I can suggest an approach to showcase different authentication and authorization methods using a Flask app and a MongoDB database with a user interface.

To start with, we can create a Flask app and connect it to a MongoDB database using the PyMongo library. We can also create HTML templates and static files (CSS and JavaScript) for different pages to display authentication and authorization methods.

Here's a step-by-step guide on how to implement this:

1. Install the necessary libraries and dependencies. You can install Flask and PyMongo using pip:

pip install Flask

pip install pymongo

1. Create a MongoDB database and collection for storing user credentials. We can use the insert\_one method of PyMongo to insert a user document into the collection:

from pymongo import MongoClient

client = MongoClient('<mongodb\_uri>')

db = client['mydatabase']

users = db['users']

user = {'username': 'alice', 'password': '<hashed\_password>'}

users.insert\_one(user)

1. Create a Flask app and define different routes for different pages. We can use the render\_template method to render the HTML templates:

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

    return render\_template('home.html')

@app.route('/login')

def login():

    return render\_template('login.html')

@app.route('/register')

def register():

    return render\_template('register.html')

@app.route('/dashboard')

def dashboard():

    return render\_template('dashboard.html')

1. Create HTML templates for different pages. We can use Bootstrap to style the templates:

home.html:

<!DOCTYPE html>

<html>

<head>

    <title>Home</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+J903MyyA6rsJpeg6vz8V+NcX6JwBf" crossorigin="anonymous">

</head>

<body>

    <div class="container">

        <h1>Welcome to my app</h1>

        <p>Please log in or register to continue</p>

        <a href="/login" class="btn btn-primary">Log in</a>

        <a href="/register" class="btn btn-secondary">Register</a>

    </div>

</body>

</html>

Login.html:

<!DOCTYPE html>

<html>

<head>

    <title>Login</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+J903MyyA6rsJpeg6vz8V+NcX6JwBf" crossorigin="anonymous">

</head>

<body>

    <div class="container">

        <h1>Login</h1>

        <form method="POST">

            <div class="form-group">

                <label>Username</label>

                <input type="text" name="username" class="form-control">

            </div>

            <div class="form-group">

                <label>Password</label>

                <input type="password" name="password" class="form-control">

            </div>

            <button type="submit" class="btn btn-primary">Log in</button>

        </form>

    </div>

</body>

</html>

Register.html:

<!DOCTYPE html>

<html>

<head>

    <title>Register</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+J903MyyA6rsJpeg6vz8V+NcX6JwBf" crossorigin="anonymous">

</head>

<body>

    <div class="container">

        <h1>Register</h1>

        <form method="POST">

            <div class="form-group">

                <label>Username</label>

                <input type="text" name="username" class="form-control">

            </div>

            <div class="form-group">

                <label>Password</label>

                <input type="password" name="password" class="form-control">

            </div>

            <div class="form-group">

                <label>Confirm Password</label>

                <input type="password" name="confirm\_password" class="form-control">

            </div>

            <button type="submit" class="btn btn-primary">Register</button>

        </form>

    </div>

</body>

</html>

1. Implement different authentication methods. We can use Flask-Login to handle user authentication and session management:

from flask\_login import LoginManager, UserMixin, login\_user, logout\_user, login\_required

app.secret\_key = '<secret\_key>'

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

class User(UserMixin):

    pass

@login\_manager.user\_loader

def user\_loader(username):

    user = users.find\_one({'username': username})

    if user:

        user\_obj = User()

        user\_obj.id = username

        return user\_obj

    return None

@app.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        user = users.find\_one({'username': username, 'password': '<hashed\_password>'})

        if user:

            user\_obj = User()

            user\_obj.id = username

            login\_user(user\_obj)

            return redirect('/dashboard')

        else:

            flash('Invalid username or password')

    return render\_template('login.html')

@app.route('/register', methods=['GET', 'POST'])

def register():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        confirm\_password = request.form['confirm\_password']

        if password == confirm\_password:

            user = {'username': username, 'password': '<hashed\_password>'}

            users.insert\_one(user)

            flash('You have successfully registered')

            return redirect('/login')

        else:

            flash('Passwords do not match')

    return render\_template('register.html')

@app.route('/logout')

@login\_required

def logout():

    logout\_user()

    return redirect('/')

1. Implement different authorization methods. We can use Flask-Principal to handle role-based access control:

from flask\_principal import Principal, Permission, RoleNeed, identity\_loaded, Identity

app.config['PRINCIPAL\_ROLE\_ADMIN'] = 'admin'

app.config['PRINCIPAL\_ROLE\_USER'] = 'user'

principal = Principal(app)

admin\_permission = Permission(RoleNeed(app.config['PRINCIPAL\_ROLE\_ADMIN']))

user\_permission = Permission(RoleNeed(app.config['PRINCIPAL\_ROLE\_USER']))

@app.route('/dashboard')

@admin\_permission.require()

def dashboard\_admin():

    return render\_template('dashboard\_admin.html')

@app.route('/dashboard')

@user\_permission.require()

def dashboard\_user():

    return render\_template('dashboard\_user.html')

@app.route('/assign-role', methods=['POST'])

@admin\_permission.require()

def assign\_role():

    username = request.form['username']

    role = request.form['role']

    user = users.find\_one({'username': username})

    if user:

        users.update\_one({'username': username}, {'$set': {'role': role}})

        flash(f'{username} has been assigned the {role} role')

    else:

        flash('User does not exist')

    return redirect('/dashboard')

@identity\_loaded.connect\_via(app)

def on\_identity\_loaded(sender, identity):

    identity.user = current\_user

    if hasattr(current\_user, 'role'):

        if current\_user.role == app.config['PRINCIPAL\_ROLE\_ADMIN']:

            identity.provides.add(RoleNeed(app.config['PRINCIPAL\_ROLE\_ADMIN']))

        if current\_user.role == app.config['PRINCIPAL\_ROLE\_USER']:

            identity.provides.add(RoleNeed(app.config['PRINCIPAL\_ROLE\_USER']))

1. Create HTML templates for each of the authorization levels:

templates/dashboard\_admin.html:

<!DOCTYPE html>

<html>

<head>

    <title>Dashboard - Admin</title>

</head>

<body>

    <h1>Dashboard - Admin</h1>

    <form method="POST" action="/assign-role">

        <div class="form-group">

            <label>Username</label>

            <input type="text" name="username" class="form-control">

        </div>

        <div class="form-group">

            <label>Role</label>

            <select name="role" class="form-control">

                <option value="admin">Admin</option>

                <option value="user">User</option>

            </select>

        </div>

        <button type="submit" class="btn btn-primary">Assign Role</button>

    </form>

    <br>

    <a href="/logout">Logout</a>

</body>

</html>

templates/dashboard\_user.html:

<!DOCTYPE html>

<html>

<head>

    <title>Dashboard - User</title>

</head>

<body>

    <h1>Dashboard - User</h1>

    <p>Welcome to your dashboard. You do not have administrative privileges.</p>

    <br>

    <a href="/logout">Logout</a>

</body>

</html>

1. Finally, we need to create the MongoDB database and collections and start the Flask app:

from pymongo import MongoClient

client = MongoClient('<mongodb\_uri>')

db = client['<db\_name>']

users = db['users']

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Make sure to replace <mongodb\_uri> with the URI for your MongoDB database and <db\_name> with the name of your database.

I hope this helps! Let me know if you have any further questions.