1. Set up your project structure:

Create a new directory for your project.

Inside the project directory, create a src/main/scala directory.

Navigate to the project directory and create a new file named build.sbt.

1. Open the build.sbt file and add the following content:

name := "IPFetcher"

version := "1.0"

scalaVersion := "2.13.6"

libraryDependencies ++= Seq(

"dev.zio" %% "zio" % "1.0.12",

"dev.zio" %% "zio-test" % "1.0.12" % Test,

"dev.zio" %% "zio-test-sbt" % "1.0.12" % Test

)

1. Create a new Scala file named IPFetcher.scala inside the src/main/scala directory, and add the following code to it:

import zio.\_

import zio.console.\_

import zio.blocking.\_

import io.circe.parser.\_

object IPFetcher extends App {

private val apiUrl = "https://api.ipify.org/?format=json"

def fetchIP: ZIO[Blocking, Throwable, String] =

blocking(Task.effect(scala.io.Source.fromURL(apiUrl).mkString))

def extractIP(response: String): Either[Throwable, String] =

for {

json <- parse(response)

ip <- json.hcursor.downField("ip").as[String]

} yield ip

def run(args: List[String]): URIO[ZEnv, ExitCode] =

fetchIP.foldM(

error => putStrLn(s"Error retrieving IP address: ${error.getMessage}").as(ExitCode.failure),

response =>

ZIO.fromEither(extractIP(response)).foldM(

error => putStrLn(s"Error extracting IP address: ${error.getMessage}").as(ExitCode.failure),

ip => putStrLn(s"Your IP address is: $ip").as(ExitCode.success)

)

)

}

1. Create a new Scala file named IPFetcherSpec.scala inside the src/test/scala directory, and add the following code to it:

import zio.\_

import zio.test.\_

import zio.test.Assertion.\_

import zio.test.environment.TestConsole

import zio.blocking.Blocking

object IPFetcherSpec extends DefaultRunnableSpec {

def spec: ZSpec[Environment, Failure] =

suite("IPFetcher")(

testM("extractIP should extract IP address from valid JSON") {

val validJson =

"""

|{

| "ip": "192.168.0.1"

|}

|""".stripMargin

val result = IPFetcher.extractIP(validJson)

assertM(ZIO.fromEither(result))(equalTo("192.168.0.1"))

},

testM("extractIP should fail for invalid JSON") {

val invalidJson = "Invalid JSON"

val result = IPFetcher.extractIP(invalidJson)

assertM(ZIO.fromEither(result))(fails(anything))

},

testM("fetchIP should succeed with mocked response") {

val mockedResponse = """{"ip": "192.168.0.1"}"""

val testEnvironment = TestEnvironment.test(

blocking = Blocking.Service.live,

console = TestConsole.Service.live

)(\_.putStrLn(mockedResponse))

val result = IPFetcher.fetchIP.provideLayer(testEnvironment)

assertM(result)(equalTo(mockedResponse))

},

testM("fetchIP should fail if connection fails") {

val testEnvironment = TestEnvironment.test(

blocking = Blocking.Service.live,

console = TestConsole.Service.live

)(\_.fail("Connection error"))

val result = IPFetcher.fetchIP.provideLayer(testEnvironment).either

assertM(result)(isLeft)

}

)

}

1. Save the files, and navigate to the project directory in the terminal.
2. Run the following command to compile the project:

sbt compile ------🡪python

1. After the compilation is successful, you can run the tool with the following command:

sbt test

1. The tests will execute without making any actual network requests, ensuring good test coverage for the IPFetcher functionality.
2. Note: This assessment uses the ZIO Test library to write and run tests. The IPFetcherSpec defines a suite of tests that cover the extraction of the IP address from JSON and the fetching of the IP address from the API. The tests use a test environment to provide mocked dependencies for blocking operations and console output.