Admin Panel

Introduction => The admin panel project is a web-based application that allows administrators to perform various tasks related to managing PCNs. The project includes several APIs that perform specific functions, such as generating PCNs, removing PCNs, verifying PCNs, and retrieving all PCNs. This technical documentation aims to provide an overview of the APIs and their usage.

This document provides information about the API endpoints, their payloads, and sample usage of the API.

API List:
1)
POST /signup
This endpoint is used to sign up a user. The endpoint requires a name, an email and a password. If the email and password are valid and match with validation criteria, the endpoint creates a new user and returns it in the response.
Payload:

```
{
Name: “User”
"email": "user@example.com",
"password": "mypassword"
}
```

Response:

```
{
Name: “User”
"email": "user@example.com",
"password": "mypassword"
}
```

2)
POST /login
This endpoint is used to login a user. The endpoint requires an email and a password. If the email and password are valid, the endpoint generates a JWT token and returns it in the response.

Payload:

```
{
"email": "user@example.com",
"password": "mypassword"
}
```

Response:

```
{
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI1YmM3MGM5Nj
IxYjJjODAwYzYwZDU2NTIiLCJpYXQiOjE1Njg0ODk5NjYsImV4cCI6MTU2OD
Q5MzE2Nn0.L7cqCzUJjpbCC1rDYU15p-7Mv9X1DXIVTKNUP2eP-Qo"
}
```

3)

POST /api/auth/verify

This endpoint is used to verify the JWT token. The endpoint requires a token in the request body. If the token is valid, the endpoint returns a success message.

Payload:

```
{
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiI1YmM3MGM5Nj
IxYjJjODAwYzYwZDU2NTIiLCJpYXQiOjE1Njg0ODk5NjYsImV4cCI6MTU2OD
Q5MzE2Nn0.L7cqCzUJjpbCC1rDYU15p-7Mv9X1DXIVTKNUP2eP-Qo"
}
```

Response:

```
{
"message": "Token verified"
}
```

4)

POST /generate-pcn

This endpoint is used to generate a new PCN. The endpoint requires a PCN in the request body. If the PCN does not exist, the endpoint generates a random token and saves the PCN and token to the database. The endpoint returns the token.

Payload:

```
{
```

```
"Pcn": "ABCD1234"
}
```

Response:

```
{
"Token":
"5fddad8c4b968bd563ac3c4e318bc0c8bde85a61a121cbb25d962abf1528b10d"
}
```

5)
POST /verify-pcn

This endpoint is used to verify PCN. The endpoint requires a PCN and an associated token with it in the request body. If the PCN and the token associated with it does not exist, the endpoint returns an error (PCN not verified) and if both the token and the PCN matches it will return a success message. The endpoint returns the token.

Payload:

```
{
"Pcn": "ABCD1234",
"Token":
"5fddad8c4b968bd563ac3c4e318bc0c8bde85a61a121cbb25d962abf1528b10d”,
}
```

Response:

```
{
Success: true,
}
```

6)
POST /remove-pcn

This endpoint is used to remove PCN. The endpoint requires a PCN in the request body. If the PCN does not exist, the endpoint returns an error (exist: false) If the PCN exists in the database. The endpoint returns the success (exist: true).

Payload:

```
{
"Pcn": "ABCD1234"
}

Response:
{
Exist: true,
}
```

7)

GET /get-all-pcn

This endpoint is used to retrieve all generated PCNs. The endpoint returns an array of all PCNs and their tokens.

Response:

```
[
{
"Pcn": "ABCD1234",
"Token":
"5fddad8c4b968bd563ac3c4e318bc0c8bde85a61a121cbb25d962abf1528b10
d",
}
]
```

API Usage:

1)

/signup: To be able to sign up to an application send a POST request to the /http://localhost:8000/signup endpoint. This endpoint is used for sign in as an admin user. It expects a name, email and password in the request body and checks if the user has provided the valid email password which fulfils the validation criteria then the server generates a new user inside the database and returns it as a response.

2)

/login: To be able to log in to an application send a POST request to the /http://localhost:8000/login endpoint This endpoint is used for logging in as an admin user. It expects an email and password in the request body and checks if the user exists in the database and if the provided password matches the stored password. If the credentials are valid, the server generates a JSON Web Token and returns it as a response.

3)
/api/auth/verify: To verify the jwt token, send a POST request to the http://localhost:8000/api/auth/verify endpoint, This endpoint is used for verifying the validity of a JSON Web Token. It expects a token in the request body and uses a middleware function (verifyToken) to check if the token is valid. If the token is valid, the server returns a success message.

4)
/generate-pcn: To generate a new PCN, send a POST request to the http://localhost:8000/generate-pcn endpoint. This endpoint is used for generating a new PCN. It expects a PCN in the request body and checks if the PCN already exists in the database. If the PCN does not exist, the server generates a random token and creates a new PCN document in the database with the PCN and token. It then returns the generated token as a response.

5)
/get-all-pcn: To retrieve all PCNs from the database, send a GET request to the http://localhost:8000/get-all-pcn endpoint. The API will return all PCNs in the response body. This endpoint is used for retrieving all the PCNs that have been generated. It retrieves all the PCN documents from the database and returns them as a response.

6)
/verify-pcn: To verify the PCN from the database, send a post request to the http://localhost:8000/verify-pcn endpoint. This endpoint is used for verifying the validity of a PCN and its associated token. It expects a PCN and a token in the request body and checks if a PCN document with the provided PCN and token exists in the database. If it does, the server returns a response indicating that the PCN exists, otherwise, it returns a response indicating that the PCN does not exist.

7)
/remove-pcn: To remove the PCN from the database, send a post request to the http://localhost:8000/remove-pcn endpoint. This endpoint is used for removing the PCN from the database. It expects a PCN in the request body and checks if the provided PCN exists in the database. If it does, the server returns a response indicating that the PCN was removed, otherwise, it returns a response indicating that the PCN was not found.

The server also defines two Mongoose models:

1) Admin: This model represents an admin user and has fields for name, email, password, createdAt, and updatedAt.

2) PCN: This model represents a PCN document and has fields for PCN, Token, createdAt, and updatedAt.

Conclusion:

The admin panel project includes several APIs that perform specific functions related to managing PCNs as well as user authentication and token validation. These APIs allow administrators to generate PCNs, remove PCNs, verify PCNs, and retrieve all PCNs from the database. This technical documentation provides an overview of the APIs and their usage.