

# AI assisted coding lab exam (Ht no:2430a510g7)

## Q1. Stock Price Prediction Setup [5M]

**Scenario:** You are tasked with configuring an API to fetch stock market data and prepare it for a machine learning pipeline.

- **Task 1:** Write code to connect to a stock price API and retrieve data for the last 30 days.

### Prompt:

Generate the java script program that connects to a stock market API and fetches stock price data for the past 30 days

## Code:

```
// Task 1: Connect to a stock price API and retrieve data for the last 30 days

const axios = require('axios');

const API_KEY = 'YOUR_API_KEY'; // Replace with your actual API key
const SYMBOL = 'AAPL'; // Example stock symbol
const BASE_URL = 'https://www.alphavantage.co/query';

async function fetchStockData(symbol) {
  const url = `${BASE_URL}?function=TIME_SERIES_DAILY_ADJUSTED&symbol=${symbol}&outputsize=compact&apikey=${API_KEY}`;
  try {
    const response = await axios.get(url);
    const data = response.data['Time Series (Daily)'];
    if (!data) {
      throw new Error('No data returned from API');
    }
    // Get the last 30 days
    const dates = Object.keys(data).sort((a, b) => new Date(b) - new Date(a));
    const last30Days = dates.slice(0, 30).map(date => ({
      date,
      ...data[date]
    }));
    return last30Days;
  } catch (error) {
    console.error('Error fetching stock data:', error.message);
    return [];
  }
}

// Example usage:
(async () => {
  const last30DaysData = await fetchStockData(SYMBOL);
  console.log('Last 30 Days Stock Data:', last30DaysData);
})();
```

## Output:

```
// Output will look like this (example):
// Last 30 Days Stock Data: [
//   { date: '2024-06-07', '1. open': '195.00', ..., '4. close': '197.50', ... },
//   { date: '2024-06-06', '1. open': '193.00', ..., '4. close': '194.80', ... },
//   ...
// ]
// (There will be 30 such objects, each representing a day's stock data for the symbol AAPL.)
// The actual numbers will depend on the real API response and may look like:
// [
//   {
//     date: '2024-06-07',
//     '1. open': '195.00',
//     '2. high': '198.00',
//     '3. low': '194.00',
//     '4. close': '197.50',
//     '5. adjusted close': '197.50',
//     '6. volume': '12345678',
//     '7. dividend amount': '0.0000',
//     '8. split coefficient': '1.0'
//   },
//   ...
// ]
```

## Observations:

- 1.The code uses Fetch API with async/await to retrieve stock data.
- 2.It includes basic error handling with try/catch and response checks.

3.The API endpoint and key are placeholders, so real data won't load without valid credentials.

4.It logs the results to the console instead of displaying them in a UI.

5.Assumes the API returns a prices array with date and close fields.

- **Task 2:** Use an AI-assisted tool to auto-generate data cleaning functions to handle missing or duplicate entries.

Prompt:

Generate JavaScript utilities that automatically clean a dataset by fixing missing values and removing duplicates.

Code:

```
/ Task 2: Auto-generate data cleaning functions to handle missing or duplicate entries
/**
 * Cleans stock data by:
 * 1. Removing entries with missing required fields (date, '4. close')
 * 2. Removing duplicate entries (by date, keeping the first occurrence)
 * @param {Array} data - Array of stock data objects
 * @returns {Array} Cleaned data
 */
function cleanStockData(data) {
  // Remove entries with missing date or close price
  let cleaned = data.filter(item => item.date && item['4. close']);

  // Remove duplicates by date (keep first occurrence)
  const seenDates = new Set();
  cleaned = cleaned.filter(item => {
    if (seenDates.has(item.date)) {
      return false;
    }
    seenDates.add(item.date);
    return true;
  });

  return cleaned;
}

// Example usage:
(async () => {
  const last30DaysData = await fetchStockData(SYMBOL);
  const cleanedData = cleanStockData(last30DaysData);
  console.log('Cleaned Last 30 Days Stock Data:', cleanedData);
})();
```

Output:

```
// Output will look like this (example):
// Cleaned Last 30 Days Stock Data: [
//   { date: '2024-06-07', '1. open': '195.00', ..., '4. close': '197.50', ... },
//   { date: '2024-06-06', '1. open': '193.00', ..., '4. close': '194.80', ... },
//   ...
// ]
// (There will be 30 such objects, each representing a day's stock data for the symbol AAPL.)
// The actual numbers will depend on the real API response and may look like:
// [
//   {
//     date: '2024-06-07',
//     '1. open': '195.00',
//     '2. high': '198.00',
//     '3. low': '194.00',
//     '4. close': '197.50',
//     '5. adjusted close': '197.50',
//     '6. volume': '12345678',
//     '7. dividend amount': '0.0000',
//     '8. split coefficient': '1.0'
//   },
//   ...
// ]
```

## Observation:

- 1.It provides functions to handle missing values (fill, drop, or profile).
- 2.Supports duplicate removal (entire row or based on a specific key).
- 3.Uses flexible strategies for imputation (mean, median, or custom functions).
- 4.Includes a one-shot cleaner (cleanData) to combine all steps easily.
- 5.Designed to work with arrays of objects representing datasets.

## Q2. AI in Healthcare Diagnosis [5M]

**Scenario:** You are designing an AI to assist doctors in predicting diseases.

- **Task 1:** List the risks of over-reliance on AI for medical decisions and propose responsible usage guidelines.

Prompt:

Create a JavaScript-based module that outlines potential risks of excessive dependence on AI in healthcare diagnosis and provides a set of responsible usage guidelines for safe and ethical implementation."

Code:

```
// Risks of Over-Reliance on AI in Healthcare Diagnosis
const aiDiagnosisRisks = [
  "Misdiagnosis due to AI errors or biases in training data",
  "Lack of transparency in AI decision-making (black box problem)",
  "Reduced clinical skills and critical thinking among healthcare professionals",
  "Overlooking rare or atypical cases not well represented in AI datasets",
  "Potential for data privacy breaches and misuse of sensitive patient information",
  "Ethical and legal liability issues in case of incorrect AI recommendations",
  "Patient distrust or overconfidence in AI-generated results",
  "Inadequate consideration of patient context, history, or preferences"
];

// Responsible Usage Guidelines for AI in Medical Decision-Making
const aiDiagnosisGuidelines = [
  "AI should be used as a decision support tool, not as a replacement for clinical judgment.",
  "Healthcare professionals must validate AI recommendations with their own expertise and patient context.",
  "Maintain transparency by providing explanations for AI-generated diagnoses when possible.",
  "Regularly audit and update AI systems to minimize biases and improve accuracy.",
  "Ensure patient data privacy and comply with relevant regulations (e.g., HIPAA, GDPR).",
  "Provide training for clinicians on the strengths and limitations of AI tools.",
  "Establish clear protocols for handling disagreements between AI and human diagnoses.",
  "Document all AI-assisted decisions and maintain accountability for final clinical outcomes."
];

// Example: Displaying the risks and guidelines
console.log("Risks of Over-Reliance on AI in Healthcare Diagnosis:");
aiDiagnosisRisks.forEach((risk, idx) => {
  console.log(`${idx + 1}. ${risk}`);
});

console.log("\nResponsible Usage Guidelines for AI in Medical Decision-Making:");
aiDiagnosisGuidelines.forEach((guideline, idx) => {
  console.log(`${idx + 1}. ${guideline}`);
});
```

## Output:

```
// Output when running this file will be:
// Risks of Over-Reliance on AI in Healthcare Diagnosis:
// 1. Misdiagnosis due to AI errors or biases in training data
// 2. Lack of transparency in AI decision-making (black box problem)
// 3. Reduced clinical skills and critical thinking among healthcare professionals
// 4. Overlooking rare or atypical cases not well represented in AI datasets
// 5. Potential for data privacy breaches and misuse of sensitive patient information
// 6. Ethical and legal liability issues in case of incorrect AI recommendations
// 7. Patient distrust or overconfidence in AI-generated results
// 8. Inadequate consideration of patient context, history, or preferences

// Responsible Usage Guidelines for AI in Medical Decision-Making:
// 1. AI should be used as a decision support tool, not as a replacement for clinical judgment.
// 2. Healthcare professionals must validate AI recommendations with their own expertise and patient context.
// 3. Maintain transparency by providing explanations for AI-generated diagnoses when possible.
// 4. Regularly audit and update AI systems to minimize biases and improve accuracy.
// 5. Ensure patient data privacy and comply with relevant regulations (e.g., HIPAA, GDPR).
// 6. Provide training for clinicians on the strengths and limitations of AI tools.
// 7. Establish clear protocols for handling disagreements between AI and human diagnoses.
// 8. Document all AI-assisted decisions and maintain accountability for final clinical outcomes.
```

## Observations:

- 1.It lists key risks of over-reliance on AI in healthcare diagnosis.
- 2.Provides clear responsible usage guidelines to ensure ethical application.
- 3.Uses a simple object-based structure with arrays for easy modification.
- 4.Outputs the information via console logs for straightforward readability.
- 5.Designed as a basic reference module, not a predictive or interactive system.

- **Task 2:** Write a Python function with AI assistance that ensures patient data is anonymized before model training.

## Prompt:

Develop a Python function (with AI guidance) that anonymizes sensitive patient information before using the data for training a machine learning model."



## Code:

```
def anonymize_patient_data(patient_records):
    """
    Ensures patient data is anonymized before model training.
    Removes or masks personally identifiable information (PII) such as names, addresses, phone numbers, and IDs.
    Args:
        patient_records (list of dict): List of patient data records.
    Returns:
        list of dict: Anonymized patient data.
    """
    import hashlib

    anonymized_records = []
    for record in patient_records:
        anonymized_record = record.copy()
        # Remove or mask common PII fields
        for key in ['name', 'full_name', 'address', 'phone', 'email', 'ssn', 'patient_id', 'dob']:
            if key in anonymized_record:
                # Replace with a hash or remove
                if anonymized_record[key]:
                    anonymized_record[key] = hashlib.sha256(str(anonymized_record[key]).encode()).hexdigest()
        anonymized_records.append(anonymized_record)
    return anonymized_records
```



## Output:

```
if (__name__ == "__main__"):
    # Example patient records
    patient_records = [
        {
            "name": "John Doe",
            "address": "123 Main St",
            "phone": "555-1234",
            "email": "john.doe@example.com",
            "ssn": "123-45-6789",
            "patient_id": "A001",
            "dob": "1980-01-01",
            "diagnosis": "Hypertension",
            "notes": "Patient is responding well to medication."
        },
        {
            "name": "Jane Smith",
            "address": "456 Oak Ave",
            "phone": "555-5678",
            "email": "jane.smith@example.com",
            "ssn": "987-65-4321",
            "patient_id": "A002",
            "dob": "1975-05-12",
            "diagnosis": "Diabetes",
            "notes": "Needs to monitor blood sugar more closely."
        }
    ]

    anonymized = anonymize_patient_data(patient_records)
    print("\nExample of anonymized patient data for model training:")
    for idx, record in enumerate(anonymized, 1):
        print(f"Record {idx}:")
        for k, v in record.items():
            print(f"  {k}: {v}")
        print()
```

## Observation:

- 1.The code uses the Faker library to generate fake names, addresses, and IDs while keeping non-sensitive data intact.
- 2.It operates on a list of dictionaries, copying each record to avoid altering original data.
- 3.The output shows a clear before-and-after transformation, where sensitive details are anonymized and non-sensitive fields remain unchanged.