# tests/test\_data\_parser.py

import unittest

from unittest.mock import mock\_open, patch

from utils.data\_parser import parse\_simulation\_log

class TestDataParser(unittest.TestCase):

@patch('builtins.open', new\_callable=mock\_open, read\_data="""

Instructions Executed: 150

Stalls: 5

Branch Mispredictions: 2

Hello, RISC-V!

""")

def test\_parse\_simulation\_log\_success(self, mock\_file):

log\_path = '/fake/log/simulation\_log.txt'

data\_path = '/fake/data/simulation\_data.json'

expected\_data = {

'instructions\_executed': 150,

'stalls': 5,

'branch\_mispredictions': 2,

'message': "Hello, RISC-V!",

'timestamp': '2024-04-27T12:00:00Z' # Example timestamp

}

with patch('utils.data\_parser.datetime') as mock\_datetime:

mock\_datetime.utcnow.return\_value.isoformat.return\_value = '2024-04-27T12:00:00Z'

data = parse\_simulation\_log(log\_path, data\_path)

self.assertEqual(data['instructions\_executed'], 150)

self.assertEqual(data['stalls'], 5)

self.assertEqual(data['branch\_mispredictions'], 2)

self.assertEqual(data['message'], "Hello, RISC-V!")

self.assertEqual(data['timestamp'], '2024-04-27T12:00:00Z')

@patch('builtins.open', new\_callable=mock\_open, read\_data="Invalid Log Content")

def test\_parse\_simulation\_log\_partial(self, mock\_file):

log\_path = '/fake/log/simulation\_log.txt'

data\_path = '/fake/data/simulation\_data.json'

expected\_data = {

'timestamp': '2024-04-27T12:00:00Z'

}

with patch('utils.data\_parser.datetime') as mock\_datetime:

mock\_datetime.utcnow.return\_value.isoformat.return\_value = '2024-04-27T12:00:00Z'

data = parse\_simulation\_log(log\_path, data\_path)

self.assertEqual(data['timestamp'], '2024-04-27T12:00:00Z')

self.assertNotIn('instructions\_executed', data)

self.assertNotIn('stalls', data)

self.assertNotIn('branch\_mispredictions', data)

self.assertNotIn('message', data)

@patch('builtins.open', side\_effect=FileNotFoundError)

def test\_parse\_simulation\_log\_missing\_file(self, mock\_file):

log\_path = '/fake/log/missing\_log.txt'

data\_path = '/fake/data/simulation\_data.json'

with self.assertRaises(FileNotFoundError):

parse\_simulation\_log(log\_path, data\_path)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()