**Assessment**

1. Implementing the Singleton Pattern

This is the code for the above program mentioned where only one instance can be created and if an further option to create an object will return to the same instance.

class Singleton {

constructor() {

if (Singleton.instance) {

return Singleton.instance;

}

this.data = "I am the only instance";

Singleton.instance = this;

return this;

}

getData() {

return this.data;

}

setData(newData) {

this.data = newData;

}

}

1. Implementing the Factory Method Pattern

Here, shape is the base class interface. Circle and square are concrete implementations. Shapefactory is the method where class instantiates different shapes based on input type.

// Product Interface

class Shape {

draw() {

throw new Error("This method should be overridden!");

}

}

// Concrete Products

class Circle extends Shape {

draw() {

console.log("Drawing a Circle");

}

}

class Square extends Shape {

draw() {

console.log("Drawing a Square");

}

}

// Creator (Factory)

class ShapeFactory {

createShape(type) {

switch (type) {

case "circle":

return new Circle();

case "square":

return new Square();

default:

throw new Error("Invalid shape type");

}

}

}

1. E-commerce Platform Search Function

// productDatabase.js

const products = [

{ id: 1, name: "Wireless Mouse", category: "Electronics" },

{ id: 2, name: "Bluetooth Headphones", category: "Electronics" },

{ id: 3, name: "Cotton T-shirt", category: "Clothing" },

{ id: 4, name: "Running Shoes", category: "Footwear" },

{ id: 5, name: "LED Monitor", category: "Electronics" },

{ id: 6, name: "Jeans Pants", category: "Clothing" }

];

module.exports = products;

// searchFunction.js

const products = require('./productDatabase');

function searchProducts(query) {

const lowercaseQuery = query.toLowerCase();

return products.filter(product =>

product.name.toLowerCase().includes(lowercaseQuery) ||

product.category.toLowerCase().includes(lowercaseQuery)

);

}

// Example usage:

const readline = require('readline-sync');

const userInput = readline.question("Search for a product: ");

const results = searchProducts(userInput);

if (results.length > 0) {

console.log("Search Results:");

results.forEach(product =>

console.log(`- ${product.name} [${product.category}]`)

);

} else {

console.log("No products found.");

}

1. Financial Forecasting

It use historical monthly revenue data, applies linear regression and also predicts revenue for upcoming months.

# financial\_forecast.py

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Sample historical data (e.g., monthly revenue in USD)

months = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1) # January to June

revenue = np.array([1000, 1200, 1300, 1500, 1700, 1800]) # Revenue per month

# Create and train the model

model = LinearRegression()

model.fit(months, revenue)

# Predict revenue for future months (e.g., July to December)

future\_months = np.array([7, 8, 9, 10, 11, 12]).reshape(-1, 1)

predicted\_revenue = model.predict(future\_months)

# Print predictions

print("📈 Forecasted Revenue:")

for month, revenue in zip(range(7, 13), predicted\_revenue):

print(f"Month {month}: ${revenue:.2f}")