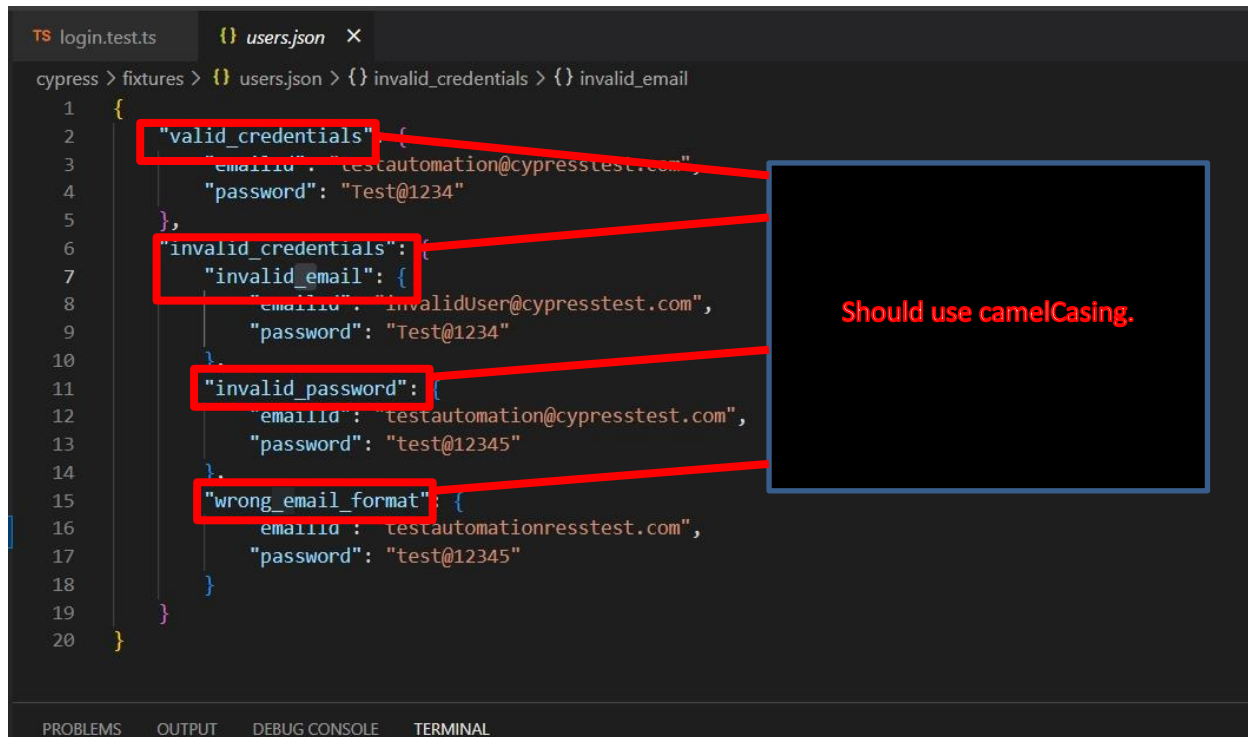


Screenshot1

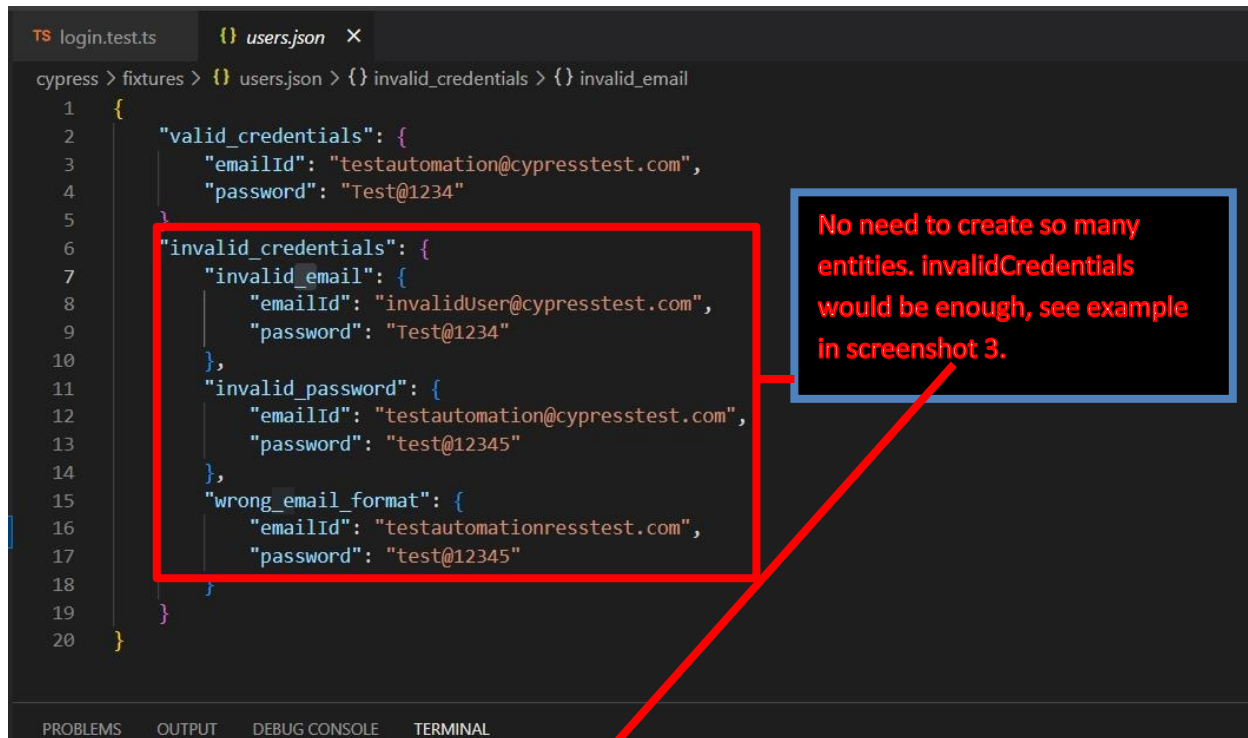


This screenshot shows a Cypress fixtures file named `users.json`. The file contains a root object with four nested objects: `valid_credentials`, `invalid_credentials`, `invalid_email`, and `wrong_email_format`. Each of these nested objects contains an `emailId` and a `password` property. Red boxes highlight each of these four nested objects. A blue box on the right contains the text "Should use camelCasing." with red arrows pointing to the `invalid_credentials`, `invalid_email`, and `wrong_email_format` objects, suggesting a more concise structure.

```
TS login.test.ts  {} users.json X
cypress > fixtures > {} users.json > {} invalid_credentials > {} invalid_email
1  {
2    "valid_credentials": {
3      "emailId": "testautomation@cypress.test.com",
4      "password": "Test@1234"
5    },
6    "invalid_credentials": {
7      "invalid_email": {
8        "emailId": "invalidUser@cypress.test.com",
9        "password": "Test@1234"
10     },
11     "invalid_password": {
12       "emailId": "testautomation@cypress.test.com",
13       "password": "test@12345"
14     },
15     "wrong_email_format": {
16       "emailId": "testautomationresstest.com",
17       "password": "test@12345"
18     }
19   }
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Screenshot2



This screenshot shows the same Cypress fixtures file, but with a more concise structure. The root object now has three main properties: `valid_credentials`, `invalid_credentials`, and `wrong_email_format`. The `invalid_credentials` property is an object containing `invalid_email` and `invalid_password`, which are themselves objects with `emailId` and `password` properties. A red box highlights the `invalid_credentials` object. A blue box on the right contains the text "No need to create so many entities. invalidCredentials would be enough, see example in screenshot 3." with a red arrow pointing to the `invalid_credentials` object.

```
TS login.test.ts  {} users.json X
cypress > fixtures > {} users.json > {} invalid_credentials > {} invalid_email
1  {
2    "valid_credentials": {
3      "emailId": "testautomation@cypress.test.com",
4      "password": "Test@1234"
5    },
6    "invalid_credentials": {
7      "invalid_email": {
8        "emailId": "invalidUser@cypress.test.com",
9        "password": "Test@1234"
10     },
11     "invalid_password": {
12       "emailId": "testautomation@cypress.test.com",
13       "password": "test@12345"
14     },
15     "wrong_email_format": {
16       "emailId": "testautomationresstest.com",
17       "password": "test@12345"
18     }
19   }
20 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Screenshot3



This screenshot shows the most concise version of the Cypress fixtures file. The root object has two main properties: `validCredentials` and `invalidCredentials`. `validCredentials` is an object with `emailId` and `password` properties. `invalidCredentials` is an object containing `emailId`, `password`, and `emailIdFormatWrong` properties. A red arrow from the blue box in Screenshot 2 points to this screenshot.

```
"validCredentials": {
  "emailId": "testautomation@cypress.test.com",
  "password": "Test@1234"
},
"invalidCredentials": {
  "emailId": "invalidUser@cypress.test.com",
  "password": "test@12345",
  "emailIdFormatWrong": "testautomationresstest.com"
}
```

Screenshot4

```
it('login with valid credentials', function () {
  loginPage.login("testautomation@cypresstest.com", "Test@1234")
  myAccountPage.validateSuccessfulLogin()
  myAccountPage.logout()
  myAccountPage.validateSuccessfulLogout()
})
it('login with valid credentials read data from fixture', function () {
  loginPage.login(this.data.valid_credentials.emailId, this.data.valid_credentials.password)
  myAccountPage.validateSuccessfulLogin()
  myAccountPage.logout()
  myAccountPage.validateSuccessfulLogout()
})
```

Pass the same login and password values, "login with valid credentials read data from fixture" is enough.

Screenshot5

```
it('login with invalid email credentials read data from fixture', function () {
  loginPage.login(this.data.invalid_credentials.invalid_email.emailId,
    this.data.invalid_credentials.invalid_email.password)
  loginPage.validateLoginError('Authentication failed.')
})
```

Since the test involves only entering the invalid email, the correct password from "valid\_credentials" block will have to be used.

Screenshot6

```
it('login with invalid password credentials read data from fixture', function () {
  loginPage.login(this.data.invalid_credentials.invalid_password.emailId,
    this.data.invalid_credentials.invalid_password.password)
  loginPage.validateLoginError('Authentication failed.')
})
```

Since the test involves only entering the invalid password, the correct email from "valid\_credentials" block will have to be used.

Screenshot7

```
it('login with wrong email format credentials read data from fixture', function () {
  loginPage.login(this.data.invalid_credentials.wrong_email_format.emailId, this.data.invalid_credentials.wrong_email_format.password)
  loginPage.validateLoginError('Invalid email addressssss.')
})
```

Since the test involves entering only an email in the wrong format, the correct password from "valid\_credentials" block will have to be used.

Screenshot8

```
it('login with wrong email format credentials read data from fixture', function () {  
  loginPage.login(this.data.invalid_credentials.wrong_email_format.emailId, this.data.invalid_credentials.wrong_email_format.passwo  
  loginPage.validateLoginError('Invalid email addressssss.')
```

The value passed to "validateLoginError" is incorrect and causes the test to crash, see screenshot 9.

Screenshot9

```
1) login with wrong email format credentials read data from fixture  
  
4 passing (1m)  
1 failing  
  
1) Login Functionality  
  login with wrong email format credentials read data from fixture:  
  
    Timed out retrying after 5000ms  
    + expected - actual  
  
    -'Invalid email address.'  
    +'Invalid email addressssss.'  
  
    at LoginPage.validateLoginError (http://automationpractice.com/__cypress/tests?p=cypress\e2e\tests\login.test.ts:119:27)  
    at Context.eval (http://automationpractice.com/__cypress/tests?p=cypress\e2e\tests\login.test.ts:199:31)
```

Screenshot10

```
TS login.test.ts TS myAccount.test.ts X  
cypress > e2e > tests > TS myAccount.test.ts > describe('My Account Functionality') callback > it('Sample Test') callback  
1 import { loginPage } from "../pages/loginPage";  
2  
3 describe('My Account Functionality', () => {  
4   beforeEach(() => {  
5     cy.visit('https://google.com');  
6     //loginPage.launchApplication()  
7   })  
8   it('Sample Test', () => {  
9     console.log("This is a sample test")  
10  })  
11 })
```

Test is useless.

Any comments should be removed from the code.

To output to the console, cypress uses the command "cy.log".