**Assessment-5**

1. Create an application for user authentication with customized exceptions and CRUD functionality using Python Flask and SQLAlchemy.

Install Flask and SQLAlchemy:

|  |
| --- |
| pip install Flask Flask-SQLAlchemy |

Create a file named app.py for your Flask application:

|  |
| --- |
| from flask import Flask, request, jsonify, abort  from flask\_sqlalchemy import SQLAlchemy  from werkzeug.security import generate\_password\_hash, check\_password\_hash  app = Flask(\_\_name\_\_)  app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///app.db' # Use SQLite for simplicity  app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  app.config['SECRET\_KEY'] = 'your\_secret\_key' # Change this to a secret key for production  db = SQLAlchemy(app)  class User(db.Model):  id = db.Column(db.Integer, primary\_key=True)  username = db.Column(db.String(50), unique=True, nullable=False)  password = db.Column(db.String(200), nullable=False)  class Product(db.Model):  id = db.Column(db.Integer, primary\_key=True)  name = db.Column(db.String(100), nullable=False)  description = db.Column(db.Text)  price = db.Column(db.Float, nullable=False)  stock = db.Column(db.Integer, nullable=False, default=0)  # Custom Exception for Unauthorized Access  class UnauthorizedAccess(Exception):  pass  # Custom Exception for Resource Not Found  class ResourceNotFound(Exception):  pass  @app.route('/register', methods=['POST'])  def register():  data = request.json  username = data.get('username')  password = data.get('password')  if not username or not password:  abort(400, 'Both username and password are required.')  existing\_user = User.query.filter\_by(username=username).first()  if existing\_user:  abort(400, 'Username is already taken.')  hashed\_password = generate\_password\_hash(password, method='sha256')  new\_user = User(username=username, password=hashed\_password)    db.session.add(new\_user)  db.session.commit()  return jsonify({'message': 'User registered successfully.'}), 201  @app.route('/login', methods=['POST'])  def login():  data = request.json  username = data.get('username')  password = data.get('password')  if not username or not password:  abort(400, 'Both username and password are required.')  user = User.query.filter\_by(username=username).first()  if not user or not check\_password\_hash(user.password, password):  raise UnauthorizedAccess('Invalid credentials.')  # You may implement token-based authentication here  return jsonify({'message': 'Login successful.'})  @app.route('/products', methods=['GET'])  def get\_all\_products():  products = Product.query.all()  result = [  {'id': product.id, 'name': product.name, 'price': product.price, 'stock': product.stock}  for product in products  ]  return jsonify(result)  @app.route('/products/<int:product\_id>', methods=['GET'])  def get\_product(product\_id):  product = Product.query.get(product\_id)  if not product:  raise ResourceNotFound('Product not found.')  result = {'id': product.id, 'name': product.name, 'price': product.price, 'stock': product.stock}  return jsonify(result)  @app.route('/products', methods=['POST'])  def create\_product():  data = request.json  new\_product = Product(name=data['name'], description=data.get('description'), price=data['price'], stock=data['stock'])    db.session.add(new\_product)  db.session.commit()  result = {'id': new\_product.id, 'name': new\_product.name, 'price': new\_product.price, 'stock': new\_product.stock}  return jsonify(result), 201  @app.route('/products/<int:product\_id>', methods=['PUT'])  def update\_product(product\_id):  product = Product.query.get(product\_id)  if not product:  raise ResourceNotFound('Product not found.')  data = request.json  product.name = data['name']  product.description = data.get('description')  product.price = data['price']  product.stock = data['stock']    db.session.commit()  result = {'id': product.id, 'name': product.name, 'price': product.price, 'stock': product.stock}  return jsonify(result)  @app.route('/products/<int:product\_id>', methods=['DELETE'])  def delete\_product(product\_id):  product = Product.query.get(product\_id)  if not product:  raise ResourceNotFound('Product not found.')  db.session.delete(product)  db.session.commit()  return '', 204  if \_\_name\_\_ == '\_\_main\_\_':  db.create\_all()  app.run(debug=True) |

RUN application: flask run

APIs:

/register [POST]

/login [POST]

/products [GET] (Get all products)

/products/<product\_id> [GET] (Get particular product)

/products [POST] (Create a product)

/products/<product\_id> [PUT] (Update a product)

/products/<product\_id> [DELETE] (delete a product)