**Automated Testing**Automated Testing is the practice of writing code to test the code then run those tests in automated test. In contrast to manual testing where we manually click and test through the app

**Code = Production Code + Test Code**

**Types of Automated Tests  
Unit Tests  
Integration Tests  
End To End Tests**

**Unit Tests: -** Test a unit of application without its external dependencies such as files, databases etc. very fast to execute

**Integration Tests: -** Test the application with its external dependencies. It tests the integration of your code with external dependencies. Such as integration with database. Takes Longer to execute

**End To End Tests: -** Drives an application through its UI. Check the User Interface by mimicking User interaction. Very slow

**Unit Testing With Jest**

To Write Test with Jest You need to create a file with name of your file.test.js for example if your code file is code.js then the test file should be code.test.js

Import the code file in test file

To run Jest Test Go to Package.json file and add a test script

  "scripts": {

    "test": "jest"

  },

Example of testing a sum function

sum.js file

function sum(a, b) {

  return a + b

}

module.exports = sum

sum.test.js file

const sum = require('./sum')

test('add 1+2 equals to 3', () => {

  expect(sum(2, 2)).toBeGreaterThan(3)

  expect(sum(2, 2)).toBeGreaterThanOrEqual(3.5)

  expect(sum(2, 2)).toBeLessThan(5)

  expect(sum(2, 2)).toBeLessThanOrEqual(4.5)

})

To write define a test function that takes a description of the test and and a callback function. In which you call except function and pass the function with input you want to call. This test used expect and different matchers to test the conditions. These chain function are called matchers and they match the output of the function called to your expected output.

For example toBeGreaterThan matches the output 4 should be greater than expected output 3

**Testing Numbers With Jest**

Test cases for one function can be grouped together using describe statement and test statement can also be replace by it statement. For Example

module.exports.absolute = function (number) {

  return number >= 0 ? number : -number

}

describe('absolute', () => {

  it('should return a positive number if input is positive', () => {

    const result = lib.absolute(1)

    expect(result).toBe(1)

  })

  it('should return a positive number if input is negative', () => {

    const result = lib.absolute(-1)

    expect(result).toBe(1)

  })

  it('should return a 0 if input is 0', () => {

    const result = lib.absolute(0)

    expect(result).toBe(0)

  })

})

These following matchers can be used with Numbers

  toBeGreaterThan()

  toBeGreaterThanOrEqual()

  toBeLessThan()

  toBeLessThanOrEqual()  
 ToBe()

**Testing String With Jest**

ToBe() can also be used to match string but it will match exact but if you want to match just a specific substring should be there in string then toMatch(//) can be used that use regex or toContain() can be used

module.exports.greet = function (name) {

  return `Welcome ${name}`

}

describe('greet', () => {

  it('should return a greeting message', () => {

    expect(lib.greet('rishabh')).toMatch(/rishabh/)

  })

})

test checks for if greet message have rishabh in it

**Testing Arrays With Jest**

module.exports.getCurrencies = function () {

  return ['USD', 'AUD', 'EUR']

}

describe('getCurrencies', () => {

  it('should return supported currencies', () => {

    const result = lib.getCurrencies()

    expect(result).toEqual(expect.arrayContaining(['EUR', 'USD', 'AUD']))

  }

Matches returning array should contain these currencies if there are other currencies added the test wont break