Password Keeper/Manager App -

Statement for Candidates:

Below is the problem statement for your project.

1. Your task is to create an **API** that supports project requirements.
2. You have 3 hours to develop the API.
3. You are free to use google to find solutions, however, copying an existing project is not allowed.
4. At the end of the 3 hour time period, you will be given get 15 minutes to do your submission on Google Form & to push your project in a GitHub repository, submission form: https://forms.gle/PqfNAakDWgz146BJ8
5. Your API submission will be evaluated by our team and a result will be published at 11:30 AM on the same day.
6. No need to create the session, authenticate using user_id
7. No need to apply JWT or any token for authentication, simply use user_id to authenticate

What is the project?
A web application that keeps a record of users' passwords. Users can log into the panel to see the username and passwords of websites they have saved.

Requirements:

1. User account registration:
   Create a user account. These credentials will be used to log into this panel.

```
[POST] /app/user

Request Data: {
    'username': str,
    'password': str
}

Response Data: {
    'status': 'account created'
}
```

2. User account login:
   Provide ability to log into the panel using the user credentials.

```
[POST] /app/user/auth

Request Data: {
    'username': str,
    'password': str
}

Response Data: {
    'status': 'success',
    'userId': int
}
```

3. List of saved usernames and passwords for different websites:
   Provide ability for user to see list of previously stored website usernames & passwords.

   ```
   [GET] /app/sites/list/?user={userId}

   Request Data: None
   Response Data: [List of stored website username & passwords]
   ```

   List returned should be belong to the `userId` passed with the request
4. Save a new username & password for website:
   Provide ability for user to add new username & password for a website.

   ```
   [POST] /app/sites?user={userId}

   Request Data: {
       'website': str,
       'username': str,
       'password': str
   }

   Response Data: {
       'status': 'success'
   }
   ```

5. **Mandatory Requirement:** All the passwords stored in the database have to be encrypted.
   No one should be able to read the stored passwords without a secret key. The key can be stored in the code itself.

Tech Stack:

1. Any web-server of your choice (Python Flask / Django, NodeJS Express / Koa, Java, etc)
2. Database: MySQL (Compulsory)