# IF Pemrograman Berorientasi Objek Praktik VI

Project Database

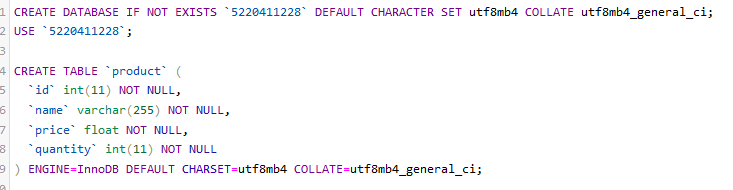


**NAMA : JA`FAR (5220411228)**

# UNIVERSITAS TEKNOLOGI YOGYAKARTA (UTY)

Jl. Siliwangi (Ringroad Utara) Jombor Kab. Sleman – D.I.Yogyakarta Tlp.(0274) 623310 & Fax : (0274) 623306

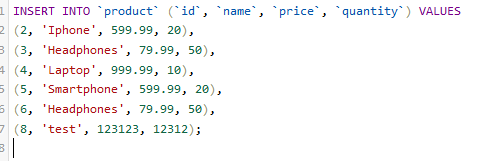
1. **Membuat Database dan Tabel**
2. Membuat Database dan Tabel product



Query ini di gunakan untuk membuat database 5220411228, dan juga di gunakan untuk membuat sebuah tabel bernama product dengan kolom-kolom berikut:

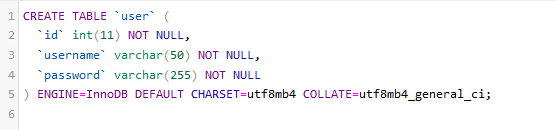
* id (tipe: int, panjang: 11): adalah kolom yang menyimpan nilai ID produk. Tipe data int(11) menunjukkan bahwa kolom menyimpan bilangan bulat dengan panjang maksimal 11 digit.
* name (tipe: varchar, panjang: 255): adalah kolom yang menyimpan nama produk. Tipe data varchar(255) menunjukkan bahwa kolom dapat menyimpan string dengan panjang maksimal 255 karakter.
* price (tipe: float): adalah kolom yang menyimpan harga produk. Tipe data float menunjukkan bahwa kolom menyimpan nilai pecahan atau bilangan desimal.
* quantity (tipe: int, panjang: 11): adalah kolom yang menyimpan jumlah produk yang tersedia. Seperti id, tipe data int(11) menunjukkan bahwa kolom menyimpan bilangan bulat dengan panjang maksimal 11 digit.

Mengisi Tabel product dengan Data.



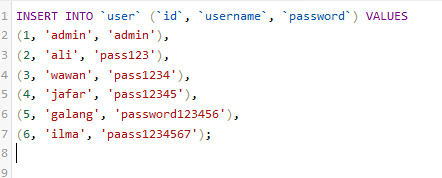
Query di atas menyisipkan beberapa baris data ke dalam tabel product dengan nilai pada kolom id, name, price, dan quantity.

1. Membuat Tabel user.



perintah SQL untuk membuat tabel bernama user

* id: Ini adalah kolom bertipe data integer dengan panjang maksimum 11 digit. NOT NULL menandakan bahwa kolom ini tidak boleh kosong dan harus selalu memiliki nilai.
* username: Ini adalah kolom bertipe data varchar(50), yang berarti dapat menyimpan string dengan panjang maksimum 50 karakter. NOT NULL juga berlaku di sini, artinya nilai kolom ini juga tidak boleh kosong.
* password: Ini adalah kolom bertipe data varchar(255), dengan panjang maksimum 255 karakter. Seperti kolom sebelumnya, NOT NULL menandakan bahwa nilai kolom ini tidak boleh kosong.

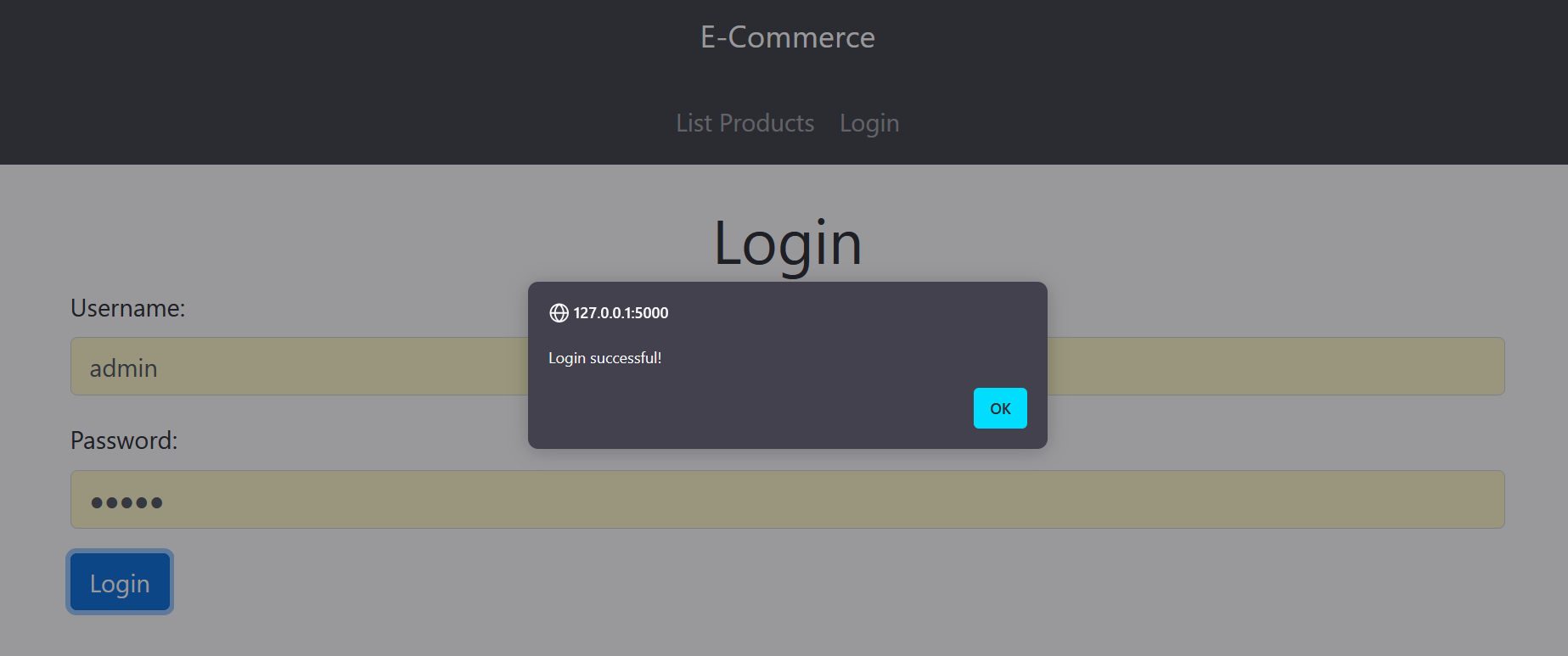


Query ini menyisipkan beberapa baris data ke dalam tabel user dengan nilai pada kolom id, username, dan password.

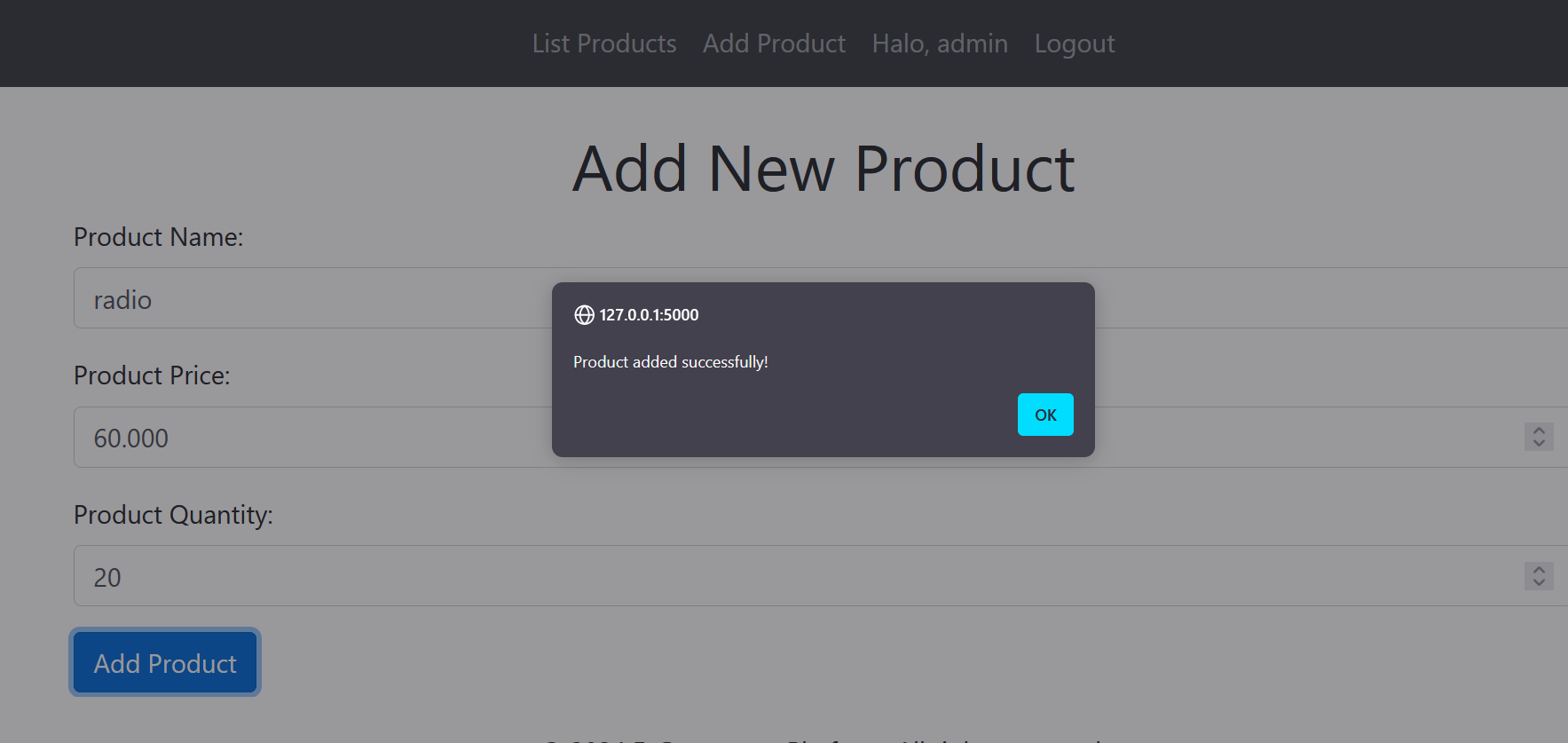
1. **Kode program app.py**

|  |
| --- |
| from flask import Flask, render\_template, request, jsonify, session, redirect, url\_for  from flask\_sqlalchemy import SQLAlchemy  from docx import Document  app = Flask(\_\_name\_\_)  app.config['SQLALCHEMY\_DATABASE\_URI'] = 'mysql://root:@localhost/5220411228'  app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False  app.config['SECRET\_KEY'] = 'test-jafar'  db = SQLAlchemy(app)  class Product(db.Model):      id = db.Column(db.Integer, primary\_key=True)      name = db.Column(db.String(255), nullable=False)      price = db.Column(db.Float, nullable=False)      quantity = db.Column(db.Integer, nullable=False)  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(255), nullable=False)  @app.route('/')  def index():      return render\_template('index.html')  @app.route('/add\_product')  def add\_product():      return render\_template('add\_product.html')  @app.route('/login')  def login\_page():      return render\_template('login.html')  @app.route('/list\_product')  def list\_product():      return render\_template('list\_product.html')  @app.route('/products', methods=['GET', 'POST'])  def products():      if request.method == 'GET':          products = Product.query.all()          product\_list = [{'id': product.id, 'name': product.name, 'price': product.price, 'quantity': product.quantity} for product in products]          return jsonify({'products': product\_list})      elif request.method == 'POST':          if 'username' not in session:              return jsonify({'error': 'User not authenticated'}), 401          data = request.get\_json()          new\_product = Product(name=data['name'], price=data['price'], quantity=data['quantity'])          db.session.add(new\_product)          db.session.commit()          products = Product.query.all()          products\_list = [{'id': product.id, 'name': product.name, 'price': product.price, 'quantity': product.quantity} for product in products]          return jsonify({'message': 'Product created successfully', 'product': {'id': new\_product.id, 'name': new\_product.name, 'price': new\_product.price, 'quantity': new\_product.quantity}, 'products': products\_list}), 201  @app.route('/products/<int:product\_id>', methods=['PUT'])  def update\_product(product\_id):      product = Product.query.get(product\_id)      if product:          data = request.get\_json()          product.name = data.get('name', product.name)          product.price = data.get('price', product.price)          product.quantity = data.get('quantity', product.quantity)          db.session.commit()          return jsonify({'message': 'Product updated successfully', 'product': {'id': product.id, 'name': product.name, 'price': product.price, 'quantity': product.quantity}})      else:          return jsonify({'message': 'Product not found'}), 404  @app.route('/products/<int:product\_id>', methods=['DELETE'])  def delete\_product(product\_id):      product = Product.query.get(product\_id)      if product:          db.session.delete(product)          db.session.commit()          return jsonify({'message': 'Product deleted successfully'}), 200      else:          return jsonify({'message': 'Product not found'}), 404  @app.route('/products/<int:product\_id>/edit', methods=['GET'])  def edit\_product\_form(product\_id):      product = Product.query.get(product\_id)      if product:          return render\_template('edit\_product.html', product=product)      else:          return jsonify({'message': 'Product not found'}), 404  @app.route('/login-api', methods=['POST'])  def login():      data = request.get\_json()      username = data.get('username')      password = data.get('password')      user = User.query.filter\_by(username=username, password=password).first()      if user:          session['user\_id'] = user.id          session['username'] = user.username          return jsonify({'message': 'Login successful'})      else:          return jsonify({'message': 'Login failed. Invalid credentials'}), 401  @app.route('/logout', methods=['GET'])  def logout():      session.clear()      return redirect(url\_for('index'))  @app.route('/check-auth', methods=['GET'])  def check\_authentication():      if 'user\_id' in session:          return jsonify({'authenticated': True, 'username': session['username']})      else:          return jsonify({'authenticated': False})  def create\_word\_document():      document = Document()      document.add\_heading('Documentation - PBO Program', 0)      document.add\_heading('Database Structure', level=1)      document.add\_paragraph('Database Name: 5220411228')      document.add\_heading('Product Table', level=2)      document.add\_paragraph('Columns: id, name, price, quantity')      document.add\_heading('User Table', level=2)      document.add\_paragraph('Columns: id, username, password')      document.add\_heading('Program Code', level=1)      with open(\_\_file\_\_, 'r') as file:          code = file.read()          document.add\_paragraph(code)      document.add\_heading('Program Execution Results', level=1)      # Add screenshots or descriptions of insert, update, delete, select operations      document.save('PBO\_Program\_Documentation.docx')  if \_\_name\_\_ == '\_\_main\_\_':      with app.app\_context():          db.create\_all()          create\_word\_document()      app.run(debug=True) |

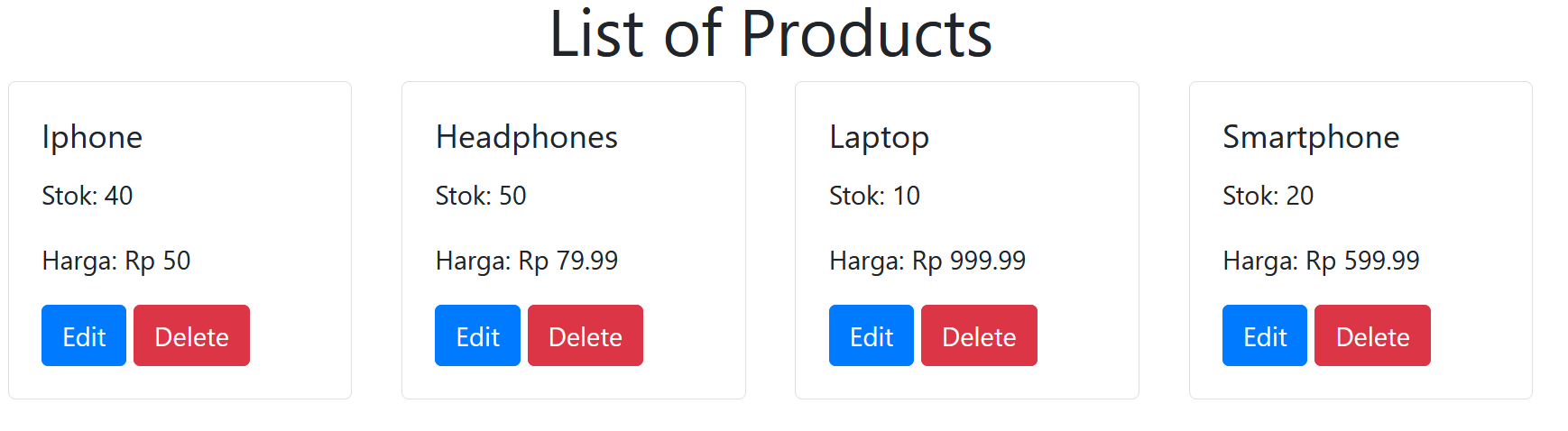
1. **Hasil running program insert, update, delete.**
2. Select.

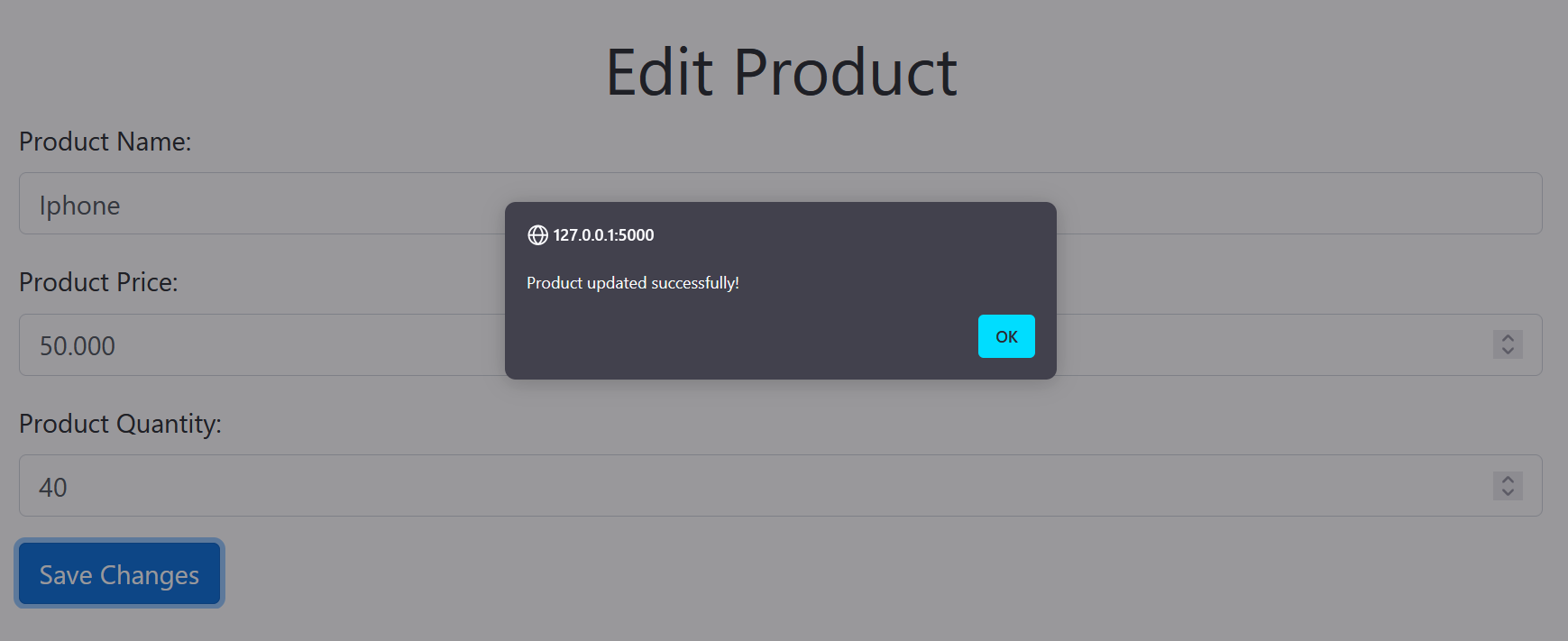


1. Insert

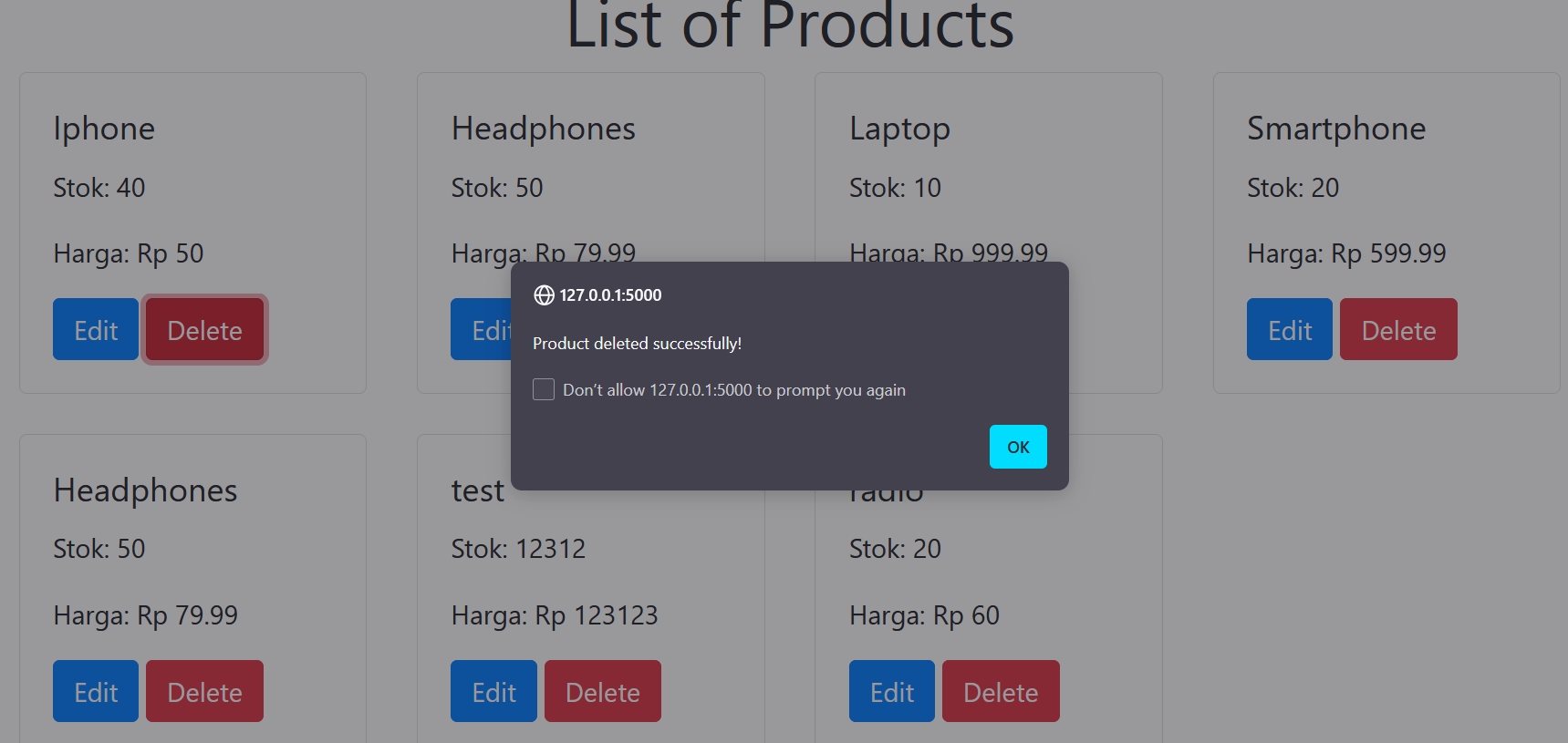


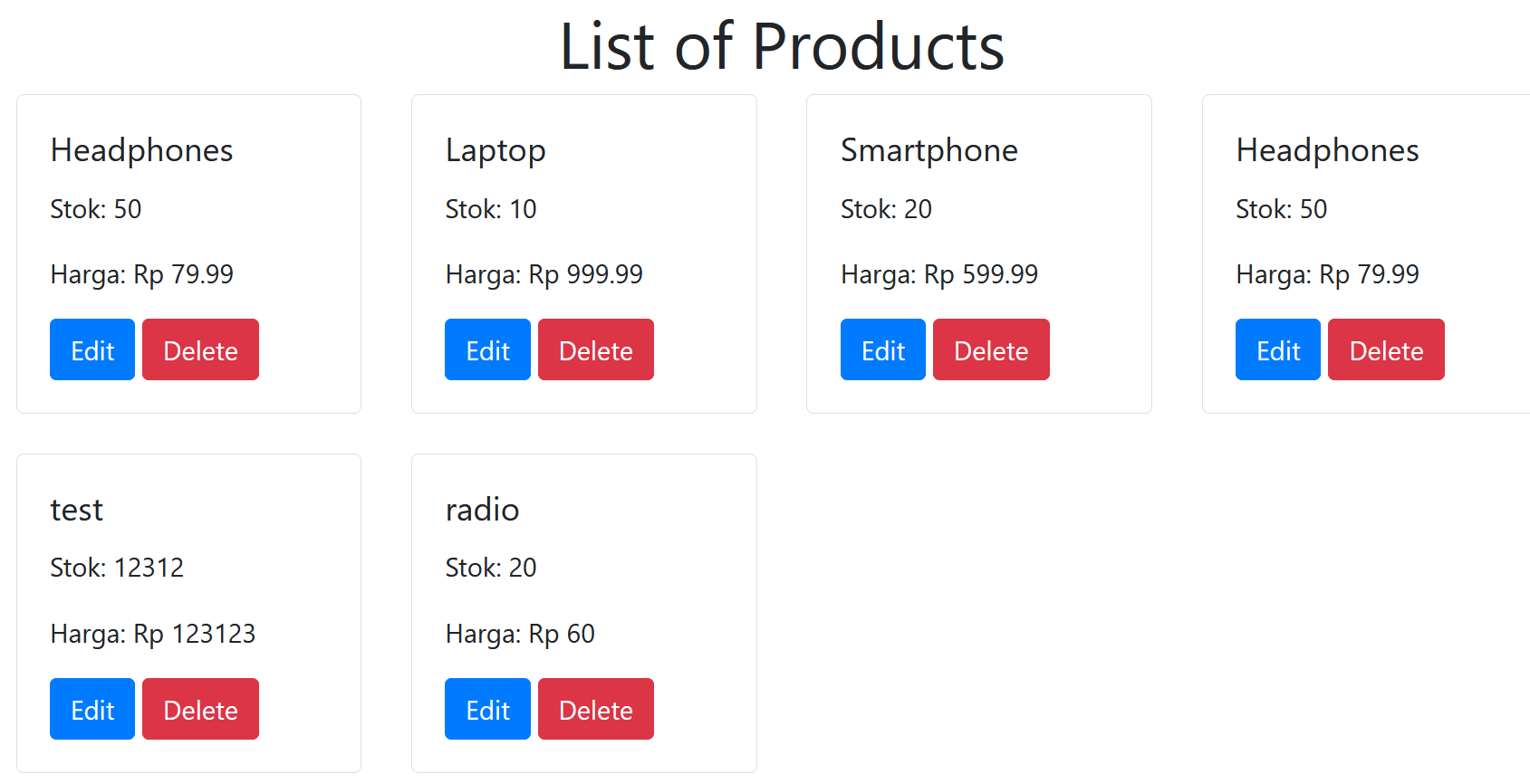
1. Update

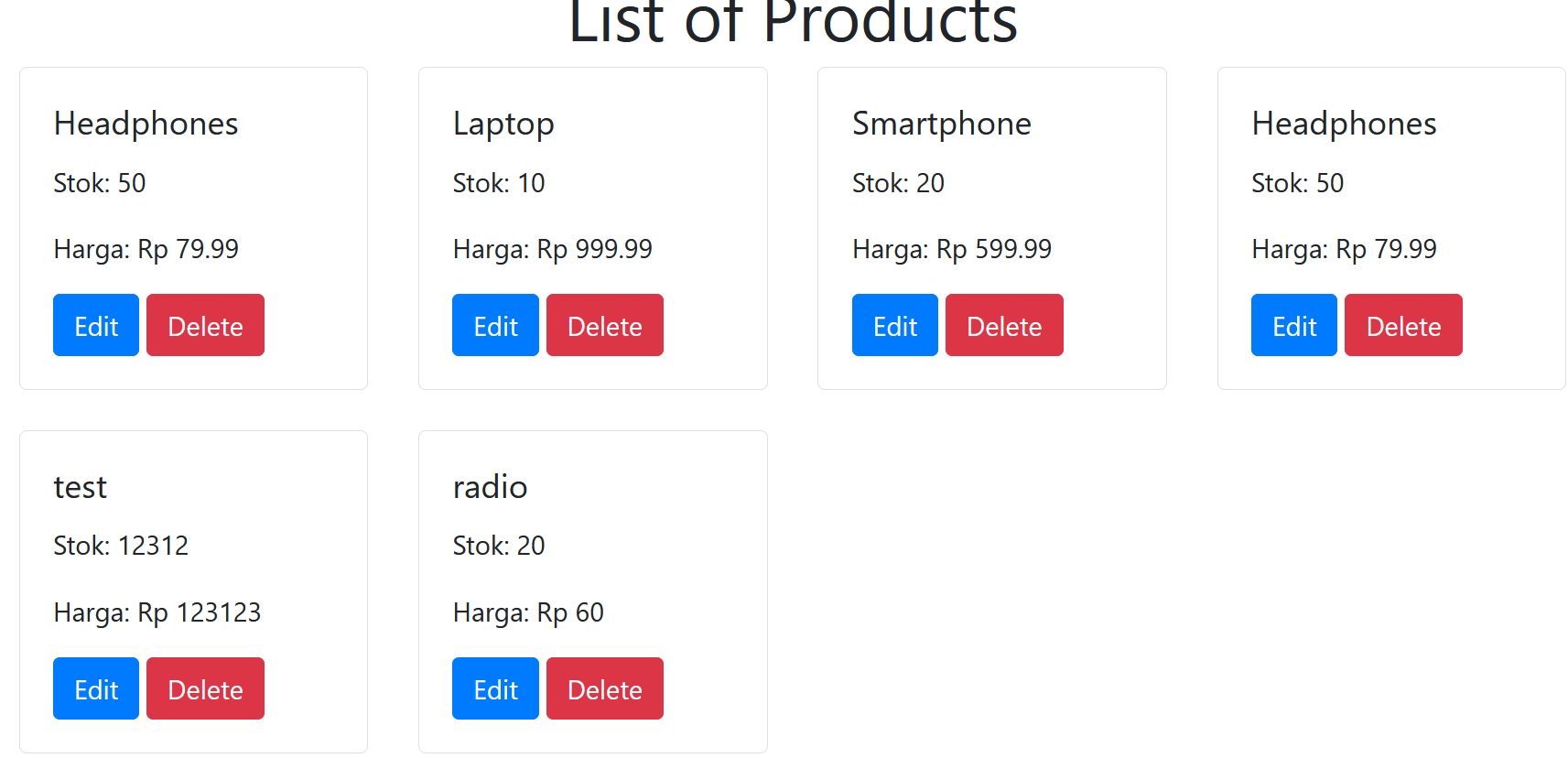




1. Delete







1. **Hasil running program select**

