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
```

2. 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)
```

3. 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')
```

4. 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>
```

5. 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('/')
```

6. 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']))
```

7. 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>
```

8. 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.