

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙞🙞 🕮 🙜🙜**

**ĐỒ ÁN MÔN HỌC**

**KIỂM THỬ PHẦN MỀM**

**ĐỀ TÀI: TÌM HIỂU VÀ PHÂN TÍCH CÔNG CỤ KIỂM THỬ CODECEPTJS**

Ngành: **KỸ THUẬT PHẦN MỀM**

Giảng viên hướng dẫn: **TS. Trần Sơn Hải**

Sinh viên thực hiện:

* Nguyễn Thái Minh Thiện

MSSV: 2000004762 – lớp: 20BITV01

* Nguyễn Hoàng Minh

MSSV: 2000001452 – lớp: 20BITV01

* Lê Nguyễn Đình Hiếu

MSSV: 2000004220 – lớp: 20BITV01

* Lê Phùng Long Nhật

MSSV: 2000002441 – lớp: 20BITV01

* Phạm Trung Tín

MSSV: 2000002491 – lớp 20BITV01

*Thành phố Hồ Chí Minh, ngày 01 tháng 04 năm 2023*

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

Tp. Hồ Chí Minh, ngày…tháng…năm 2023

Giảng viên hướng dẫn

# **LỜI CẢM ƠN**

Để hoàn thành đồ án môn học kiểm thử phần mềm với đề tài “Tìm hiểu và phân tích công cụ kiểm thử CodeceptJS”, trước tiên cho phép em xin gửi lời cảm ơn tới thầy Trần Sơn Hải đã giúp đỡ em rất nhiệt tình trong suốt thời gian qua. Hơn nữa, đồ án của em sẽ không thể hoàn thành tốt nếu không có sự hướng dẫn tận tình của quý thầy/cô giảng viên khoa Công nghệ thông tin - Trường Đại học Nguyễn Tất Thành – Viện đào tạo quốc tế Nguyễn Tất Thành.

Trong vòng đời phát triển phần mềm thường có các giai đoạn là nghiên cứu sơ bộ, phân tích yêu cầu, thiết kế, cài đặt, kiểm thử và bảo trì phần mềm. Cụ thể hơn, thì hai giai đoạn đầu là đặt vấn đề và phân tích các yêu cầu của hệ thống, giai đoạn tiếp theo là quá trình xây dựng trên lý thuyết các module, các thành phần cần có của phần mềm, giai đoạn tiếp nữa là đi hiện thực hóa các lý thuyết đó. Hai giai đoạn cuối luôn yêu cầu sự chuẩn xác và rõ ràng của các giai đoạn trên, để có thể làm việc dễ dàng.

Theo IEEE thì tổng quan về kiểm thử phần mềm là như sau: Kiểm thử là một hoạt động thực hiện để đánh giá chất lượng sản phẩm, để cải thiện nó, bằng cách định ra các nhược điểm và vấn đề của nó. Kiểm thử bao gồm các xác minh về sự đáp ứng của phần mềm trước một tập các test case, được lựa chọn phù hợp từ các module thực thi phổ biến của phần mềm, so sánh với các đáp ứng đáng mong đợi.

Em chọn đề tài:”Tìm hiểu và phân tích công cụ kiểm thử CodeceptJS”để làm đồ án môn học Kỹ thuật phần mềm nhằm mục đích tìm hiểu về các kỹ thuật kiểm thử tại công ty trên thực tế.

Với điều kiện thời gian cũng như kinh nghiệm còn hạn chế của chúng em, luận văn này không thể tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của các thầy cô để tôi có điều kiện bổ sung, nâng cao ý thức của mình, phục vụ tốt hơn công tác thực tế sau này.

Em xin cảm ơn thầy Trần Sơn Hải, quý thầy/cô giảng viên trong Khoa đã giúp đỡ chúng em hoàn thành đồ án và bài báo cáo này.

# **MỤC LỤC**

**BẢNG MỤC LỤC**

[**LỜI CẢM ƠN** 3](#_Toc131434875)

[**MỤC LỤC** 4](#_Toc131434876)

[**CHƯƠNG 1: TỔNG QUAN** 6](#_Toc131434877)

[**1.1.** **Thực trạng hiện nay** 6](#_Toc131434878)

[**1.2.** **Mục đích và nội dung kiểm thử** 6](#_Toc131434879)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 7](#_Toc131434880)

[**2.1.** **Kiểm thử phần mềm** 7](#_Toc131434881)

[***2.1.1.*** ***Tổng quan về kiểm thử phần mềm*** 7](#_Toc131434882)

[***2.1.2.*** ***Vai trò của kiểm thử phần mềm*** 9](#_Toc131434883)

[***2.1.3.*** ***Mục tiêu của kiểm thử phần mềm*** 9](#_Toc131434884)

[***2.1.4.*** ***Kiểm thử tự động*** 10](#_Toc131434885)

[***2.1.5.*** ***Phương pháp thử nghiệm*** 11](#_Toc131434886)

[***2.1.6.*** ***Kỹ thuật kiểm thử hộp trắng*** 13](#_Toc131434887)

[***2.1.7.*** ***Kỹ thuật kiểm thử hộp đen*** 15](#_Toc131434888)

[***2.1.8.*** ***Các mức độ kiểm thử*** 16](#_Toc131434889)

[**2.2.** **CodeceptJS** 18](#_Toc131434890)

[***2.2.1.*** ***CodeceptJS là gì?*** 18](#_Toc131434891)

[***2.2.2.*** ***Vì sao chúng ta nên sử dụng CodeceptJS?*** 18](#_Toc131434892)

[***2.2.3.*** ***Puppeteer là gì?*** 19](#_Toc131434893)

[***2.2.4.*** ***DOM (Document Object Model)*** 20](#_Toc131434894)

[***2.2.5.*** ***XPath Selector*** 20](#_Toc131434895)

[***2.2.6.*** ***End-to-End Testing là gì?*** 21](#_Toc131434896)

[***2.2.7.*** ***Các ưu điểm và nhược điểm của End-to-End Testing*** 21](#_Toc131434897)

[***2.2.8.*** ***So sánh giữa CodeceptJS và Selenium*** 22](#_Toc131434898)

[***2.2.9.*** ***Kiến trúc của CodeceptJS*** 23](#_Toc131434899)

[***2.2.10.*** ***Testomat.io*** 23](#_Toc131434900)

[***2.2.11.*** ***Mục tiêu của việc sử dụng CodeceptJS trong đề án này*** 24](#_Toc131434901)

[**CHƯƠNG 3: THỰC HÀNH VỚI CODECEPTJS** 25](#_Toc131434902)

[**3.1.** **Cài đặt và Setup** 25](#_Toc131434903)

[***3.1.1.*** ***Cài đặt Visual Studio Code*** 25](#_Toc131434904)

[***3.1.2.*** ***Cài đặt thư viện Node.js*** 28](#_Toc131434905)

[**3.2.** **Demo CodeceptJS** 31](#_Toc131434906)

[**3.3.** **Kết luận** 42](#_Toc131434907)

[**REFERENCES** 43](#_Toc131434908)

# **CHƯƠNG 1: TỔNG QUAN**

## **Thực trạng hiện nay**

Hiện nay, ngành kiểm thử phần mềm đóng vai trò rất quan trọng trong quy trình phát triển phần mềm. Các doanh nghiệp đang tích hợp quy trình kiểm thử trong quy trình phát triển phần mềm của mình. Điều này có nghĩa là Tester không chỉ kiểm thử phần mềm mà còn phải đảm bảo rằng sản phẩm phần mềm đáp ứng được các yêu cầu và tiêu chuẩn chất lượng nhất định.

Ngày nay, nhiều công ty tập trung vào phương pháp kiểm thử tự động (Automated Testing) nhằm tăng cường độ chính xác, nâng cao độ tin cậy và giảm thiểu những sai sót do con người gây ra. Đồng thời, các công nghệ mới như Machine Learning (ML) hay AI (Artificial Intelligence) cũng đang được áp dụng để tăng khả năng kiểm thử phần mềm.

Trong các công nghệ mới như Agile Methodology, kiểm thử phần mềm được hội nhập sâu vào quá trình phát triển và cần phải được thực hiện từ đầu đến cuối (end-to-end) quá trình phát triển phần mềm.

Về tổng quan về ngành kiểm thử phần mềm hiện nay, ngành này đang phát triển mạnh mẽ, có vai trò đặc biệt trong quy trình phát triển phần mềm và được coi là quan trọng đối với chất lượng sản phẩm phần mềm.

Để đáp ứng được các yêu cầu mà bên trên vừa đề cập, nhóm chúng em đã nghiên cứu và tìm ra được một phần mềm có thể đáp ứng được, đó là phần mềm CodeceptJS. Để có thể hiểu sâu hơn CodeceptJS là gì, vì sao chúng lại có thể giải quyết được các bài toán kiểm thử chất lượng sản phẩm phần mềm hiện nay một cách hiệu quả thì chúng em sẽ đề cập đến chi tiết hơn **ở phần sau (nhớ bổ sung mục đàng hoàng).**

## **Mục đích và nội dung kiểm thử**

Trong nội dung của bài luận văn này, chúng em tập trung nghiên cứu về những tính năng hữu hiệu của CodeceptJS, những điểm mạnh - yếu và các giải pháp mà CodeceptJS mang lại.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **Kiểm thử phần mềm**

### ***Tổng quan về kiểm thử phần mềm***

Kiểm thử phần mềm là việc kiểm tra kết quả thực hiện của chương trình máy tính xem có đúng với mục tiêu đã đặt ra với nó không thông qua việc thực hiện ở một số mẫu thử.

Kiểm thử phần mềm là việc tìm ra lỗi trong bản thân phần mềm, việc kiểm thử này trong phần mềm sẽ biểu thị ra những thiếu sót mà ta có thể nhận thấy trong hành vi của phần mềm, và tìm ra những phần mềm không tuân theo quy định, đi lệch ra khỏi những yêu cầu của phần mềm.

Kiểm thử không thể xác định hoàn toàn được tất cả các lỗi bên trong phần mềm. Thay vào đó, nó so sánh trạng thái và hành vi của sản phẩm với các oracle - các nguyên tắc hay cơ chế để phát hiện vấn đề. Các oracle này có thể bao gồm (nhưng không giới hạn ở) các đặc tả phần mềm, hợp đồng, sản phẩm tương đương, các phiên bản trước của cùng một sản phẩm, phù hợp với mục đích dự kiến nhằm đáp ứng sự kỳ vọng của người dùng, khách hàng, quy định của pháp luật hiện hành và các tiêu chuẩn liên quan khác.

Mục đích chính của kiểm thử là phát hiện ra các lỗi phần mềm để từ đó khắc phục và sửa chữa. Việc kiểm thử không thể khẳng định được rằng các chức năng của sản phẩm đúng trong mọi điều kiện, mà chỉ có thể khẳng định rằng nó không hoạt động đúng trong những điều kiện cụ thể. Phạm vi của kiểm thử phần mềm thường bao gồm việc kiểm tra mã, thực hiện các mã trong môi trường và điều kiện khác nhau, và việc kiểm thử các khía cạnh của mã: nó có làm đúng nhiệm vụ của nó hay không, và nó có làm những gì cần phải làm hay không. Trong môi trường phát triển phần mềm hiện nay, một đội kiểm thử có thể tách biệt với đội phát triển. Các thành viên trong đội kiểm thử giữ các vai trò khác nhau. Các thông tin thu được từ kiểm thử có thể được sử dụng để điều chỉnh quá trình phát triển phần mềm.

Mỗi sản phẩm phần mềm có một đối tượng phục vụ riêng. Ví dụ như đối tượng của phần mềm trò chơi điện tử là hoàn toàn khác với đối tượng của phần mềm ngân hàng. Vì vậy, khi một tổ chức phát triển hoặc đầu tư vào một sản phẩm phần mềm, họ có thể đánh giá liệu các sản phẩm phần mềm có được chấp nhận bởi người dùng cuối, đối tượng phục vụ, người mua, hay những người giữ vai trò quan trọng khác hay không. Và việc kiểm thử phần mềm là một quá trình nỗ lực để đưa ra những đánh giá này.

Kiểm thử phần mềm bao gồm Xác thực phần mềm (Verification) và Xác minh phần mềm (Validation).

#### **Xác thực phần mềm**

Xác thực là quá trình kiểm tra xem phần mềm có đáp ứng các yêu cầu của người dùng hay không. Nó được thực hiện vào cuối Vòng đời phát triển phần mềm. Nếu phần mềm phù hợp với các yêu cầu mà nó được tạo ra, nó sẽ được xác nhận.

* Việc xác nhận đảm bảo sản phẩm đang được phát triển theo yêu cầu của người dùng.
* Xác thực trả lời cho câu hỏi :"Chúng tôi có đang phát triển sản phẩm đáp ứng tất cả những gì người dùng cần từ phần mềm này không?".
* Xác thực nhấn mạnh vào yêu cầu của người dùng.

#### **Xác minh phần mềm**

Xác minh là quá trình xác nhận xem phần mềm có đáp ứng các yêu cầu kinh doanh hay không và được phát triển tuân theo các đặc điểm kỹ thuật và phương pháp luận phù hợp hay không.

* Việc xác minh đảm bảo sản phẩm đang được phát triển là theo thông số kỹ thuật của thiết kế.
* Xác minh trả lời câu hỏi:"Chúng tôi có đang phát triển sản phẩm này bằng cách tuân thủ chặt chẽ tất cả các thông số kỹ thuật của thiết kế không ?".
* Xác minh tập trung vào thiết kế và thông số kỹ thuật hệ thống.

Mục tiêu của các bài kiểm thử là:

* Error - Đây là những lỗi mã hóa thực tế do các nhà phát triển thực hiện. Ngoài ra, có sự khác biệt trong đầu ra của phần mềm và đầu ra mong muốn, được coi là một lỗi.
* Fault - Khi có error tồn tại thì fault xảy ra. Fault, còn được gọi là bug, là kết quả của một lỗi có thể khiến hệ thống bị lỗi.
* Failure - lỗi được cho là hệ thống không có khả năng thực hiện tác vụ mong muốn. Failure xảy ra khi có lỗi trong hệ thống.

### ***Vai trò của kiểm thử phần mềm***

Kiểm thử để tìm ra lỗi, ghi nhận thông tin về lỗi nhưng không sửa lỗi.

Kiểm thử phần mềm không chỉ cần tìm lỗi phần mềm, mà còn là quá trình kiểm tra và xác minh một phần mềm đã đáp ứng được yêu cầu và mong đợi của khách hàng.

Một số lỗi phần mềm đã gây ra thiệt hại nghiêm trọng trong lịch sử như:

* Máy bay Airbus A300 do lỗi Phần mềm và bị tai nạn ngày 26/4/1994 giết chết 264 người.
* Năm 1985, Máy xạ trị Therac-25 của Canada do lỗi phần mềm mà phát ra tia gây chết người đã giết chết 3 người và làm 3 người khác bị thương nặng.
* Vào tháng Tư năm 1999, một lỗi phần mềm gây ra sự thất bại của một vụ phóng vệ tinh quân sự gây thiệt hại 1,2 tỷ USD, vụ tai nạn đắt đỏ nhất trong lịch sử.
* 5/1996, Một lỗi phần mềm gây ra các tài khoản ngân hàng của 823 khách hàng của một ngân hàng lớn của Hoa Kỳ được ghi với 920 triệu đô la Mỹ.

Chính vì vậy, việc kiểm thử phần mềm là vô cùng quan trọng vì lỗi phần mềm nếu để lọt thì không chỉ thiệt hại về kinh tế mà còn thiệt hại đến tính mạng con người.

### ***Mục tiêu của kiểm thử phần mềm***

Việc thực hiện kiểm thử nhằm mục tiêu chính như sau:

* Bằng việc kiểm thử sẽ tìm ra lỗi trong phần mềm (Myers, 1979) và thiết lập chất lượng của phần mềm (Helzel, 1988).
* Việc kiểm thử thành công khi bạn tìm được ít nhất một lỗi và đưa ra sự đánh giá với độ tin cậy lớn.
* Kiểm thử phần mềm đảm bảo Phần mềm đáp ứng đúng yêu cầu đề ra.
* Kiểm thử phần mềm giúp chúng ta biết rằng phần mềm đang được thử nghiệm thành công.
* Kiểm thử phần mềm giúp xác nhận được rằng Phần mềm đủ điều kiện đến tay Kiểm thử phần mềm sử dụng.
* Kiểm thử phần mềm để đảm bảo chất lượng phần mềm.
* Kiểm thử phần mềm cung cấp và duy trì một sản phẩm chất lượng cho khách hàng.

### ***Kiểm thử tự động***

#### **Kiểm thử tự động là gì?**

Kiểm thử tự động là việc sử dụng các công cụ để thực hiện các test case. Kiểm thử tự động cũng có thể nhập dữ liệu thử nghiệm vào hệ thống kiểm thử, so sánh kết quả mong đợi với kết quả thực tế và tạo ra các báo cáo kiểm thử chi tiết.

#### **Kiểm thử tự động sử dụng khi nào?**Kiểm thử tự động được sử dụng khi:

* Các trường hợp kiểm thử được thực hiện lặp đi lặp lại để đảm bảo tính năng của phần mềm/sản phẩm.
* Thực hiện ở các trường hợp mà kiểm thử thủ công khó thực hiện.
* Các trường hợp kiểm thử cần tốn nhiều thời gian.

#### **Kiểm thử tự động sử dụng ở đâu?**

* Kiểm thử tự động sử dụng trong các giai đoạn kiểm thử: + Unit Testing (Kiểm thử đơn vị) + Integration Testing (Kiểm thử tích hợp).
* Và trong các loại kiểm thử: + Smoke Testing( Kiểm thử khói) + Functional Testing ( Kiểm thử chức năng) + Regression Testing( Kiểm thử hồi quy).
* Và trong kỹ thuật kiểm thử: + Black Box Testing( Kiểm thử hộp đen).

#### **Tại sao phải kiểm thử tự động?**

* Kiểm thử tự động sử dụng các công cụ có thể ghi lại bộ kiểm tra này và phát lại nó theo yêu cầu.
* Tiết kiệm thời gian kiểm thử.
* Tự động hóa không cần can thiệp của con người nên có thể chạy tự động kiểm tra mà không cần giám sát.
* Tự động tăng tốc độ thực hiện kiểm tra.
* Tự động hóa giúp tăng phạm vi kiểm tra.
* Kiểm tra bằng tay có thể trở nên nhàm chán và do đó dễ bị lỗi.
* Cải thiện độ chính xác.
* Nhanh hơn 70% so với kiểm tra thủ công.

### ***Phương pháp thử nghiệm***

Việc kiểm thử có thể được tiến hành dựa trên hai cách tiếp cận, đó là:

* Kiểm thử chức năng (Functional Testing)
* Kiểm thử thực thi (Implementation Testing)

#### **Kiểm thử chức năng (Functional Testing)**

Kiểm thử chức năng (hay **Functional Testing**) là một trong các quy trình đảm bảo chất lượng của lĩnh vực kiểm thử phần mềm. Đây là một loại kiểm thử hộp đen (black box testing), tức là các trường hợp nó cần xét đến sẽ dựa vào đặc tả của ứng dụng/phần mềm hoặc hệ thống đang thử nghiệm. Các chức năng sẽ được kiểm tra bằng cách nhập các giá trị đầu vào và sau đó sẽ kiểm tra, đánh giá các kết quả đầu ra mà không cần quan tâm đến các cấu trúc hay cài đặt bên trong của ứng dụng.

Kiểm thử chức năng là một quy trình so sánh sự khác biệt giữa đặc tả bên ngoài của phần mềm với các chức năng thực tế mà phần mềm cung cấp. Các đặc tả này phần nhiều sẽ dựa vào góc nhìn của người sử dụng về phần mềm, không liên quan đến các công nghệ sử dụng hay các thiết lập bên trong nó.Người sử dụng có thể là bất cứ ai, nếu họ tiếp xúc với phần mềm khi nó là thành phẩm có thể sử dụng được.

#### **Kiểm thử thực thi (Implementation Testing)**

Kiểm thử thực thi (hay Implementation Testing) là hoạt động nơi mà các thiết kế kiểm thử được thực hiện như các trường hợp kiểm thử, các thủ tục kiểm thử và các dữ liệu kiểm thử. Một số tổ chức ví dụ như chuẩn IEE829 định nghĩa các đầu vào và các kết quả được mong đợi liên quan trong tài liệu đặc tả trường hợp kiểm thử và các bước thực hiện kiểm thử trong các tài liệu đặc tả thủ tục kiểm thử. Thông thường, các đầu vào kiểm thử, các kết quả mong đợi và các bước thực hiện đều được ghi lại cùng nhau.

Implementation Testing cũng bao gồm việc tạo dữ liệu kiểm thử được lưu trữ.

Implementation Testing cũng gọi ra việc kiểm tra cuối cùng để đảm bảo cho đội kiểm thử sẵn sàng thực hiện test. Việc kiểm tra cũng bao gồm việc đảm bảo bàn giao môi trường kiểm thử được yêu cầu, dữ liệu kiểm tra và mã code và tất cả các trường hợp kiểm thử đã được viết, được review và sẵn sàng để có thể được chạy.

Implementation Testing cũng có thể liên quan đến phát triển một môi trường kiểm thử miêu tả chi tiết và dữ liệu kiểm thử. Mức độ chi tiết và độ phức tạp của công việc liên quan được làm trong suốt test implementation có thể bị ảnh hưởng bởi chi tiết của sản phẩm công việc kiểm thử ( ví dụ như các trường hợp kiểm thử và các điều kiện kiểm thử). Trong một số trường hợp, đặc biệt nơi mà các kiểm thử là có thể được lưu trữ để sử dụng lại trong kiểm thử hồi quy, các kiểm thử có thể cung cấp các miêu tả chi tiết các bước cần thiết để thực hiện kiểm thử nhằm đảm bảo tính nhất quán , độ tin cậy. Nếu áp dụng các luật quy định, kiểm thử nên cung cấp các bằng chứng của việc tuân thủ các chuẩn có thể được áp dụng.

Trong Implementation Testing, trật tự nơi mà kiểm thử tự động hay bằng tay được chạy nên được bao gồm trong một lịch thực hiện kiểm thử. Các nhà quản lý kiểm thử nên kiểm tra một cách cẩn thận cho các ràng buộc, bao gồm mức độ rủi ro và độ ưu tiên, mà có thể yêu cầu các kiểm thử chạy theo trật tự đặc biệt hoặc trên thiết bị cụ thể. Sự phụ thuộc trên môi trường kiểm thử hoặc dữ liệu kiểm thử phải được biết đến và được kiểm tra.

Có một số không thuận lợi ở Implementation Testing sớm như trong mô hình agile, mã nguồn có thể thay đổi, công việc thực hiện trở nên lỗi thời. Thậm chì với một chu trình phát triển ít thay đổi như chu trình phát triển lặp hay tăng dần có thể dẫn đến những thay đổi đáng kế giữa các vòng lặp, tạo ra các kịch bản kiểm thử không đáng tin cậy hoặc bắt buộc yêu cầu bảo trì cao. Điều này cũng đúng với chu trình phát triển tuần tự được quản lý kém, nơi mà các yêu cầu thay đổi thường xuyên, thậm chí vào lúc cuối của dự án. Effort khi tham gia vào một Implementation Testing lớn thì nên nểu về chu trình phát triển của phần mềm và dự đoán được các đặc trưng của phẩn mềm sẽ có cho việc kiểm thử. Cũng có những thuận lợi trong test implementation sớm ví dụ như các kiểm thử cụ thể cung cấp các ví dụ của phần mềm nên có các hành vi như thế nào đã thực hiện nếu được viết phù hợp với kiểm thử cơ bản. Các chuyên gia ở lĩnh vực kinh doanh thường tìm việc xác minh của các kiểm thử cụ thể thì dễ hơn việc xác minh các quy tắc kinh doanh trừu tượng do đó có thể xác định thêm các yếu kém trong đặc tả phần mềm. Như các kiểm thử được xác minh có thể cung cấp các miêu tả của các hành vi được yêu cầu cho các nhà thiết kế phần mềm và các kỹ thuật viên phát triển phần mềm.

### ***Kỹ thuật kiểm thử hộp trắng***

#### **Kiểm thử hộp trắng là gì?**

Kiểm thử Hộp Trắng (còn gọi là Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing hoặc Structural Testing) là một phương pháp kiểm thử phần mềm trong đó tester biết về cấu trúc nội bộ / thiết kế. Người kiểm tra chọn đầu vào để thực hiện các đường dẫn thông qua mã và xác định đầu ra thích hợp. Kiến thức lập trình và kiến thức thực hiện là rất cần thiết trong kiểm thử hộp trắng.

Kiểm thử hộp trắng bao gồm phân tích dòng dữ liệu, điều khiển dòng, dòng thông tin, mã thực hành, ngoại lệ và những lỗi trình bày trong hệ thống để kiểm tra những hành động của phần mềm không được định hướng trước.

#### **Đối tượng áp dụng**

Đối tượng được kiểm thử là 1 thành phần phần mềm. Thành phần phần mềm có thể là 1 hàm chức năng, 1 module chức năng, 1 phân hệ chức năng…

#### **Mức độ áp dụng**

Phương pháp Kiểm tra Hộp trắng áp dụng cho các mức độ kiểm tra phần mềm sau đây:

* Unit Testing(Kiểm thử đơn vị): Để kiểm tra đường dẫn trong một đơn vị.
* Integration Testing(Test tích hợp): Để kiểm tra đường dẫn giữa các đơn vị.
* System Testing(Test hệ thống): Để kiểm tra các đường dẫn giữa các hệ thống con. Tuy nhiên, nó là chủ yếu áp dụng cho các kiểm thử đơn vị .

#### **Ưu điểm và nhược điểm của hộp trắng**

**Ưu điểm:**

* Test có thể bắt đầu ở giai đoạn sớm hơn, không cần phải chờ đợi cho GUI để có thể test
* Test kỹ càng hơn, có thể bao phủ hầu hết các đường dẫn
* Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh
* Cho phép tìm kiếm các lỗi ẩn bên trong
* Các lập trình viên có thể tự kiểm tra
* Giúp tối ưu việc mã hoá
* Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.

**Nhược điểm:**

* Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình và thực hiện.
* Maintenance test script có thể là một gánh nặng nếu thể hiện thay đổi quá thường xuyên.
* Vì phương pháp thử nghiệm này liên quan chặt chẽ với ứng dụng đang được test, nên các công cụ để phục vụ cho mọi loại triển khai / nền tảng có thể không sẵn có.

### ***Kỹ thuật kiểm thử hộp đen***

#### **Kiểm thử hộp đen là gì?**

Kiểm thử hộp đen: là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.

* Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.
* Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình.
* Cách tiếp cận của các tester đối với hệ thống là không dùng bất kỳ một kiến thức về cấu trúc lập trình bên trong hệ thống, xem hệ thống là một cấu trúc hoàn chỉnh, không thể can thiệp vào bên trong.

#### **Đối tượng áp dụng**

Đối tượng được kiểm thử là 1 thành phần phần mềm. Thành phần phần mềm có thể là 1 chức năng của phần mềm,…

#### **Mức độ áp dụng**

Black Box Testing chủ yếu là được thực hiện trong Function test và System test.

#### **Ưu điểm và nhược điểm của hộp đen**

**Ưu điểm:**

* Các tester được thực hiện từ quan điểm của người dùng và sẽ giúp đỡ trong việc sáng tỏ sự chênh lệch về thông số kỹ thuật.
* Các tester theo phương pháp black box không có “mối ràng buộc” nào với code, và nhận thức của một tester rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bug ở nơi mà các DEV không tìm thấy.
* Tester có thể không phải IT chuyên nghiệp, không cần phải biết ngôn ngữ lập trình hoặc làm thế nào các phần mềm đã được thực hiện.
* Các tester có thể được thực hiện bởi một cơ quan độc lập từ các developer, cho phép một cái nhìn khách quan và tránh sự phát triển thiên vị.
* Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.
* Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

**Nhược điểm:**

* Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn
* Nhiều dự án không có thông số rõ ràng thì việc thiết kế test case rất khó và do đó khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.
* Khả năng để bản thân kỹ sư lạc lối trong khi kiểm thử là khá cao.
* Chỉ có một số nhỏ các đầu vào có thể được kiểm tra và nhiều đường dẫn chương trình sẽ được để lại chưa được kiểm tra.
* Kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin” bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào. Có nhiều trường hợp khi một tester viết rất nhiều trường hợp test để kiểm tra một số thứ có thể chỉ được test bằng một trường hợp test và/hoặc một vài phần cuối cùng không được test hết.

### ***Các mức độ kiểm thử***

#### **Kiểm thử đơn vị (Unit Testing)**

Loại kiểm thử này được chạy bởi developer trước khi cài đặt để chuyển giao cho đội kiểm thử thực hiện những test case chính thống. Unit test được developer chạy tương ứng cho từng đơn vị mã nguồn. Developer sử dụng bộ dữ liệu kiểm thử từ test case để đảm bảo chất lượng. Mục đích của unit test là cô lập từng phần của chương trình và kiểm tra các bộ phận độc lập đấy đã hoạt động chính xác theo yêu cầu hay chưa. Kiểm thử theo phương pháp này không thể tìm được mọi lỗi của ứng dụng. Không thể đánh giá mọi tình huống có thể xảy ra trong chương trình. Nó bị giới hạn bởi các tình huống và bộ dữ liệu test mà developer sử dụng để kiểm thử chương trình.

#### **Kiểm thử tích hợp (Integration Testing)**

Tích hợp có nghĩa là kết hợp. Ví dụ, trong giai đoạn kiểm thử này, các module phần mềm khác nhau được kết hợp và kiểm tra thành một nhóm để đảm bảo rằng hệ thống tích hợp đã sẵn sàng để kiểm thử hệ thống. Kiểm thử tích hợp kiểm tra luồng dữ liệu từ một module này đến một module khác. Loại kiểm thử này được thực hiện bởi những tester. Có hai kiểu : Tích hợp từ dưới đi lên (Bottom-up integration) và tích hợp đi trên đi xuống (Top-down integration).

**Tích hợp từ dưới đi lên:**

* Việc kiểm thử được bắt đầu bằng unit test, sau đó sẽ đi đến mức cao hơn, đó là kết hợp các thành phần và chức năng của hệ thống.

**Tích hợp từ trên đi xuống:**

* Những module cấp cao nhất được thử nghiệm đầu tiên rồi đến các module thấp hơn được thử nghiệm.

Trong một môi trường phát triển phần mềm hiện đại,  kiểm thử từ dưới lên thường được thực hiện đầu tiên và sau đó là kiểm thử từ trên xuống. Quá trình này được kết thúc sau nhiều vòng kiểm thử trên ứng dụng hoàn chỉnh, tốt nhất là sử dụng các kịch bản kiểm thử dựa trên các tình huống thực tế.

#### **Kiểm thử hệ thống (System Testing)**

Kiểm thử hệ thống được thực hiện trên một hệ thống hoàn chỉnh, tích hợp. Nó cho phép kiểm tra sự tuân thủ của hệ thống theo yêu cầu. Loại kiểm thử này kiểm tra sự tương tác tổng thể của các thành phần. Nó liên quan đến tải, hiệu suất, độ tin cậy và kiểm tra bảo mật. Kiểm thử hệ thống thường xuyên nhất là kiểm tra cuối cùng để xác minh rằng hệ thống đáp ứng các đặc điểm kỹ thuật. Nó đánh giá cả nhu cầu chức năng và phi chức năng để thử nghiệm.

Đây là bước đầu tiên trong vòng đời phát triển phần mềm nơi mà ứng dụng sẽ được kiểm tra toàn bộ. Ứng dụng sẽ được kiểm thử kĩ để xác minh xem nó có đáp ứng các mô tả về chức năng và kĩ thuật không. Ứng dụng được kiểm thử trong môi trường gần giống với môi trường thực tế nơi mà sản phẩm sẽ được cài đặt.

#### **Kiểm thử chấp nhận (Acceptance Testing)**

Kiểm thử chấp nhận là kiểm thử được tiến hành để tìm xem các yêu cầu của đặc điểm kỹ thuật hoặc hợp đồng có được đáp ứng với những yêu cầu của khách hàng hay không. Kiểm thử chấp nhận về cơ bản được thực hiện bởi người dùng hoặc khách hàng.

Đây là loại kiểm thử quan trọng vì nó được thực hiện bởi đội QA đánh giá xem liệu ứng dụng có đáp ứng các thông số kĩ thuật và yêu cầu của khách hàng. Đội QA sẽ tập hợp các kịch bản trước và các test case đó sẽ được sử dụng để kiểm thử ứng dụng.

## **CodeceptJS**

### ***CodeceptJS là gì?***

CodeceptJS là một thư viện kiểm thử tự động (end-to-end testing) cho việc kiểm thử ứng dụng web. End to End Testing khác với Unit Testing (chỉ kiểm thử từng bộ phận), Manual Testing (kiểm thử thủ công, giống ở chỗ là kiểm thử sản phẩm cuối cùng, khác ở chỗ End to End Testing kiểm thử tự động). Nó được thiết kế để giúp cho việc viết và chạy các ca kiểm thử đơn giản và dễ dàng hơn, với hỗ trợ cho nhiều loại trình duyệt và các công nghệ khác nhau như WebDriver, Puppeteer, Playwright...(Helpers). CodeceptJS cũng cung cấp tích hợp với các công cụ như Mocha, Chai, và Sinon để cung cấp cho người dùng một trải nghiệm kiểm thử toàn diện và linh hoạt hơn. CodeceptJS cho phép viết các ca kiểm thử bằng các kịch bản (scenario) có cú pháp dễ đọc và dễ hiểu, với hỗ trợ cho nhiều ngôn ngữ lập trình khác nhau như JavaScript, TypeScript và PHP. Ngoài ra, CodeceptJS còn cung cấp những tính năng tiện ích như chụp ảnh và video khi kiểm thử, và hiển thị kết quả kiểm thử bằng các báo cáo rõ ràng và dễ hiểu.

### ***Vì sao chúng ta nên sử dụng CodeceptJS?***

Chúng ta nên sử dụng CodeceptJS để viết, chạy và quản lý các ca kiểm thử tự động cho ứng dụng web vì nó có các ưu điểm sau:

* Cú pháp đơn giản và dễ hiểu: CodeceptJS cho phép viết các ca kiểm thử bằng các kịch bản đơn giản và dễ hiểu, giúp cho người dùng có thể viết ca kiểm thử nhanh chóng và hiểu được nội dung cvủa chúng
* Hỗ trợ nhiều loại trình duyệt và công nghệ khác nhau: CodeceptJS hỗ trợ nhiều loại trình duyệt và các công nghệ khác nhau như WebDriver, Puppeteer, Playwright..., giúp cho người dùng có thể kiểm thử trên nhiều nền tảng khác nhau.
* Tích hợp với các công cụ như Mocha, Chai, Sinon: CodeceptJS tích hợp với các công cụ này để cung cấp cho người dùng trải nghiệm kiểm thử toàn diện và linh hoạt hơn.
* Chụp ảnh và video khi kiểm thử: CodeceptJS cung cấp các tính năng tiện ích như chụp ảnh và video khi thực hiện kiểm thử, giúp cho người dùng có thể đánh giá kết quả kiểm thử một cách dễ dàng và rõ ràng.
* Hiển thị kết quả kiểm thử bằng các báo cáo rõ ràng và dễ hiểu: CodeceptJS cung cấp các báo cáo rõ ràng và dễ hiểu, giúp cho người dùng có thể theo dõi kết quả kiểm thử một cách dễ dàng và hiệu quả.

Hiện tại chưa có các nhược điểm nổi bật của CodeceptJS được nêu ra. Tuy nhiên, việc sử dụng CodeceptJS có thể đòi hỏi người dùng nắm vững kiến thức về lập trình JavaScript hoặc TypeScript để có thể sử dụng được các tính năng của thư viện một cách tối ưu. Ngoài ra, việc tìm hiểu và triển khai CodeceptJS cũng có thể yêu cầu thời gian và nguồn lực đầu tư từ người dùng.

### ***Puppeteer là gì?***

Puppeteer là một thư viện Node.js cung cấp một API cấp cao để điều khiển Chrome hoặc Chromium bằng cách sử dụng giao thức DevTools. Nó cho phép kiểm tra các chức năng của trình duyệt một cách tự động và là một trong những công cụ phổ biến nhất để kiểm tra giao diện người dùng (UI) trong quá trình automation testing. Puppeteer cũng hỗ trợ chạy headless, tức là chạy trình duyệt mà không có giao diện người dùng. Sử dụng Puppeteer, bạn có thể tắt và mở trang, điều hướng đến các trang khác nhau, điền vào các biểu mẫu và thu thập dữ liệu.

* Tự động hóa các ứng dụng **SPA** (**Single Page Application**): Nếu SPA trong câu hỏi là viết tắt của "Single Page Application", thì SPA trong Testing là thực hiện kiểm thử phần mềm cho các ứng dụng web dạng Single Page Application (SPA). Kiểm thử SPA có một số thách thức đặc biệt, bởi vì thay vì tải lại toàn bộ trang web khi người dùng thực hiện một hành động, SPA thường tải lại chỉ một phần nhỏ của trang. Điều này đôi khi làm cho việc tạo ra các kịch bản kiểm thử chính xác trở nên khó khăn hơn. Tuy nhiên, các công cụ kiểm thử tự động như Selenium và Puppeteer cũng hỗ trợ kiểm thử các ứng dụng SPA bằng cách sử dụng các API cung cấp bởi trình duyệt, giúp đơn giản hóa quá trình kiểm thử.
* Tạo các môi trường kiểm thử tự động với JavaScripts.
* Là một tiện ích mở rộng kiểm tra của Chrome.
* Tự động hóa các biểu mẫu, UI testings, các input của keyboard,…

### ***DOM (Document Object Model)***

**DOM** là chữ viết tắt của **Document Object Model**, là một trong những thuật ngữ phổ biến trong lập trình web. DOM là một chuẩn được định nghĩa bởi W3C và dùng để biểu diễn trang web thành các thành phần dễ dàng truy cập và thao tác thông qua các ngôn ngữ lập trình như JavaScript. Cụ thể, khi một trang web được tải, trình duyệt sẽ tạo ra một cây DOM được tạo thành từ các đối tượng, mỗi đối tượng đại diện cho một phần của trang. Điều này cho phép các lập trình viên tương tác với các phần tử HTML trên trang web, thêm, xóa hoặc chỉnh sửa nội dung bằng các lệnh JavaScript hoặc các thư viện JavaScript như jQuery. DOM là một trong những khái niệm quan trọng trong lập trình web và được ứng dụng rộng rãi trong các ứng dụng có giao diện người dùng động.

### ***XPath Selector***

**XPath selectors** là một loại locator được sử dụng trong automation testing để xác định vị trí của các phần tử trên trang web bằng cách sử dụng cú pháp XPath. XPath là ngôn ngữ truy vấn sử dụng để truy xuất các phần tử trong tài liệu XML hoặc HTML. XPath selector sử dụng các thuộc tính của phần tử như class, ID, name và các thuộc tính khác để xác định vị trí của một phần tử cụ thể. XPath selector cũng có thể sử dụng các toán tử so sánh, hàm và biểu thức để xác định các phần tử trên trang web dễ dàng hơn. XPath selector được sử dụng phổ biến trong Selenium và các công cụ kiểm thử tự động khác để xác định vị trí của các phần tử trên trang web và thực hiện các tương tác với chúng.

* **XPath** viết tắt cho **XML Path Language**
* Xpath sử dụng cú pháp “path like” để xác định và điều hướng các node trong XML/HTML documents.
* Xpath chứa hơn **200** built-in functions.
* Xpath là một thành phần chủ chốt trong tiêu chuẩn của **XSLT**.



Cú pháp đơn giản của XPath: **//tagname[@attribute = “value”]**

### ***End-to-End Testing là gì?***

End-to-end testing là một phương pháp kiểm thử toàn diện, được sử dụng để kiểm tra tính đúng đắn và tương tác giữa các thành phần trong một ứng dụng. Trong end-to-end testing, một ca kiểm thử được thiết kế để kiểm tra hành vi của toàn bộ ứng dụng từ đầu đến cuối, thông qua giao diện người dùng hoặc API. Mục đích của end-to-end testing là để đảm bảo rằng ứng dụng hoạt động như mong đợi từ góc độ của người dùng, đồng thời phát hiện và sửa các lỗi liên quan đến tính tương tác và tính hội tụ của các thành phần trong ứng dụng.

### ***Các ưu điểm và nhược điểm của End-to-End Testing***

#### **Ưu điểm**

* Kiểm thử tự động hệ thống phát triển qua thời gian thì chi phí kiểm thử sẽ giảm xuống.
* Rút ngắn thời gian kiểm thử rất nhiều so với kiểm thử thủ công.
* Gia tăng được số test case theo thời gian. Có tính tích luỹ và lặp lại.
* Có thể lập trình những kỹ thuật phức tạp ví dụ nhập dữ liệu vào form, rồi truy vấn vào cơ sở dữ liệu để kiểm tra. Hoặc kích hoạt một sự kiện, rồi kiểm tra email gửi tới sau đó vài phút...

#### **Nhược điểm**

* Phải lập trình: lập trình viên phải tự kiểm thử thủ công từng bước một rồi sau đó ghi lại, chuyển hoá sang code. Lập trình viên sẽ phải hiểu cơ chế DOM, CSS,… vững để bóc tách từng thành phần.
* Khi giao diện thay đổi liên tục thì kịch bản có thể không đúng nữa. Do đó phải lập trình web, đặt tên các DOM element, các CSS class có quy tắc chặc chẽ, ổn định.

### ***So sánh giữa CodeceptJS và Selenium***

Selenium là thư viện kiểm thử web site dựa trên cơ chế tự động hoá tương tác lên trình duyệt. Trong một thời gian dài, Selenium là thư viện defacto (mặc định sử dụng) để viết kiểm thử tự động End to End Testing. Selenium có thể lập trình kiểm thử bằng Java, Javascript hoặc Python.

CodeceptJS là thư viện kiểm thử End to End testing, chỉ lập trình bằng JavaScript. Rõ ràng với lập trình viên web thì JavaScript sẽ là ngôn ngữ thuận tiện nhất, dễ hiểu, dễ dùng nhất và không phải học lại.  CodeceptJS còn làm được nhiều thứ hay ho hơn mà Selenium làm chưa tốt hoặc không làm được:

1. Cho phép tuỳ chọn thư viện hỗ trợ (helpers) phù hợp với tình huống kiểm thử nhất: hỗ trợ nhiều loại trình duyệt cross browser sử dụng Selenium - Test Cafe, nếu cần tốc độ dùng Chrome Puppeteer.
2. CodeceptJS hỗ trợ rất nhiều helpers:

* **WebDriver:** thư viện của Selenium
* **Protractor:** kiểm thử web site viết bằng Angular
* **Puppeteer:** kiểm thử sử dụng trình duyệt Chrome, chiếm 90% thị phần tốc độ thực thi nhanh
* **Playwright:** thư viện NodeJS tự động hoá trình duyệt Chrome, Edge, Firefox, Web Kit do Microsoft viết. Đang phát triển rất mạnh.
* **Nightmare:** thư viện tự động hoá trình duyệt, hiện không còn cập nhật nhiều nữa. Hãy chuyển sang Pupetter hoặc Playwright sẽ tốt hơn.
* **TestCafe:** thư viện tương tự như CodeceptJS. Nếu bạn đã quen CodeceptJS bạn có thể chuyển sang TestCafe và ngược lại. TestCafe tích hợp với nhiều thư viện CI/CD.
* CodeceptJS còn hỗ trợ test **REST API**, mobile app dùng **Appmium**.

Rõ ràng CodeceptJS và TestCafe tốt hơn, hiện đại hơn, tích hợp với CI/CD tiện hơn Selenium. Selenium ra đời trước, hỗ trợ nhiều loại ngôn ngữ lập trình. Tuy nhiên với sự phổ biến của JavaScript, và buộc phải dùng khi lập trình Web, hỗ trợ tốt JavaScript để bổ xung nhanh nhiều tính năng mạnh lại là lựa chọn khôn ngoan.

### ***Kiến trúc của CodeceptJS***

Diagram

Description automatically generated

Hình : Mô hình kiến trúc của CodeceptJS

### ***Testomat.io***

Testomat.io là một công cụ quản lý ca kiểm thử cho CodeceptJS, được các tác giả của CodeceptJS phát triển. Công cụ này cung cấp tính năng tích hợp kiểm thử tự động (automated testing integration), dự án báo cáo và phân tích (real-time reporting and analytics), hỗ trợ BDD và Gherkin, tài liệu sống (living documentation), tích hợp với Jira và CI/CD. Testomat.io giúp quản lý test case dễ dàng, cho phép đánh dấu các ca kiểm thử đã hoàn thành, tạo và quản lý phiên bản, và chia sẻ kết quả kiểm thử với đồng đội.

### ***Mục tiêu của việc sử dụng CodeceptJS trong đề án này***

Mục tiêu của việc sử dụng CodeceptJS với Puppeteer helper trong đề án này là chúng em sẽ đưa ra 3 test cases đơn giản về việc login thất bại, login thành công và logout (bằng cách truy vấn các elements thông qua Xpath). Sau đó chúng em sẽ tích hợp với công cụ Testomat.io để quản lý các ca kiểm thử cho CodeceptJS.

# **CHƯƠNG 3: THỰC HÀNH VỚI CODECEPTJS**

## **Cài đặt và Setup**

Yêu cầu cơ bản khi sử dụng CodeceptJS thì chúng ta phải có NodeJS vì CodeceptJS chạy trên nền của NodeJS. Hiện nay chúng ta có rất nhiều IDE phổ biến để lập trình JavaScript như Visual Studio Code, Atom, IntelliJ IDEA, WebStorm, Brackets,…Nhưng ở đây, chúng em sẽ sử dụng Visual Studio Code vì:

* **Đa nền tảng:** Visual Studio Code hỗ trợ đa nền tảng bao gồm Windows, macOS và Linux, cho phép người dùng dễ dàng sử dụng trên nhiều hệ điều hành.
* **Tùy biến:** Visual Studio Code cung cấp cho người dùng các công cụ để tùy chỉnh giao diện và tính năng của nó để phù hợp với nhu cầu sử dụng của từng người dùng.
* **Tiện ích:** Visual Studio Code tích hợp tính năng như IntelliSense, Debugging, Git, Terminal, Emmet, Snippets..., tăng cường sức mạnh và hiệu quả cho việc lập trình.
* **Dễ sử dụng:** Visual Studio Code đơn giản và dễ sử dụng, với cú pháp, mã màu và các phím tắt hữu ích giúp cho việc lập trình và chỉnh sửa mã nguồn trở nên thuận tiện và nhanh chóng hơn.
* **Cộng đồng phát triển lớn:** Visual Studio Code có một cộng đồng phát triển lớn và tích cực, cung cấp các tiện ích và trợ giúp cho người dùng trên các diễn đàn và trang web.
* **Miễn phí và mã nguồn mở:** Visual Studio Code là một phần mềm miễn phí và mã nguồn mở, cho phép người sử dụng dễ dàng truy cập và thay đổi mã nguồn theo nhu cầu của mình.

Một số ưu điểm khác của Visual Studio Code bao gồm hiệu năng cao, tính mở rộng và tính linh hoạt.

### ***Cài đặt Visual Studio Code***

Bước 1, chúng ta sẽ tiến hành tải xuống trình cài đặt tại địa chỉ <https://code.visualstudio.com/download>. Chọn hệ điều hành, phiên bản phù hợp với máy tính cá nhân của chúng ta. Ở đây, chúng em sẽ cài đặt theo hệ điều hành Windows 11, phiên bản 64 bit (VSCodeUserSetup-x64-1.77.0.exe). Ở đây, chúng ta sẽ nhấn nút tải **WINDOWS**.

Graphical user interface, website

Description automatically generated

Hình : Giao diện màn hình cài đặt Visual Studio Code

Sau khi chúng ta nhấn vào nút tải về, chúng ta sẽ thấy File Explorer hiện ra với tên file như hình bên dưới. Sau đó, chúng ta nhấn **SAVE**. Chúng ta muốn lưu file .exe này ở đâu thì chúng ta sẽ chọn địa chỉ lưu tùy ý mỗi người.

Graphical user interface

Description automatically generated with medium confidence

Hình : File Explorer sau khi chúng ta tải về

Bước 2, Sau khi chúng ta tải tệp đã cài đặt về, chúng ta chạy tệp đó, chọn ngôn ngữ và đồng ý với các điều khoản của Microsoft. Chúng ta nhấn nút **NEXT**:

Graphical user interface, text, application, email

Description automatically generated

Hình : Các điều khoản của Microsoft

Bước 3, Sau khi theo các hướng dẫn cài đặt, chúng ta sẽ thấy màn hình như sau:

Graphical user interface, text, application

Description automatically generated with medium confidence

Hình : Màn hình Setup sau khi cài đặt thành công

Ở đây, chúng ta sẽ nhấn nút **FINISH** để xác nhận hoàn tất việc cài đặt. Sau đó, chúng ta sẽ thấy giao diện của Visual Studio Code sẽ hiện lên như thế này:

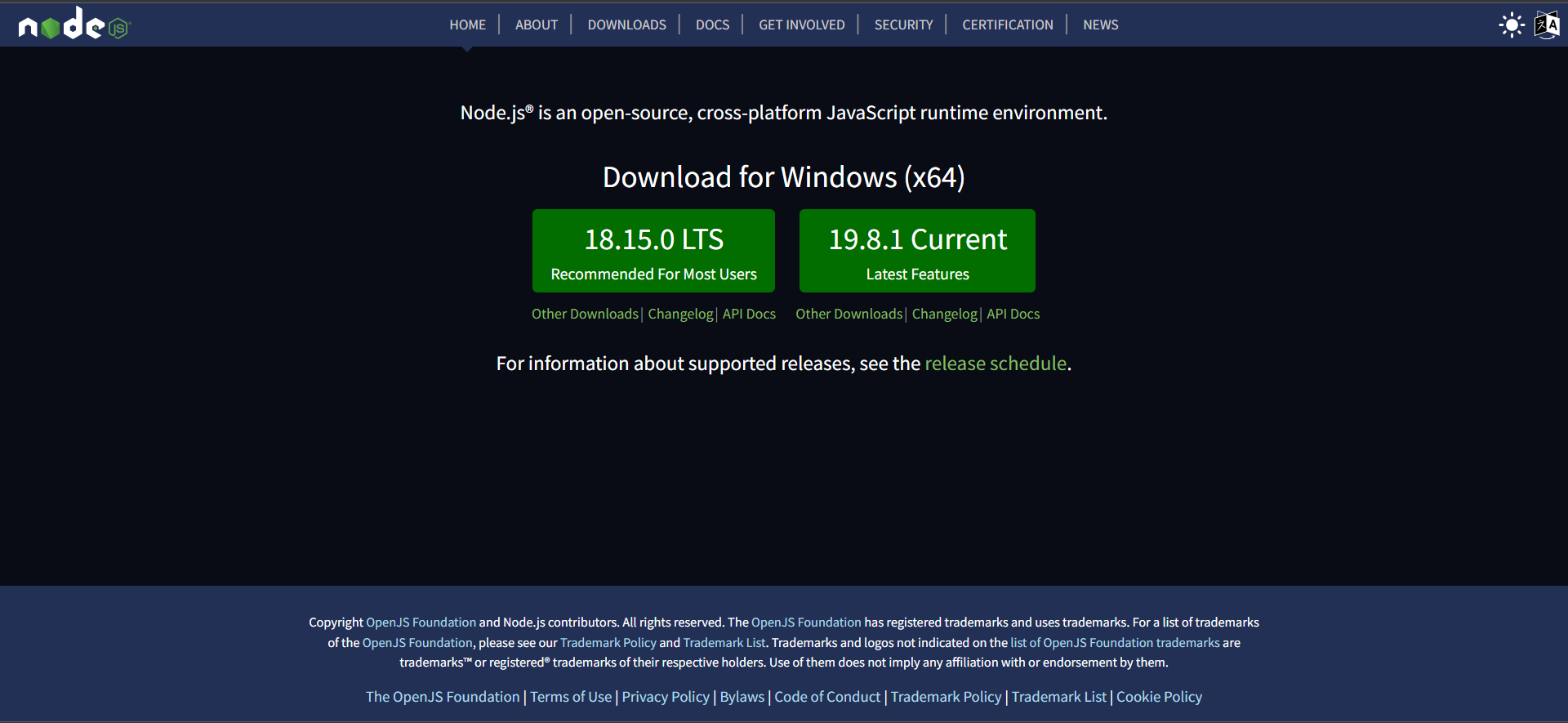
A screenshot of a computer

Description automatically generated with medium confidence

Hình : Giao diện của Visual Studio Code

### ***Cài đặt thư viện Node.js***

Bước 1, chúng ta sẽ tải về trình cài đặt của Node.js tại địa chỉ <https://nodejs.org/en>. Sau đó chúng ta chọn phiên bản phù hợp cho máy tính cá nhân của chúng ta.



Hình : Giao diện cài đặt của Node.js

Sau đó chúng ta sẽ tải trình cài đặt về địa chỉ tùy ý mình, nhấn **SAVE**:

Graphical user interface

Description automatically generated

Hình : File Explorer sau khi nhấn nút cài đặt

Bước 2, việc tiếp theo của chúng ta rất đơn giản, đó chỉ là chấp nhận các tùy chọn mặc định và nhấn “Next…Next” cho tới bước cuối cùng.

Graphical user interface, text, application

Description automatically generated

Hình : Trình cài đặt của Node.js

Chấp nhận các điều khoản của Node.js và tiếp tục nhấn **NEXT**:

Graphical user interface, text, application, email

Description automatically generated

Hình : Các điều khoản của Node.js

Tiếp tục nhấn **FINISH** để xác nhận việc hoàn tất cài đặt Node.js

Graphical user interface, text, application, chat or text message

Description automatically generated

Hình : Màn hình trình cài đặt của Node.js đã hoàn tất

Bước 3, kiểm tra lại kết quả cài đặt và cấu hình của Node.js. Mở cửa sổ **CMD** lên và thực thi các lệnh sau để kiểm tra phiên bản của **NodeJS** và **NPM**:

Graphical user interface, text

Description automatically generated

Hình : Màn hình CMD kiểm tra phiên bản của NodeJS và NPM

Vậy là bước cài đặt & Setup của chúng ta đã hoàn tất.

## **Demo CodeceptJS**

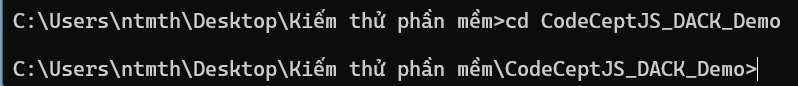
Đầu tiên, chúng ta sẽ tạo thư mục mới với địa chỉ tùy ý với tên mình tự đặt tùy ý (ở đây, chúng em sẽ tạo thư mục mới với tên ***CodeCeptJS\_DACK\_Demo*** tại đường dẫn ***C:\Users\ntmth\Desktop\Kiểm thử phần mềm>***)***.*** Tại đây, chúng em sẽ tạo mới trên CMD với cú pháp ***mkdir CodeCeptJS\_DACK\_Demo***:

Text

Description automatically generated

Hình : Tạo mởi thư mục với tên CodeCeptJS\_DACK\_Demo

Sau đó, chúng em sẽ chuyển vào đường dẫn của thư mục vừa mới tạo với cú pháp ***cd CodeCeptJS\_DACK\_Demo:***



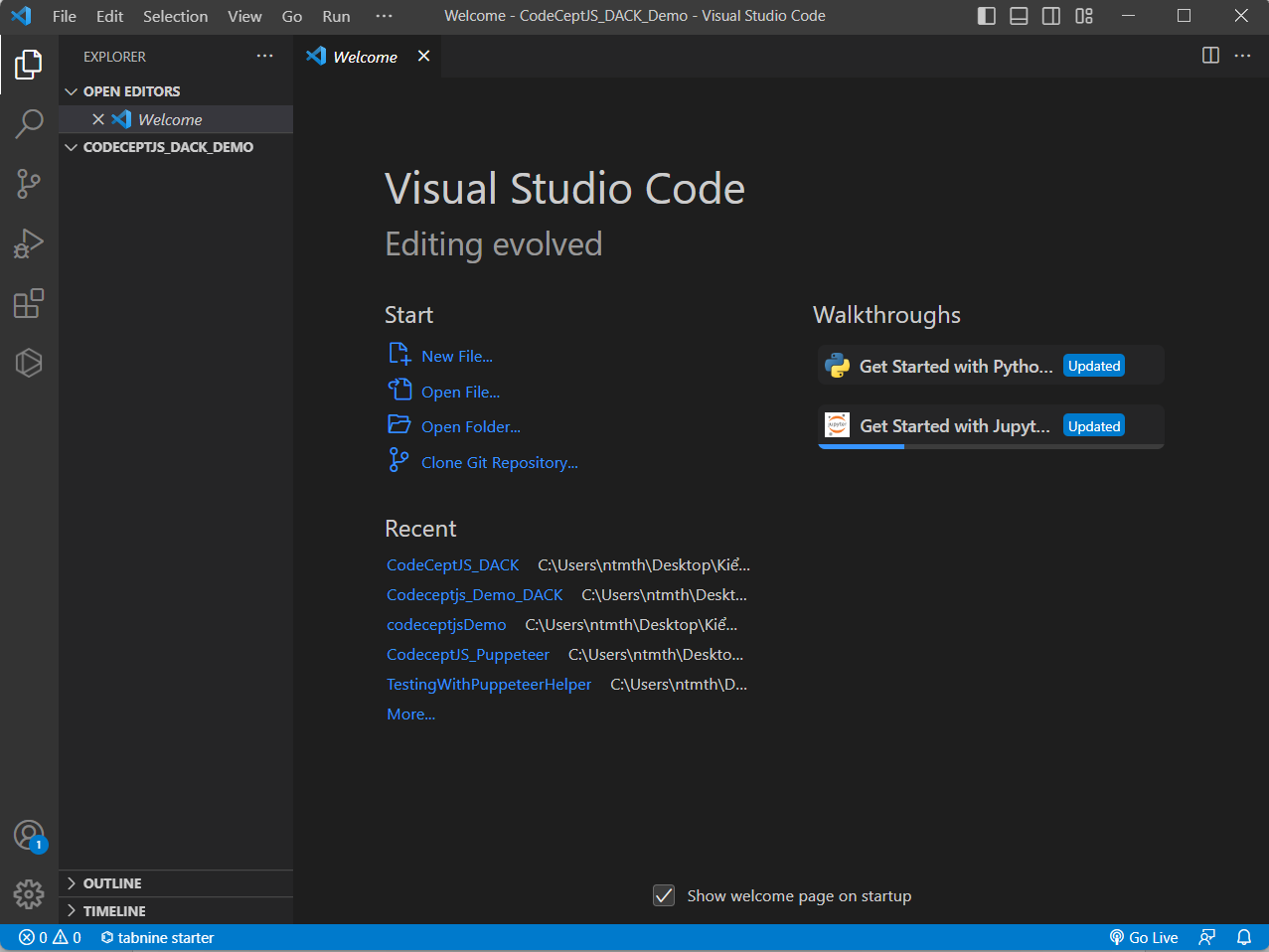
Hình : Chuyển vào đường dẫn của thư mục vừa mới tạo

Tiếp theo, chúng em sẽ mở Visual Studio Code lên với cú pháp ***code .***



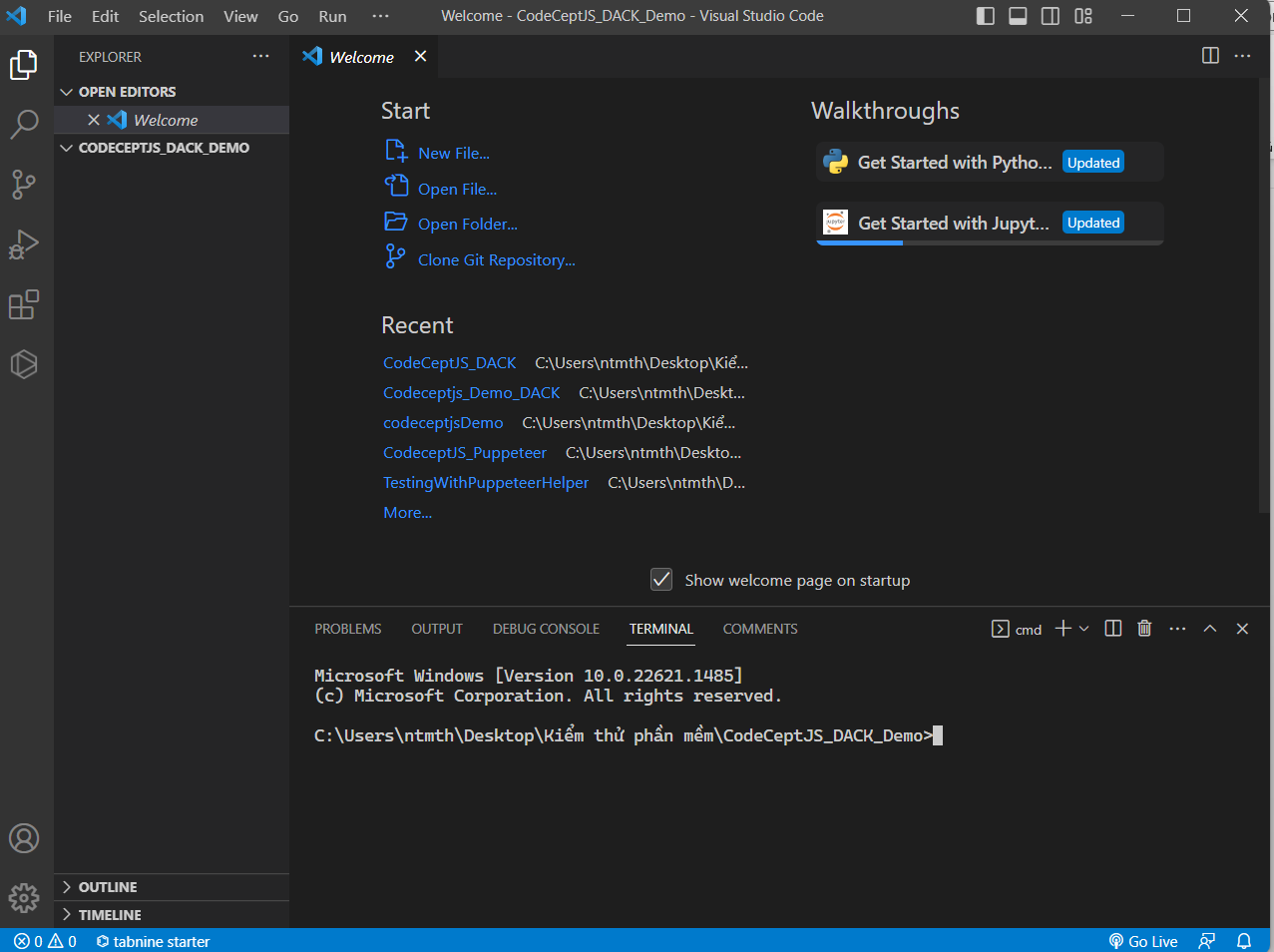
Hình : Mở Visual Studio Code với cú pháp như hình đề cập

Và màn hình Visual Studio Code sẽ hiện lên:



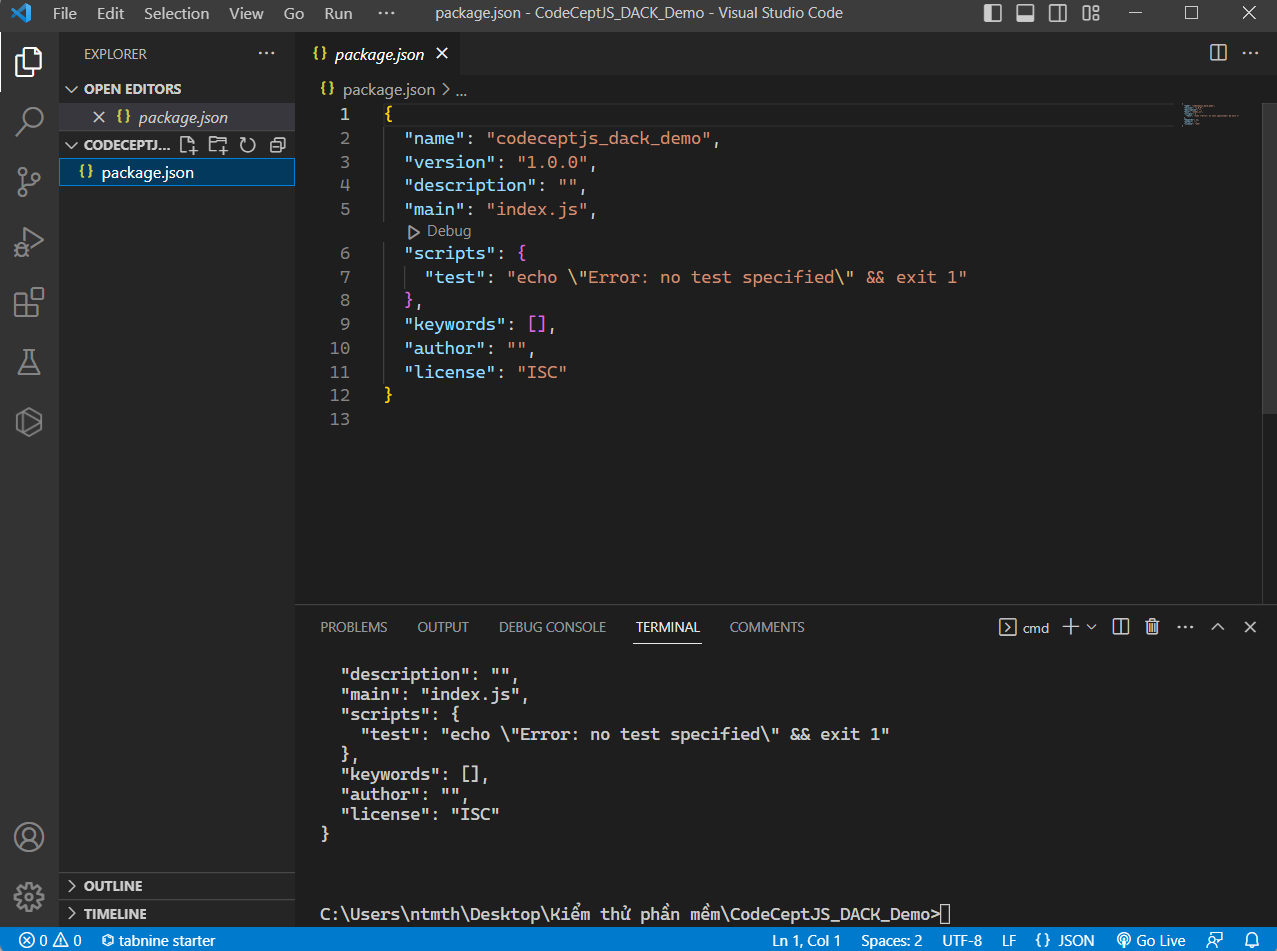
Hình : Giao diện của Visual Studio Code

Tiếp theo, chúng em sẽ mở Terminal trên Visual Studio Code (một trong những tiện ích được tích hợp mặc định trong Visual Studio Code) bằng shortcut ***Ctrl + Shift + ~`***



Hình : Mở Terminal trên Visual Studio Code

Bắt đầu chúng em sẽ tiến hành demo CodeceptJS, trước tiên, chúng em sẽ gõ câu lệnh ***npm init -y*** để khởi tạo **NPM**:



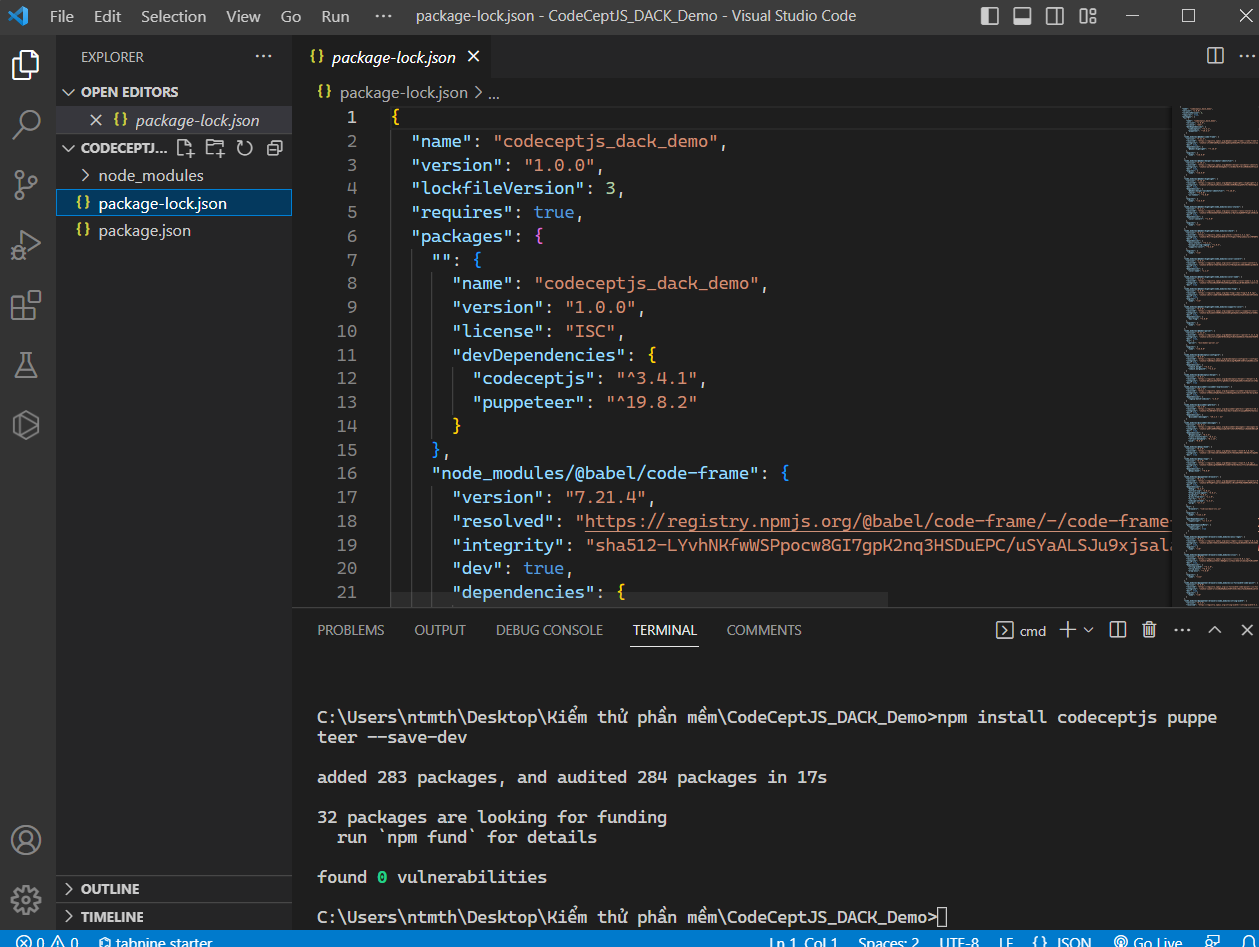
Hình : Màn hình kết quả sau khi khởi tạo NPM

Lúc này, chúng ta sẽ thấy xuất hiện file ***package.json*** với nội dung như trên hình miêu tả.

Về cơ bản, tệp ***package.json*** là phần cốt lõi của hệ sinh thái Node.js và là phần cơ bản nhất để hiểu và làm việc với Node.js, npm và cả JavaScript. Nó được sử dụng như một bản kê khai thông tin ứng dụng, các modules, thư viện và nhiều thứ khác được sử dụng trong ứng dụng của chúng ta. Để làm việc hiệu quả với Node.js, JavaScript, một lập trình viên của hệ sinh thái này cần phải biết đến package.json. Vì sự hiểu biết cơ bản về package.json rất cần thiết để phát triển với Node.js, chúng ta hãy cùng xem qua và phác thảo một số thuộc tính phổ biến và quan trọng nhất của package.json mà bạn sẽ cần sử dụng.

Để biết thêm thông tin chi tiết các trường dữ liệu, chúng ta có thể tham khảo tại <https://codestus.com/posts/kien-thuc-co-ban-ve-file-packagejson>

Tiếp theo, chúng em sẽ tải package của Puppeteer với câu lệnh ***npm install codeceptjs puppeteer –save-dev*** :



Hình : Màn hình kết quả sau khi tải gói Puppeteer về máy

Ở đây, chúng ta sẽ thấy xuất hiện một file mới có tên ***package-lock.json***. Chúng ta có thể tham khảo để hiểu thêm tại <https://duthanhduoc.com/blog/tai-sao-package-lock-json-ton-tai-va-cach-no-hoat-dong>

Tiếp theo, chúng em sẽ khởi tạo codeceptjs bằng cú pháp ***npx codeceptjs init:***

Text

Description automatically generated

Hình : Khởi tạo CodeceptJS

Ở đây, sau khi chúng ta gõ lệnh khởi tạo xong, chúng ta sẽ thấy một loạt các câu hỏi và chúng em sẽ setup như sau:

Text

Description automatically generated

Hình : Kết quả sau khi nhập đầy đủ các yêu cầu

Sau đó, chúng em sẽ tạo một file test, đầu tiên chúng em sẽ test đăng nhập thất bại trước:

Graphical user interface, text

Description automatically generated

Hình : Tạo một file Test mới (01\_InvalidLoginTest\_test.js) với chức năng 01\_InvalidLoginTest

Và giao diện của chúng em sẽ như thế này:

A screenshot of a computer

Description automatically generated

Hình : Màn hình kết quả sau khi tạo file Test mới

Sau đó, chúng em sẽ tiến hành code (ở đây chúng em sẽ demo code chúng em đã code sẵn nên sẽ thấy tên file và tên thư mục, đường dẫn bị thay đổi):

Text

Description automatically generated

Hình : File test đăng nhập thất bại

Text

Description automatically generated

Hình : File test đăng nhập thành công

Text

Description automatically generated

Hình : File test đăng xuất thành công

Ở đây, chúng em sẽ giải thích một số phương thức sau:

* Chúng em sẽ có một hàm tên là ***Scenario*** để thực thi test case của chúng em
* ***I*** là một đối tượng, là một actor, một abstraction cho tester, ***I*** còn là một đối tượng ủy quyền cho **Helpers** đang được kích hoạt (**Puppeteer**).
* Hàm ***.amOnPage(“[url]”)*** dùng để truy cập đến đường dẫn url mà mình muốn test.
* Hàm ***.wait()*** dùng để chờ đợi trong một khoảng thời gian (giây).
* Hàm ***.fillField(“Xpath Selector”, “replaced text”)*** dùng để nhập thông tin, văn bản vào textbox.
* Hàm ***.click(“Xpath selector”)*** dùng để nhấn vào đối tượng nút.
* Hàm ***.saveScreenshot(“Image name.extensions”)*** dùng để chụp screenshot lưu lại kết quả test.
* Hàm ***.tag(“@string”)*** dùng để tag chuỗi cho các file test, dùng để chạy các file test trên các browser khác nhau (Chrome, Firefox,…) cùng một lúc (**Multiple Browsers Execution**). (chúng ta có thể tham khảo thêm tại <http://xt1.org/codecept/multi/>)
* Hàm ***.see(“string”)*** dùng để check chuỗi ký tự, nếu có sẽ trả về chuỗi ký tự trên màn hình Terminal.

Ngoài ra, chúng ta sẽ có vô số các phương thức hữu ích khác và có thể tham khảo thêm tại <https://codecept.io/helpers/Puppeteer/>

Sau khi demo test xong xuôi, chúng ta sẽ tạo reporter để lưu trữ và báo cáo kết quả test của chúng ta.

Graphical user interface, application

Description automatically generated

Hình : Reporter trên công cụ Testomat.io

Text

Description automatically generated

Hình : Kết quả test được lưu lại

Tại đây, chúng em chỉ nói tổng quan về quá trình demo, để cho thầy/cô và mọi người có thể hiểu chi tiết hơn về quá trình demo thì chúng em sẽ demo trực tiếp tại buổi báo cáo cuối kỳ môn kiểm thử phần mềm.

## **Kết luận**

Hiện tại, đề án của chúng em chỉ giới thiệu tổng quan về framework CodeceptJS và Puppeteer và các chức năng, hàm cơ bản và thông dụng nhất. CodeceptJS còn rất nhiều tính năng rất tuyệt vời, vì thế hướng đề án trong tương lai của chúng em sẽ khai thác sâu hơn về CodeceptJS và demo ra các sản phẩm mang tính thực tiễn hơn, phức tạp hơn.

# **REFERENCES**

<https://codecept.io/>

<https://testomat.io/>

<https://techmaster.vn/posts/36137/codeceptjs-phan-1-codeceptjs-la-gi-cai-dat-nhu-the-nao>

<https://www.youtube.com/watch?v=rwfGf4cGUNc>

<http://xt1.org/codecept/multi/>

<https://codestus.com/posts/kien-thuc-co-ban-ve-file-packagejson>

<https://duthanhduoc.com/blog/tai-sao-package-lock-json-ton-tai-va-cach-no-hoat-dong>

<https://www.youtube.com/watch?v=BRMWstiOTks&t=708s>