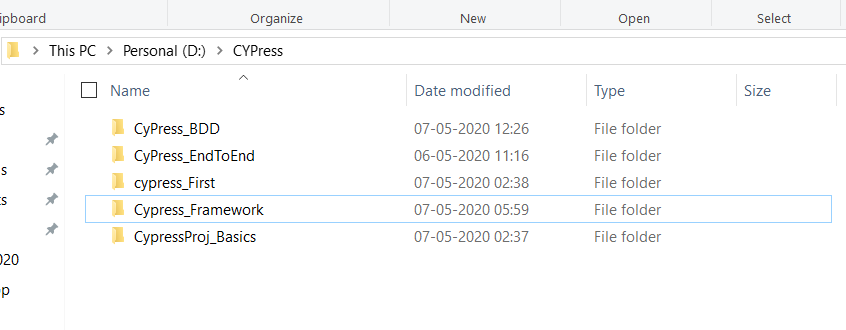
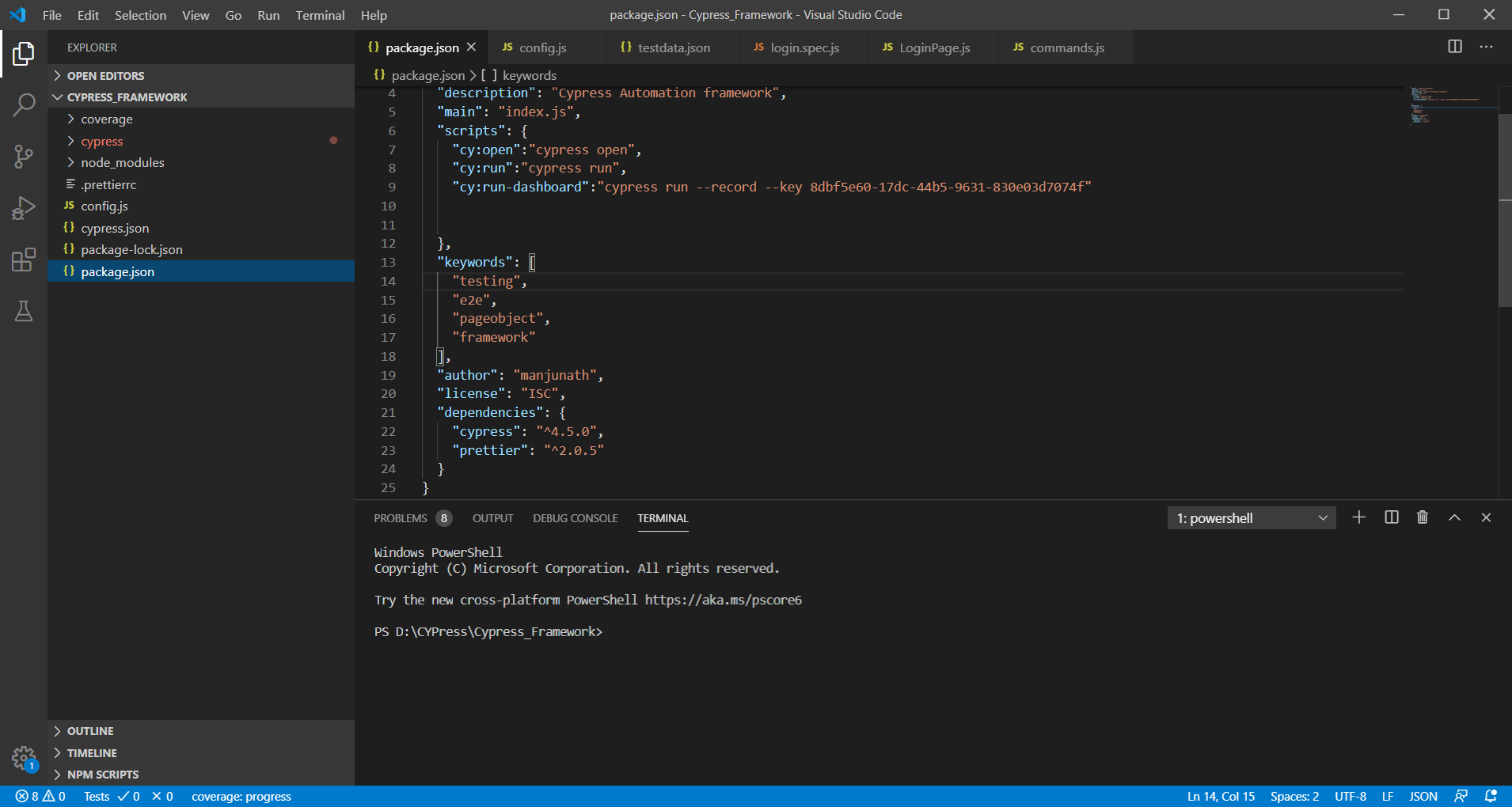
Cy Press Framework:

Create a dummy project and open in vsc:

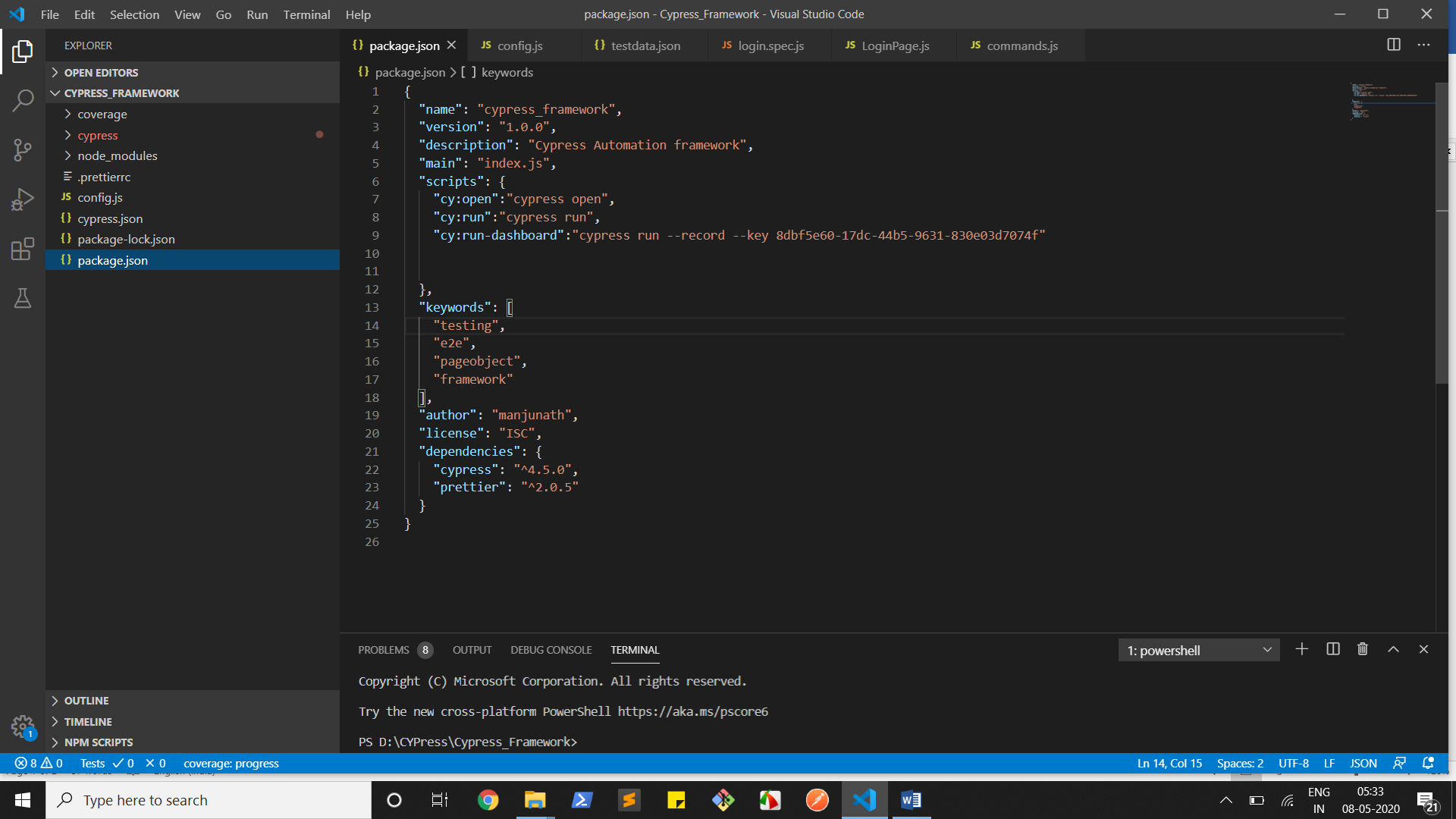
Open VSC editor->



Click on Terminal-> new terminal->

1. npm init
2. npm install cypress
3. Npm install prettier

Now open the Project > Package .json



Under framework folder leve create one file .prettierrc.js file

And copy paste this content;

{

"semi": false,

"singleQuote": true,

"useTabs": true,

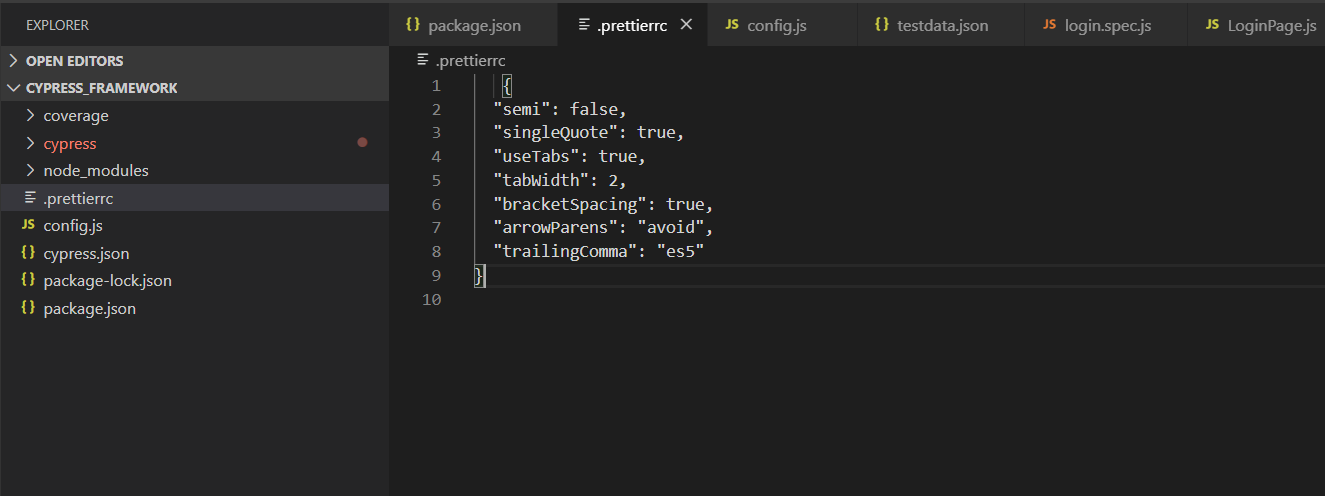
"tabWidth": 2,

"bracketSpacing": true,

"arrowParens": "avoid",

"trailingComma": "es5"

}



Then go to package.json file and under scripts add below piece of code

"scripts": {

    "cy:open":"cypress open",

    "cy:run":"cypress run",

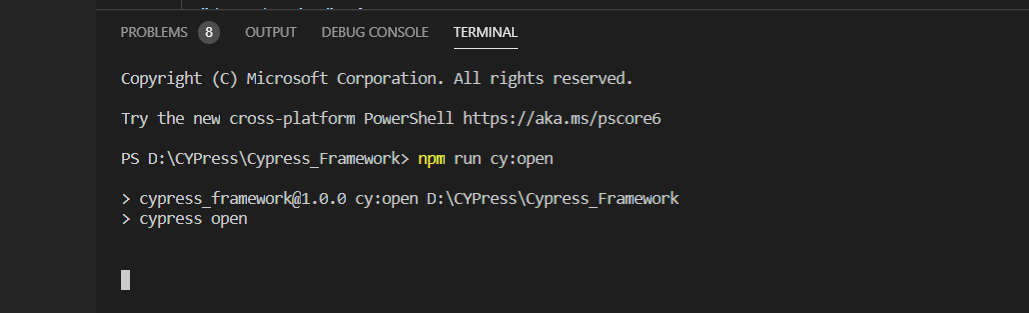
    }

    "cy:open":"cypress open", It will open in Normal UI Mode with all default examples.

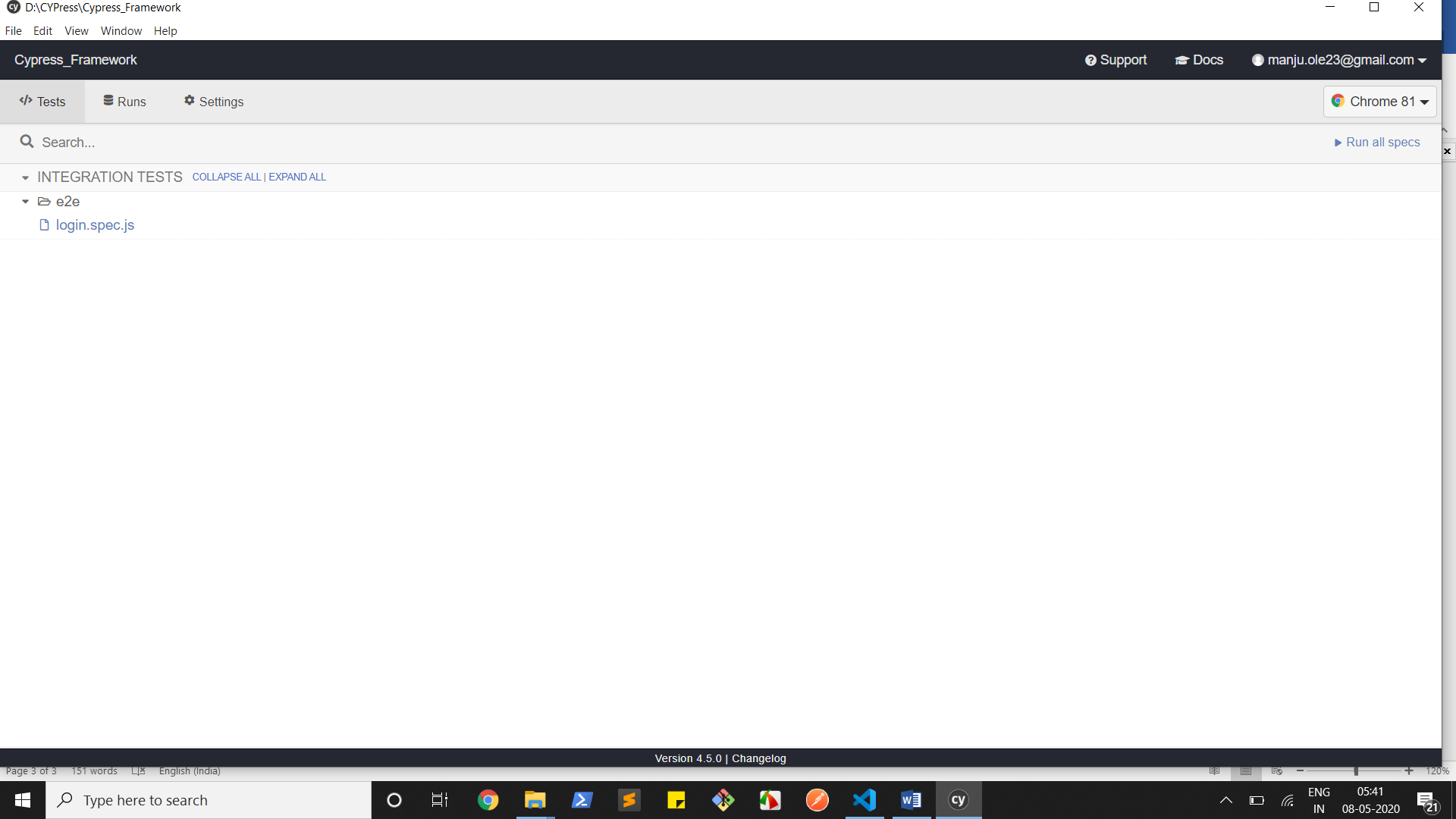
    "cy:run":"cypress run", to execute in headless mode we are are going to use this.

What this commands does mean-. It will go to node\_modules and check the cypress folder and then execute the open/ run based on the condition

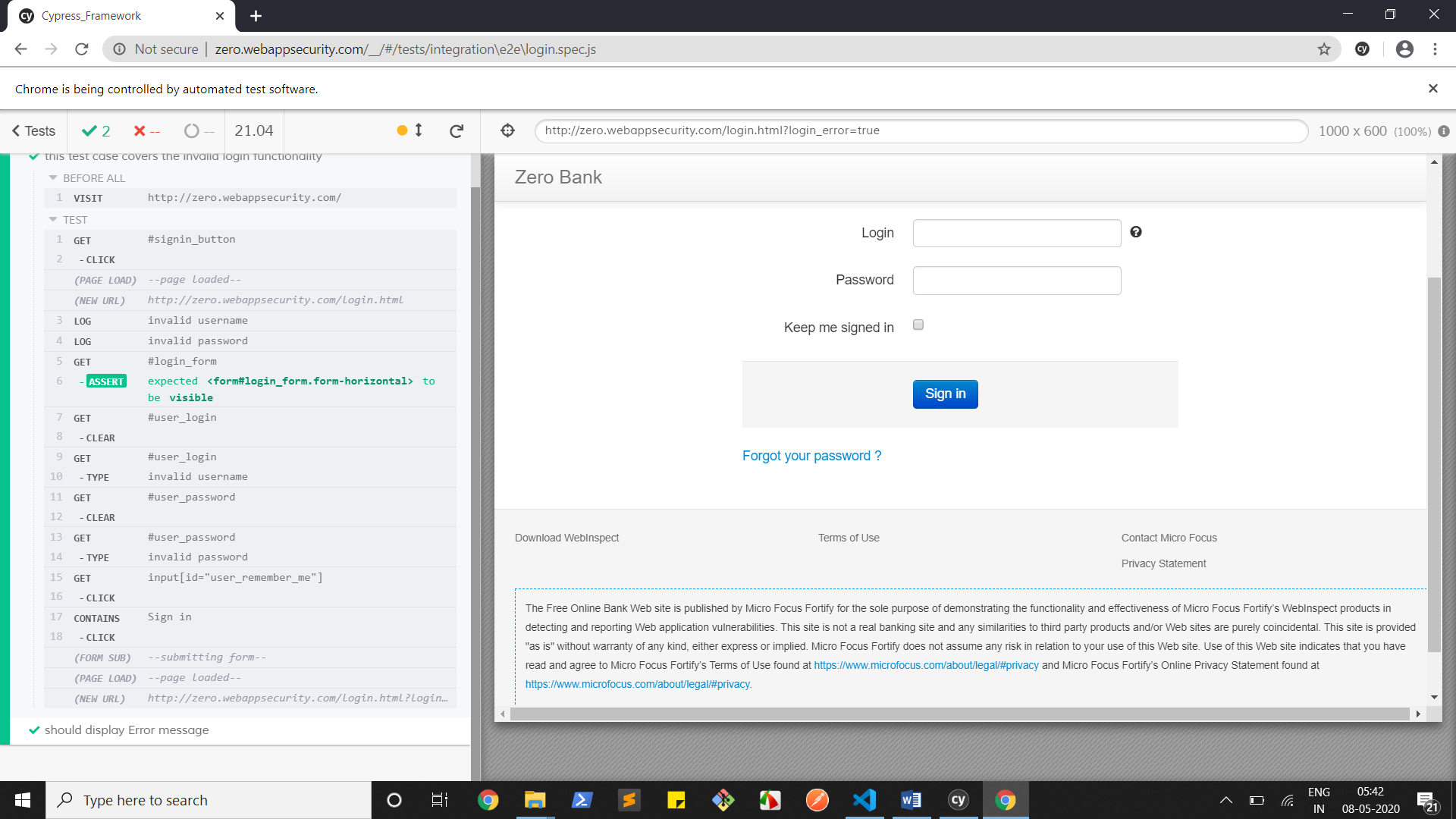
Now in cmd Enter npm run cy:open



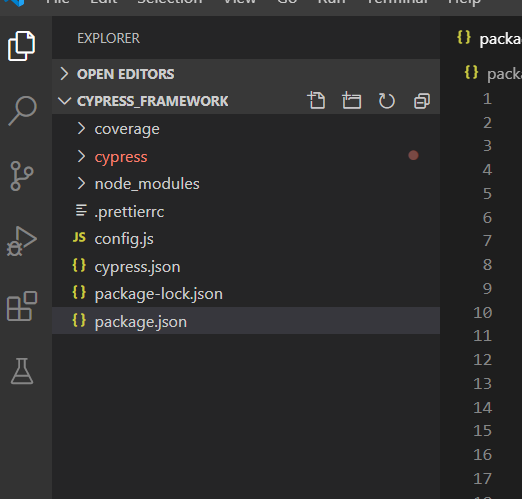
UI will open like this



Click on Login.spec.js

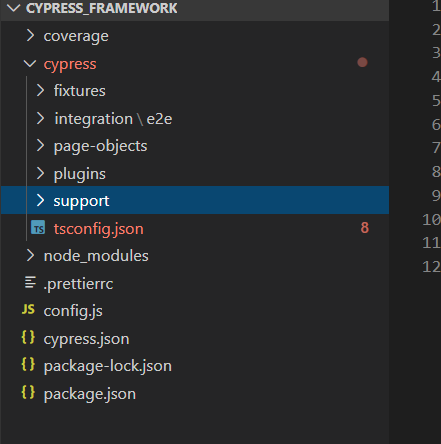


If we execute cy:open command by default all framework related modules will be downloaded and and we can see the folder structure.



1. Cypress.json file will be created and add below piece of code to control the things.
2. {
3. "watchForFileChanges": false,
4. "viewportWidth": 1000,
5. "viewportHeight": 600,
6. "waitForAnimations": true,
7. "animationDistanceTreshold": 20,
8. "defaultCommandTimeout": 5000,
9. "execTimeout": 60000,
10. "pageLoadTimeout": 60000,
11. "requestTimeout": 150000,
12. "responseTimeout": 15000,
13. "video": false,
14. "failOnStatusCode": false,
15. "projectId": "aim3nz"
16. }

2.cypress folder is created.



Under cypress folder we can see.

Fixtures: It is mainly used to store our test data.

Plugins> we can add our plugins

Support- is used to add support methods like login function and just use in the extended place.

Integration-. It is the main or core area where our core test cases are located.

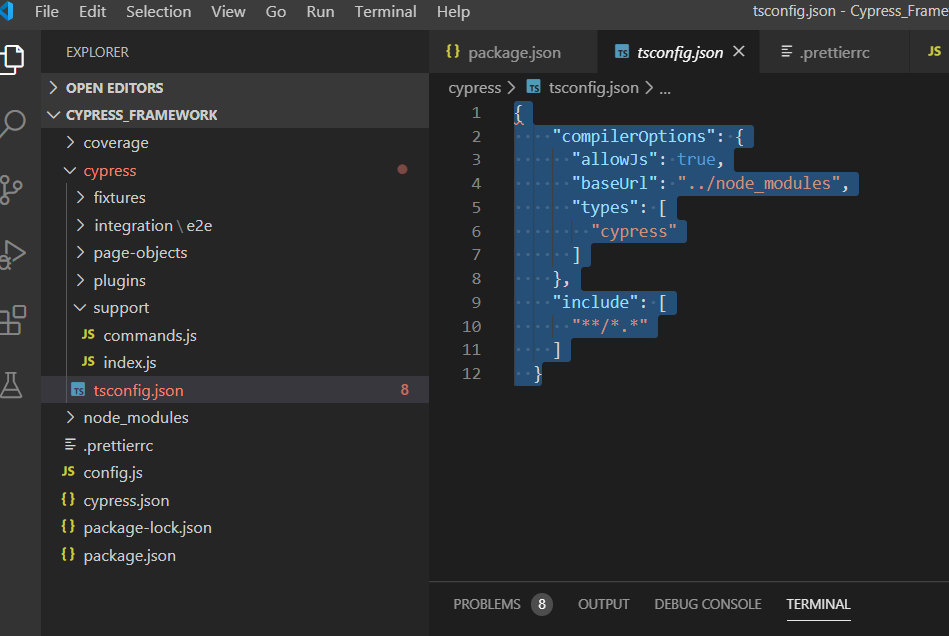
Under this we can/have created different folders.

2. Under cypress model create tsconfig.json file

It is very much important bz all auto intelligent or auto suggestion on pressing of cy. Methods will come from here

//if still it is not coming then in your class file or js file add below piece of code

<reference types=”cypress”>



In tsconfig.json copy and paste below code.

{

    "compilerOptions": {

      "allowJs": true,

      "baseUrl": "../node\_modules",

      "types": [

        "cypress"

      ]

    },

    "include": [

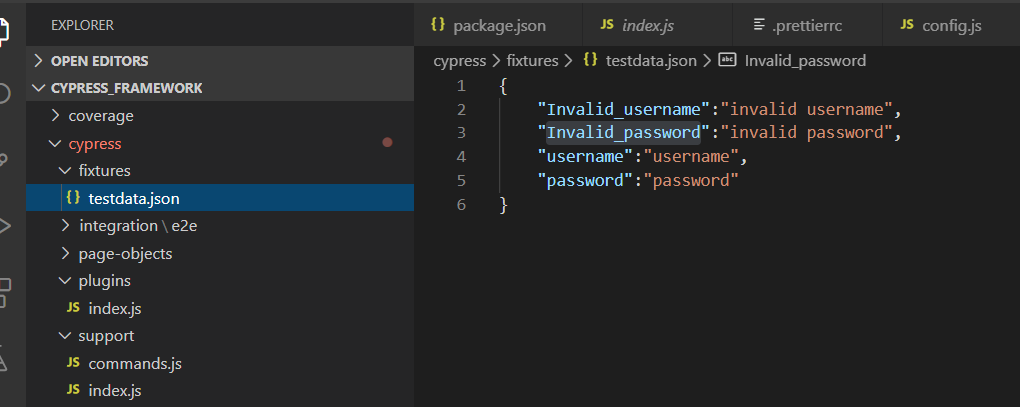
      "\*\*/\*.\*"

    ]

  }

Cypress folder:

Fixtures: Created testdata.json file and added below code as we know that json stores the data in key and value pair.



Support folder

Commands.js it’s default created file and where we can have our custom related commands or code here.

Example:

Cypress.Commands.add('isVisible', selector =>{

    cy.get(selector).should('be.visible')

})

Cypress.Commands.add('isHidden', selector=>{

    cy.get(selector).should('not.exist')

})

const LOGINFORM='#login\_form'

const USERNAME\_INPUT='#user\_login'

const PASSWORD\_INPUT='#user\_password'

const REMEMBER\_CHECKBOX='input[id="user\_remember\_me"]'

Cypress.Commands.add('login',(username,password) =>{

    cy.get(LOGINFORM).should('be.visible')

    cy.get(USERNAME\_INPUT).clear()

    cy.get(USERNAME\_INPUT).type(username)

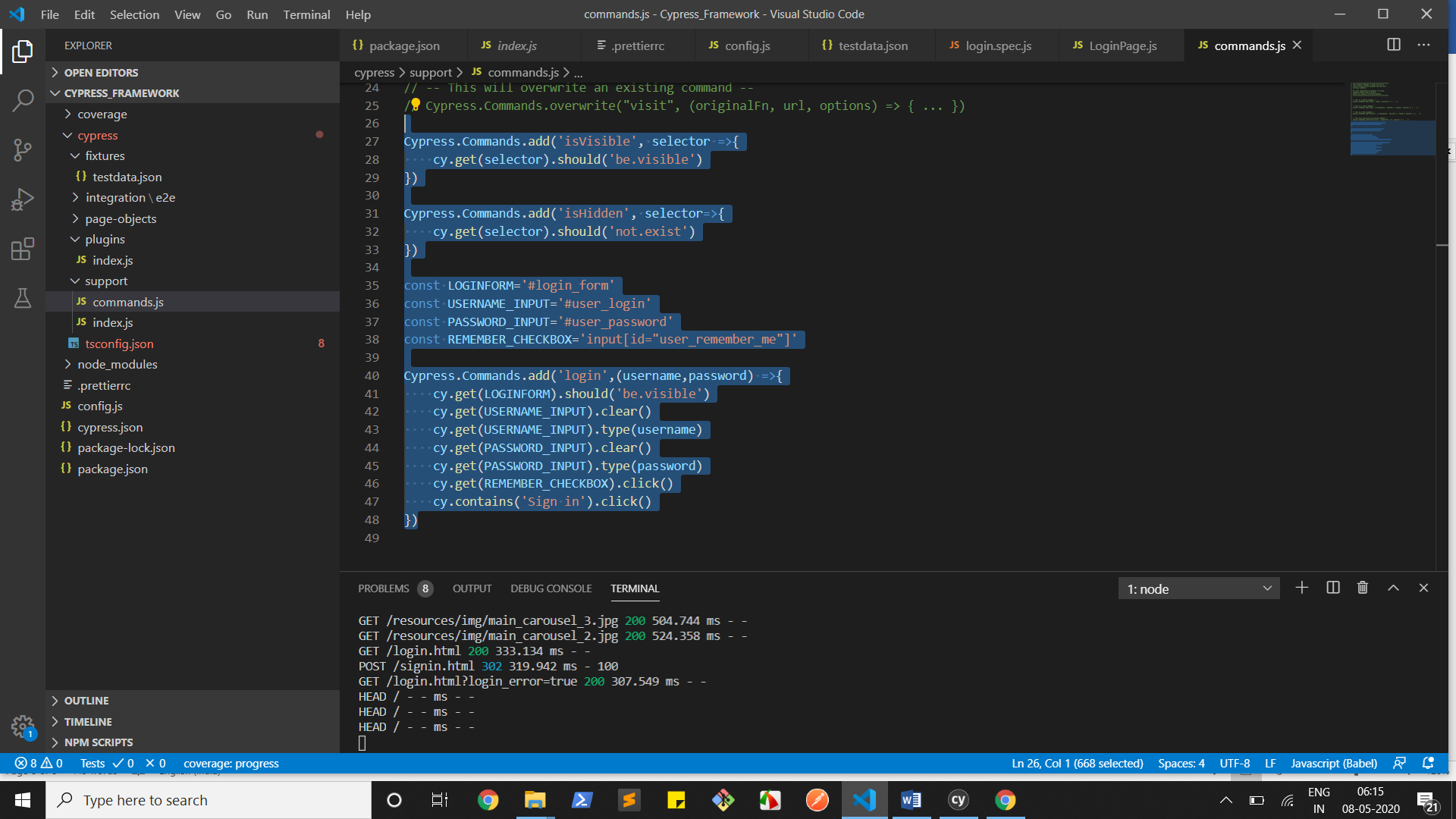
    cy.get(PASSWORD\_INPUT).clear()

    cy.get(PASSWORD\_INPUT).type(password)

    cy.get(REMEMBER\_CHECKBOX).click()

    cy.contains('Sign in').click()

})



In the index.js we can add other info:

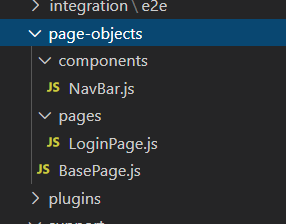
Need to investigate.

Integration Folder:

I have deleted examples folder(which will come as default) and created e2e and

Created another under this Page-objects -> under this we have 2 more folders and a file

1. Components
2. Pages
3. BasePage.js



BasePage.js

It has all static methods which are required in our framework.

export default class BasePage{

    static pause(ms){

        cy.wait(ms)

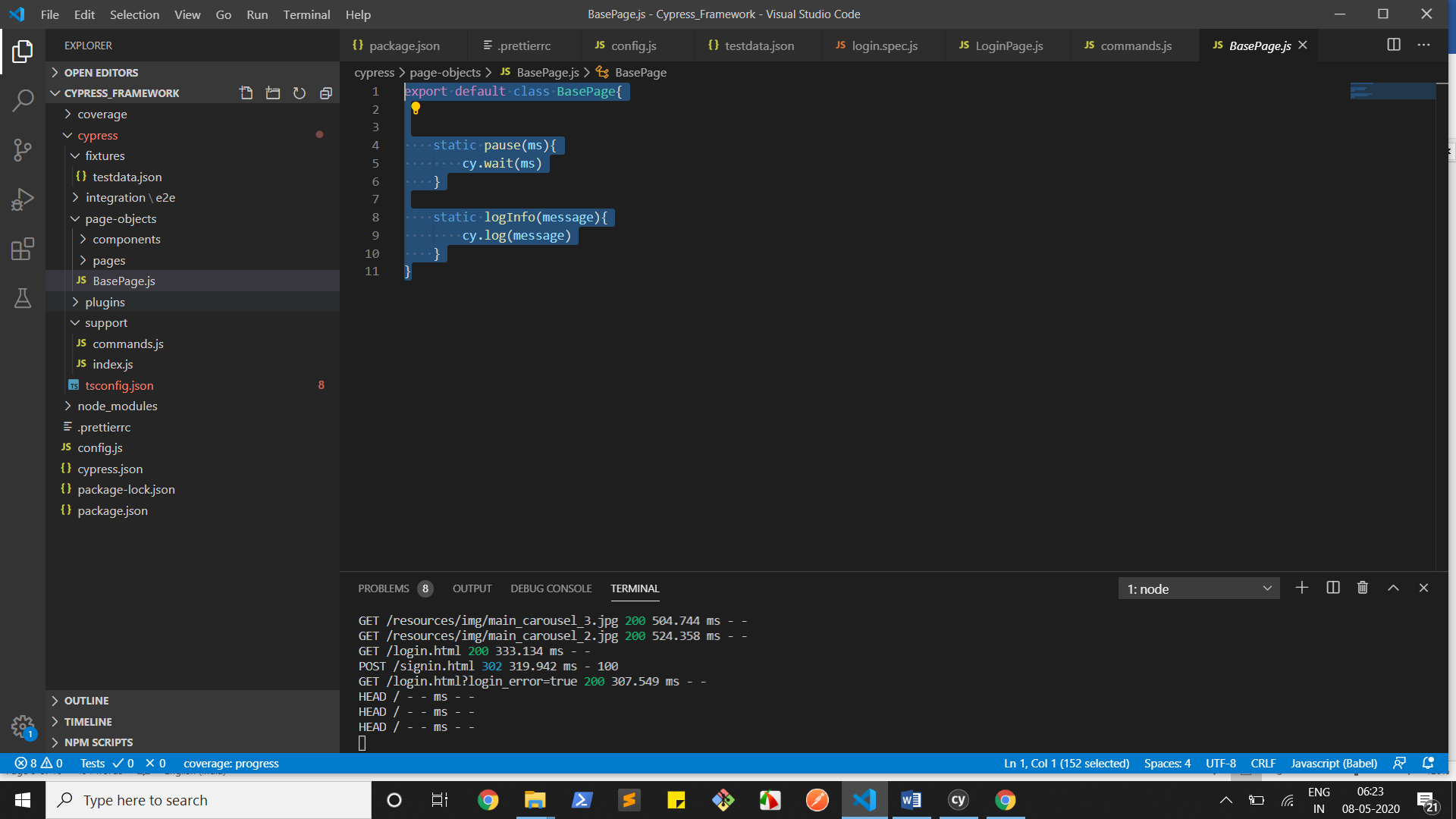
    }

    static logInfo(message){

        cy.log(message)

    }

}



Components-> The whole idea of components is.

In a web page there is header part which is constant for most of the web site.

So related those we have stored in one file.

export default class NavBar{

    static clickOnLogo()

    {

        cy.get('.brand').should('be.visible').click()

    }

    static searchBoxEntry(text){

        cy.get('#searchTerm').type(`${text}{enter}`)

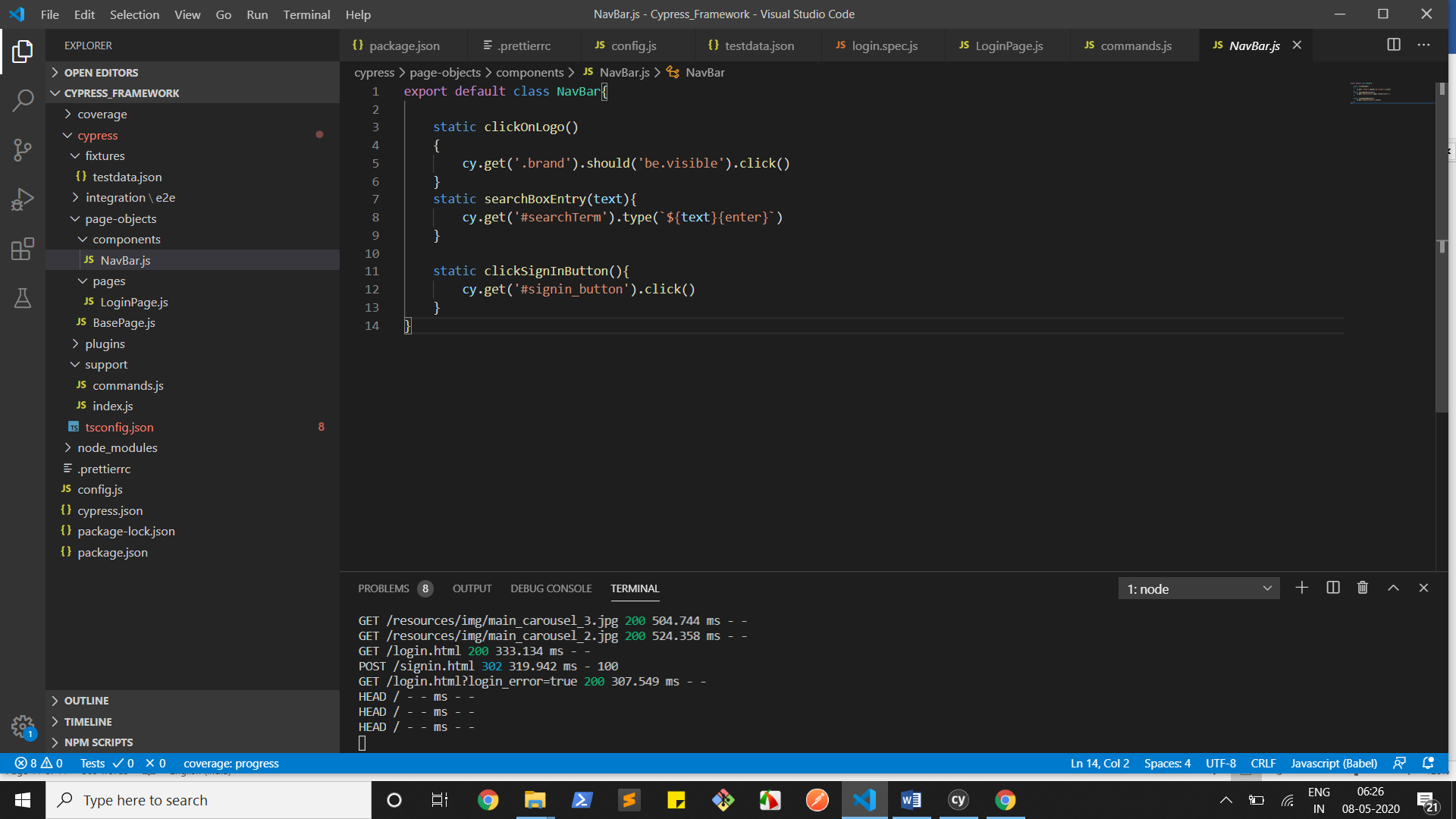
    }

    static clickSignInButton(){

        cy.get('#signin\_button').click()

    }

}



Login Page:

Login page extends commands and fixtures

import BasePage from "../BasePage";

const ERROR\_MSG='.alert-error'

export default class LoginPage extends BasePage{

    static login(username,password){

        cy.login(username,password)

    }

    static clickForgetPasswordLink(){

        cy.contains('Forgot your password ?').should('be.visible').click()

    }

    static verifyErrorMessage(){

        cy.isVisible(ERROR\_MSG).contains('Login and/or password are wrong.')

    }

    static getTheErrorMessage(){

        const na=cy.get(ERROR\_MSG).invoke('text')

    }

}

// ERROR\_MSG is a selector /locator and I have saved in the const var so that I can use the reference,

Everywhere we are creating static methods so, no need to create the objects just use them directly.

For a methods login

I have used cy.login(username,password)

Basically this method we are calling from here

Cypress.Commands.add('login',(username,password) =>{

    cy.get(LOGINFORM).should('be.visible')

    cy.get(USERNAME\_INPUT).clear()

    cy.get(USERNAME\_INPUT).type(username)

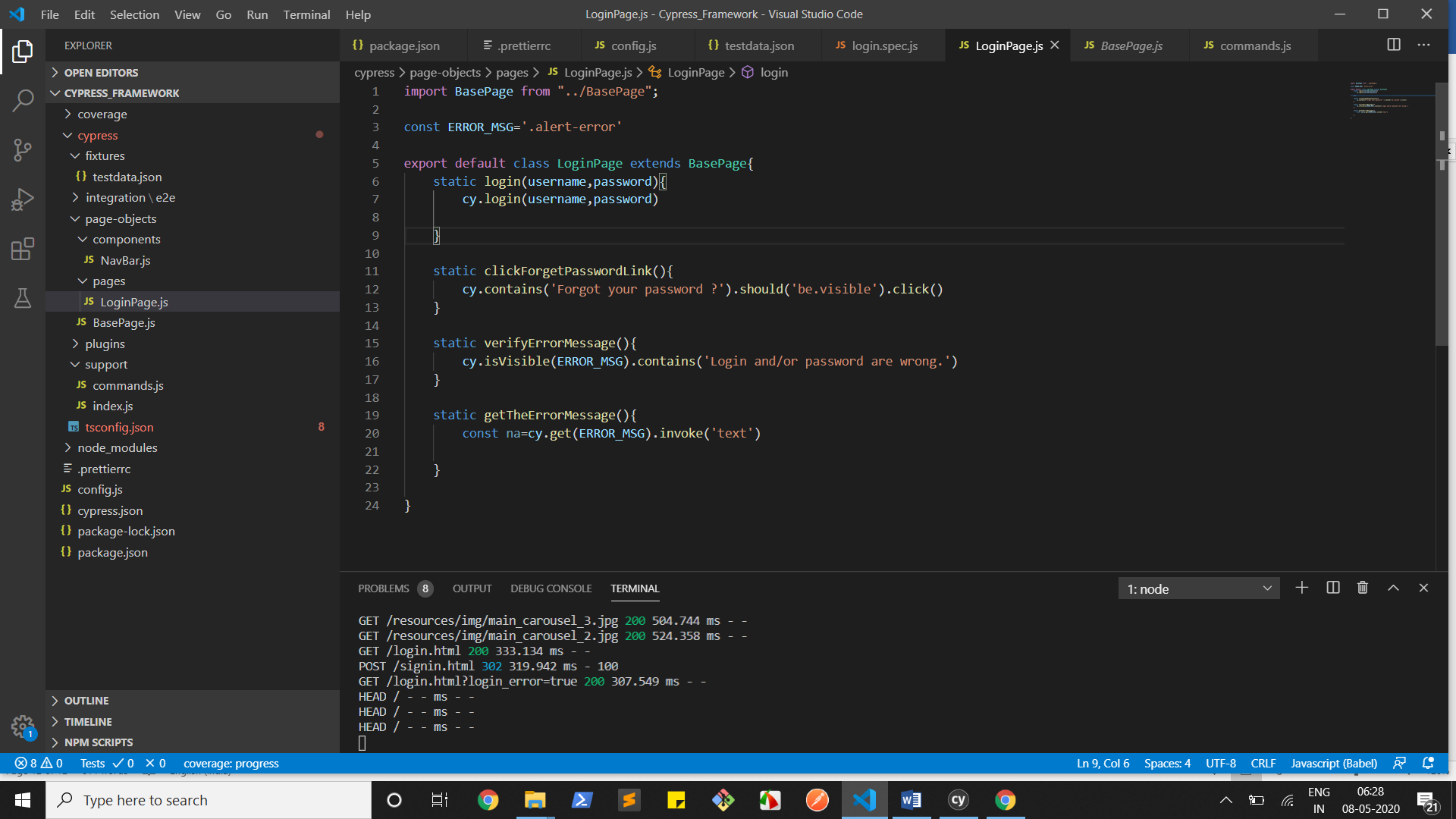
    cy.get(PASSWORD\_INPUT).clear()

    cy.get(PASSWORD\_INPUT).type(password)

    cy.get(REMEMBER\_CHECKBOX).click()

    cy.contains('Sign in').click()

})



Create a file under e2e ->

Login.spec.js(no need to mention login.spec/ you canwrite loginspec.js)

import {url} from '../../../config'

import NavBar from '../../page-objects/components/NavBar'

import LoginPage from '../../page-objects/pages/LoginPage'

describe('this is the first scenario implemented with Framework',()=>{

    before(function(){

        cy.visit(url)

    })

    it('this test case covers the invalid login functionality',()=>{

        NavBar.clickSignInButton()

        cy.fixture("testdata").then(user=>{

            const invalidname=user.Invalid\_username

            const invalidpass=user.Invalid\_password

            LoginPage.logInfo(invalidname)

            LoginPage.logInfo(invalidpass)

            LoginPage.login(invalidname,invalidpass)

        })

    })

    it('should display Error message',()=>{

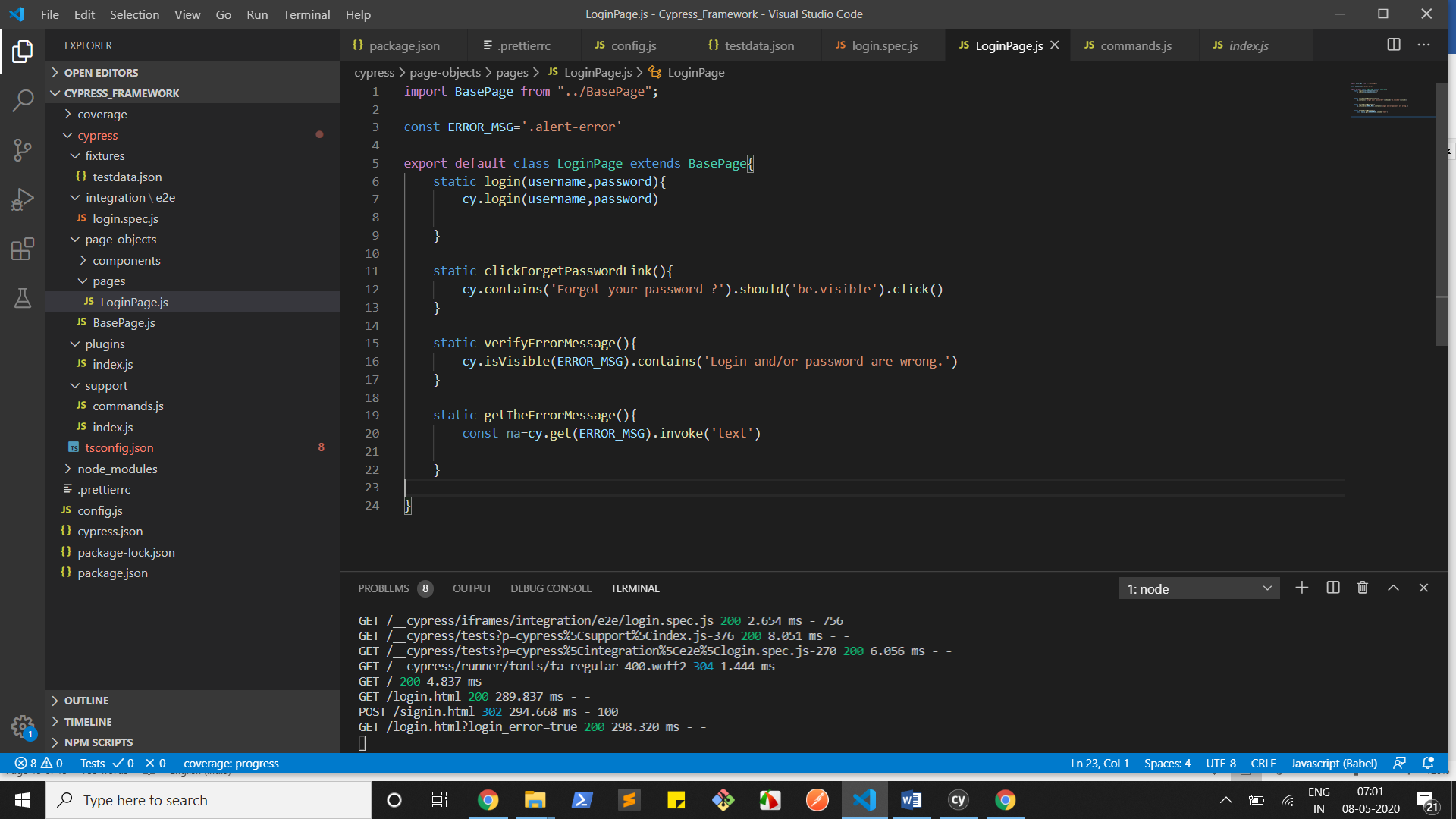
        LoginPage.verifyErrorMessage()

        LoginPage.pause(3000)

    })

})

Used fixtures here to read the test data from the tesdata file.



Here we can see only one login.spec.js file

If we create more files means then it will display all the pages and we can click on run all pages

It will execute the comeplete suite.

