user needs to provide username and password, and must be authenticated before allowed to the actual endpoint- you can pass username and password in body/headers/query params

Response 200

```
{
  "username": "dana",
  "encryptedMessagesCount": 3
}
```

Description
Return user data from MongoDB.

5) List My Messages (bonus)

GET `/api/messages`
Basic Auth required - user needs to provide username and password, and must be authenticated before allowed to the actual endpoint- you can pass username and password in body/headers/query params

Response 200

```
{
  "items": [
    { "id": 12, "cipherType": "ATBASH", "encryptedText": "..." }
  ]
}
```

Description
Return all messages of the given username. Pass the username in body/path query/path param

---

Submission

- Node.js project
- README with:
    - Docker run commands for MongoDB and MySQL
    - How to start the server
    - Endpoint list and sort explanation for each. Include path params/queries/body if needed and return value.
    - explanation which db configuration you used and why

Bonus ciphers (in addition to bonus endpoints):

1. implement atbash cipher - can be revered
2. implement shuffle cipher - cannot be reversed

```
RANDOM_SHUFFLE Cipher Algorithm
Convert message to array of characters
Repeat 5000 times:
pick two different random indexes - if they are the same: redo
swap their characters
Join back to string
Decryption is not possible.
```

Grading Table

Area	**Points**
POST /api/auth/register	20
POST /api/messages/encrypt	25
POST /api/messages/decrypt	20
GET /api/users/me	15
General API Flow & Clean Code	15
Project Setup & README	5
Bonus Features	**+15**

Base Total: 100 points
Maximum Score With Bonus: 115 points