Understanding JWT in Flask API

# 1. jwt\_required()

The @jwt\_required() decorator protects a route, ensuring only authenticated users with a valid token can access it.  
  
How it works:  
- After login, the client receives a JWT access token.  
- To access protected routes, the token must be sent in the Authorization header:  
 Authorization: Bearer <your-token>  
- If the token is missing or invalid, the request is rejected with a 401 error.  
  
Used in:  
- AdminCreateStudent.post()  
- AdminCreateEducator.post()  
- ChangePassword.post()  
- UserProfile.get()

# 2. get\_jwt\_identity()

This function extracts the user's identity from the JWT after it's validated.  
  
It returns the data provided when creating the access token with create\_access\_token(identity=...).  
  
Example:  
identity = {  
 'id': user.id,  
 'role': user.role,  
 'school\_id': user.school\_id,  
 'full\_name': user.full\_name  
}  
get\_jwt\_identity() will return this dictionary so the app knows who is making the request.

# 3. JWTManager

JWTManager(app) initializes JWT functionality in a Flask application. It enables token creation and validation and ties into Flask’s request lifecycle.

# Putting It All Together

Login:  
- The Login.post() route uses create\_access\_token(identity=...) to create a JWT.  
- The token is returned to the frontend.  
  
Accessing Protected Routes:  
- The frontend sends the JWT in request headers.  
- @jwt\_required() validates it.  
- get\_jwt\_identity() extracts the user info for role-based access control.

# Summary

|  |  |
| --- | --- |
| Concept | Description |
| JWTManager(app) | Initializes JWT support for Flask app |
| @jwt\_required() | Protects routes, allows only valid JWT token requests |
| get\_jwt\_identity() | Gets identity info from the JWT |
| create\_access\_token | Generates a JWT for authenticated users |