TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THỦ ĐỨC

KHOA CÔNG NGHỆ THÔNG TIN



PRJ – REPORT  
DIAGRAMS & EXTENSION

Giảng Viên: Phan Thanh Nhuần

**Nhóm D**:

1.Trần Phong Nhã - 18211TT1000

2.Nguyễn Phương Tân - 18211TT2680

3.Trần Triều Thuận - 18211TT0530

Ngày 17/11/2020



**MỤC LỤC**

[**I. QUY TRÌNH LÀM VIỆC NHÓM** 11](#_Toc60531201)

[**1. Bảng phân công công việc** 11](#_Toc60531202)

[**2. Sprint Backlog** 13](#_Toc60531203)

[**3. Kết thúc một sprint** 13](#_Toc60531204)

[**4. Tiến độ công việc** 13](#_Toc60531205)

[**II. DIAGRAMS** 17](#_Toc60531206)

[**1. Activity Diagram - Bản vẽ hoạt động** 17](#_Toc60531207)

[**1.1 Mở đầu** 17](#_Toc60531208)

[**1.2 Các thành phần của Dynamic Model** 17](#_Toc60531209)

[**1.3 Activity Diagram** 18](#_Toc60531210)

[**1.4 Cách xây dựng Activity Diagram** 21](#_Toc60531211)

[**1.5 Khi nào cần sử dụng bản vẽ Activity Diagram** 21](#_Toc60531212)

[**1.6 Lợi ích của bản vẽ Activity Diagram** 21](#_Toc60531213)

[**1.7 Áp dụng vào ví dụ** 21](#_Toc60531214)

[**2. Deployment Diagram – Bản Vẽ Triển Khai** 24](#_Toc60531215)

[**2.1 Những ký hiệu trong bản vẽ triển khai** 24](#_Toc60531216)

[**2.2 Cách vẽ một sơ đồ triển khai** 25](#_Toc60531217)

[**2.3 Mục đích của sơ đồ triển khai** 27](#_Toc60531218)

[**2.4 Áp dụng Deployment Diagram vào source code php-training** 27](#_Toc60531219)

[**3. Sequence Diagram - Bản vẽ tuần tự** 30](#_Toc60531220)

[**3.1 Khái niệm** 30](#_Toc60531221)

[**3.2 Các thành phần của Sequence Diagram** 31](#_Toc60531222)

[**3.3 Các bước xây dựng Sequence Diagram** 39](#_Toc60531223)

[**3.4 Ứng dụng Sequence Diagram** 40](#_Toc60531224)

[**4. Use Case Diagram - Bản vẽ Use Case** 41](#_Toc60531225)

[**4.1 Khái niệm** 41](#_Toc60531226)

[**4.2 Các thành phần trong bản vẽ Use Case** 41](#_Toc60531227)

[**4.2.1 Actor** 42](#_Toc60531228)

[**4.2.2 Use Case** 42](#_Toc60531229)

[**4.2.3 Relationship (Quan hệ)** 42](#_Toc60531230)

[**4.2.4 System Boundary** 45](#_Toc60531231)

[**4.3 Đặc tả Use Case** 46](#_Toc60531232)

[**4.4 Mục đích sử dụng Use Case Diagram** 46](#_Toc60531233)

[**4.5 Các bước xây dựng Use Case Diagram** 47](#_Toc60531234)

[**4.6 Áp dụng Use Case Diagram vào extension AutoCheck và CheckAuto** 47](#_Toc60531235)

[**5. Class Diagram - Bản vẽ về Class (lớp)** 50](#_Toc60531236)

[**5.1 Khái niệm** 50](#_Toc60531237)

[**5.2 Các thành phần trong bản vẽ Class** 50](#_Toc60531238)

[**5.3 Relationship (Quan hệ)** 51](#_Toc60531239)

[**5.4 Đặc tả Class** 54](#_Toc60531240)

[**5.5 Mục đích sử dụng bản vẽ Class** 54](#_Toc60531241)

[**5.6 Cách xây dựng bản vẽ Class** 54](#_Toc60531242)

[**5.7 Áp dụng Class Diagram vào extension Auto Login** 55](#_Toc60531243)

[**5.8 Áp dụng Class Diagram vào Extension AutoCheck và CheckAuto** 58](#_Toc60531244)

[**6. Package Diagram - Sơ đồ gói** 60](#_Toc60531245)

[**6.1 Khái niệm** 60](#_Toc60531246)

[**6.2 Mục đích của Sơ đồ gói** 60](#_Toc60531247)

[**6.3 Các thành phần trong bản vẽ Package Diagram** 60](#_Toc60531248)

[**6.4 Các khái niệm cơ bản về Package Diagram** 61](#_Toc60531249)

[**6.5 Các bước vẽ Package diagram** 63](#_Toc60531250)

[**6.6 Ví dụ áp dụng** 63](#_Toc60531251)

[**6.7 Áp dụng Package diagram vào extension Autologin** 65](#_Toc60531252)

[**7. Object Diagram - Sơ đồ đối tượng** 66](#_Toc60531253)

[**7.1 Khái niệm** 66](#_Toc60531254)

[**7.2 Mục đích sơ đồ đối tượng** 66](#_Toc60531255)

[**7.3 So sánh giữa Class Diagram và Object Diagram** 67](#_Toc60531256)

[**7.4 Áp dụng object diagram vào source code php-training** 67](#_Toc60531257)

[**8. State Diagram - Sơ đồ trạng thái** 70](#_Toc60531258)

[**8.1 Khái niệm** 70](#_Toc60531259)

[**8.2 Các thành phần trong State Diagram** 71](#_Toc60531260)

[**8.3 Mục đích sơ đồ trạng thái** 71](#_Toc60531261)

[**8.4 Áp dụng State Diagram vào source code php-training** 72](#_Toc60531262)

[**9. Collaboration Diagram - Sơ đồ cộng sinh** 74](#_Toc60531263)

[**9.1 Khái niệm** 74](#_Toc60531264)

[**9.2 Các thành phần trong Collaboration Diagram** 74](#_Toc60531265)

[**9.3 Mục đích sơ đồ cộng sinh** 75](#_Toc60531266)

[**9.4 Áp dụng Collaboration Diagram vào source code php-training** 76](#_Toc60531267)

[**10. Component Diagram - Sơ đồ thành phần** 78](#_Toc60531268)

[**10.1 khái niệm** 78](#_Toc60531269)

[**10.2 Mục đích sơ đồ Component Diagram** 78](#_Toc60531270)

[**10.3 Các thành phần trong Component Diagram** 79](#_Toc60531271)

[**10.4 Áp dụng Component Diagram vào extension AutoCheck và CheckAuto** 80](#_Toc60531272)

[**10.5 Áp dụng Component Diagram vào source code php-training** 81](#_Toc60531273)

[**11. Interaction Diagram - Sơ đồ tương tác** 83](#_Toc60531274)

[**11.1 Khái niệm** 83](#_Toc60531275)

[**11.2 Mục đích Interaction Diagram** 84](#_Toc60531276)

[**12. Communication Diagram - Sơ đồ giao tiếp** 85](#_Toc60531277)

[**12.1 Khái niệm** 85](#_Toc60531278)

[**12.2 Mục đích sơ đồ Communication Diagram** 85](#_Toc60531279)

[**12.3 Các thành phần trong Communication Diagram** 85](#_Toc60531280)

[**12.4 Các bước vẽ Communication Diagram** 87](#_Toc60531281)

[**12.5 So sánh Communication Diagram & Sequence Diagram** 88](#_Toc60531282)

[**12.6 Áp dụng Communication Diagram vào extension AutoCheck và CheckAuto** 88](#_Toc60531283)

[**12.7 Áp dụng Communication Diagram vào extension Autologin** 90](#_Toc60531284)

[**III. EXTENSION DIỂM DANH ONLINE** 92](#_Toc60531285)

[**1. Giới thiệu** 92](#_Toc60531286)

[**2. Hướng dẫn code và cách cài đặt sử dụng các extension** 92](#_Toc60531287)

[**2.1 Hướng dẫn code extension Autocheck** 92](#_Toc60531288)

[**2.2 Hướng dẫn cài đặt extension Autocheck cho chrome** 100](#_Toc60531289)

[**2.3 Hướng dẫn code extension Checkauto** 104](#_Toc60531290)

[**2.4. Hướng dẫn cài đặt extension Checkauto cho chrome** 113](#_Toc60531291)

[**III. LARAVEL** 117](#_Toc60531292)

[**1. Hướng Dẫn Cài Laravel 8.x** 117](#_Toc60531293)

[**1.1 Giới thiệu** 117](#_Toc60531294)

[**1.2 Hướng dẫn cài đặt và tạo project laravel 8.x** 117](#_Toc60531295)

[***1.2.1 Cài đặt Wamp Server*** 117](#_Toc60531296)

[***1.2.2 Cài đặt Composer*** 119](#_Toc60531297)

[***1.2.3 Cách cài project laravel 8.x đơn giản*** 126](#_Toc60531298)

[**2. Tạo một gói Composer/Packagist cơ bản** 130](#_Toc60531299)

[**2.1 Tạo một kho lưu trữ với github** 130](#_Toc60531300)

[**2.2 Tạo Package** 136](#_Toc60531301)

[**3. Tạo Chức năng login thông qua thư viện jetstream** 145](#_Toc60531302)

[**3.1 Tạo chức năng login thông qua email** 145](#_Toc60531303)

[**3.2 Tạo chức năng login thông qua email hoặc username** 153](#_Toc60531304)

[**4. List User** 155](#_Toc60531305)

[**TÀI LIỆU THAM KHẢO** 162](#_Toc60531306)

**MỤC LỤC HÌNH**

[Hình 1: Các thành phần của mô hình Dynamic Model 17](#_Toc60531307)

[Hình 2: Ví dụ về Activity Diagram của hoạt động rút tiền từ ATM 18](#_Toc60531308)

[Hình 3: Ký hiệu về Swimlane 19](#_Toc60531309)

[Hình 4: Ký hiệu nút Start 19](#_Toc60531310)

[Hình 5: Ký hiệu nút End 19](#_Toc60531311)

[Hình 6: Ký hiệu Activity 19](#_Toc60531312)

[Hình 7: Ký hiệu Branch 20](#_Toc60531313)

[Hình 8: Ký hiệu Fork 20](#_Toc60531314)

[Hình 9: Ký hiệu Join 20](#_Toc60531315)

[Hình 10: Sơ đồ hoạt động - Mô hình hóa một bộ xử lý tạo tài liệu văn bản 23](#_Toc60531316)

[Hình 11: Ký hiệu của node 24](#_Toc60531317)

[Hình 12: Ký hiệu của Artifacts 24](#_Toc60531318)

[Hình 13: Ký hiệu quan hệ Association 25](#_Toc60531319)

[Hình 14: Ký hiệu quan hệ Dependence 25](#_Toc60531320)

[Hình 15: Ký hiệu về một device 25](#_Toc60531321)

[Hình 16: Ví dụ về Deployment Diagram với một người dùng Internet thông thường 26](#_Toc60531322)

[Hình 17: Deployment Diagram source code php-training 29](#_Toc60531323)

[Hình 18: Ví dụ Sequence Diagram cho hoạt động rút tiền ở ATM 30](#_Toc60531324)

[Hình 19: Tác nhân tương tác với hệ thống 31](#_Toc60531325)

[Hình 20: Ký hiệu về đối tượng trong bản vẽ Sequence Diagram 31](#_Toc60531326)

[Hình 21: Ký hiệu Obkect Lifetime 32](#_Toc60531327)

[Hình 22: Ký hiệu về Stimulus trong bản vẽ Sequence Diagram 32](#_Toc60531328)

[Hình 23: Cách biểu diễn một Self Message 33](#_Toc60531329)

[Hình 24: Cách biểu diễn Call Message 33](#_Toc60531330)

[Hình 25: Cách biểu diễn một Self Call Mesage 34](#_Toc60531331)

[Hình 26: Cách biểu diễn Return mesage 35](#_Toc60531332)

[Hình 27: Cách biểu diễn Self Return message 36](#_Toc60531333)

[Hình 28: Biểu diễn activation 36](#_Toc60531334)

[Hình 29: Biểu diễn Activation 37](#_Toc60531335)

[Hình 30: Guard Condition 37](#_Toc60531336)

[Hình 31: Frame 37](#_Toc60531337)

[Hình 32: Frame thể hiện là một vòng lặp ta sẽ sử dụng với ký hiệu context là chữ loop 38](#_Toc60531338)

[Hình 33: Frame thể hiện với ý nghĩa chỉ thực hiện khi điều kiện là true thì ta sẽ sử dụng với ký hiệu context là chữ opt 38](#_Toc60531339)

[Hình 34: Frame thể hiện với ý nghĩa là 2 điều kiện loại trừ nhau thì ta sẽ sử dụng với ký hiệu context là chữ alt 39](#_Toc60531340)

[Hình 35: Bản vẽ Use Case về ứng dụng ATM 41](#_Toc60531341)

[Hình 36: Ký hiệu của Actor 42](#_Toc60531342)

[Hình 37: Ký hiệu của Actor 42](#_Toc60531343)

[Hình 38: Ký hiệu về Use Case 42](#_Toc60531344)

[Hình 39: Ký hiệu quan hệ Association 43](#_Toc60531345)

[Hình 40: Ví dụ thể hiện Actor User sử dụng Use Case Login 43](#_Toc60531346)

[Hình 41: Ký hiệu quan hệ Generalization 43](#_Toc60531347)

[Hình 42: Ví dụ Actor User thừa kế toàn bộ quyền của Actor Guest 44](#_Toc60531348)

[Hình 43: Ký hiệu quan hệ Include 44](#_Toc60531349)

[Hình 44: Ví dụ về quan hệ Include giữa các Use Case 44](#_Toc60531350)

[Hình 45: Ký hiệu quan hệ Extend 44](#_Toc60531351)

[Hình 46: Ví dụ về quan hệ Extend giữa các Use Case 45](#_Toc60531352)

[Hình 47: Vùng System Boundary 45](#_Toc60531353)

[Hình 48: Use Case diagram extension autucheck & checkauto 49](#_Toc60531354)

[Hình 49: Ví dụ về Class Diagram của ATM 50](#_Toc60531355)

[Hình 50: Ký hiệu về Class 51](#_Toc60531356)

[Hình 51: Ví dụ về một Class 51](#_Toc60531357)

[Hình 52: Ví dụ về Association 52](#_Toc60531358)

[Hình 53: Ví dụ về Aggregation 52](#_Toc60531359)

[Hình 54: Ví dụ về Composition 53](#_Toc60531360)

[Hình 55: Ví dụ về Generalization 53](#_Toc60531361)

[Hình 56: Các nguồn thông tin có thể tìm Class dự kiến 55](#_Toc60531362)

[Hình 57: Bản vẽ Class dự kiến 56](#_Toc60531363)

[Hình 58: Xác định thuộc tính và quan hệ cho các lớp 57](#_Toc60531364)

[Hình 59: Bổ sung phương thức cho các lớp 57](#_Toc60531365)

[Hình 60: Class diagram Extension AutoCheck và CheckAuto 59](#_Toc60531366)

[Hình 61: Ví dụ về Package Diagram 61](#_Toc60531367)

[Hình 62: Cú pháp của tên có đủ điều kiện của một gói 62](#_Toc60531368)

[Hình 63: Các ký hiệu hiển thị các gói 62](#_Toc60531369)

[Hình 64: Ví dụ về import package 62](#_Toc60531370)

[Hình 65: Ví dụ về Access Package 63](#_Toc60531371)

[Hình 66: Các package được xác định 63](#_Toc60531372)

[Hình 67: Hai mô-đun đang truy cập lẫn nhau 64](#_Toc60531373)

[Hình 68: Điều hướng thông tin vận chuyển 64](#_Toc60531374)

[Hình 69: Package xử lý đơn hàng 65](#_Toc60531375)

[Hình 70: Object diagram 66](#_Toc60531376)

[Hình 71: Class diagram 67](#_Toc60531377)

[Hình 72: Object diagram 67](#_Toc60531378)

[Hình 73: Object diagram áp dụng vào source code php-training 69](#_Toc60531379)

[Hình 74: State diagram 70](#_Toc60531380)

[Hình 75: Ký hiệu chi tiết UML 71](#_Toc60531381)

[Hình 76: Sơ đồ state diagram hoàn chỉnh 71](#_Toc60531382)

[Hình 77: State diagram áp dụng vào source code php-training 73](#_Toc60531383)

[Hình 78: Collaboration diagram 74](#_Toc60531384)

[Hình 79: Object 75](#_Toc60531385)

[Hình 80: Message 75](#_Toc60531386)

[Hình 81: Seft link 75](#_Toc60531387)

[Hình 82: Backend 75](#_Toc60531388)

[Hình 83: State diagram áp dụng vào source code php-training 77](#_Toc60531389)

[Hình 84: Ký hiệu của component 78](#_Toc60531390)

[Hình 85: Ký hiệu của component dependency 78](#_Toc60531391)

[Hình 86: Component diagram 78](#_Toc60531392)

[Hình 87: Các kí hiệu trong Component diagram 79](#_Toc60531393)

[Hình 88: Component diagram for extension 80](#_Toc60531394)

[Hình 89: Component diagram for source code php-training 82](#_Toc60531395)

[Hình 90: Sơ đồ tuần tự 83](#_Toc60531396)

[Hình 91: Sơ đồ giao tiếp 83](#_Toc60531397)

[Hình 92: Sơ đồ giao tiếp 85](#_Toc60531398)

[Hình 93: Liên kết giữa actor và object 86](#_Toc60531399)

[Hình 94: Liên kết giữa 2 object 86](#_Toc60531400)

[Hình 95: Mô tả tương tác giữa các đối tượng 87](#_Toc60531401)

[Hình 96: Communication Diagram for extensions 90](#_Toc60531402)

[Hình 97: Communication Diagram for extension Autologin 91](#_Toc60531403)

[**Hình 98: Cây thư mục extension autocheck** 93](#_Toc60531404)

[**Hình 99: Giao diện nút tùy chỉnh và điều khiển chrome** 101](#_Toc60531405)

[**Hình 100: Giao diện công cụ khác** 101](#_Toc60531406)

[**Hình 101: Giao diện khi nhấn vào tiện ích mở rộng** 102](#_Toc60531407)

[**Hình 102: Tải tiện ích đã giải nén** 102](#_Toc60531408)

[**Hình 103: extension autocheck đã được cài** 103](#_Toc60531409)

[**Hình 104: Ghim extension vừa cài trong tiện ích** 103](#_Toc60531410)

[**Hình 105: Biểu tượng extension cạnh tiện ích** 104](#_Toc60531411)

[**Hình 106: Giao diện extension autocheck chứa nút Load và Clean** 104](#_Toc60531412)

[**Hình 107: Giao diện extension autocheck chứa nút Clean và Reload** 104](#_Toc60531413)

[**Hình 108: Cây thư mục extension checkauto** 105](#_Toc60531414)

[**Hình 109: Giao diện nút tùy chỉnh và điều khiển chrome** 113](#_Toc60531415)

[**Hình 110: Giao diện công cụ khác** 113](#_Toc60531416)

[**Hình 111: Tải tiện ích đã giải nén** 114](#_Toc60531417)

[**Hình 112: extension checkauto đã được cài** 115](#_Toc60531418)

[**Hình 113: Ghim extension vừa cài trong tiện ích** 115](#_Toc60531419)

[**Hình 114: Biểu tượng extension cạnh tiện ích** 116](#_Toc60531420)

[**Hình 115: Giao diện extension checkauto** 116](#_Toc60531421)

[**Hình 116: Biểu tượng Wamp Server** 117](#_Toc60531422)

[**Hình 117: Biểu tượng Wampserver khi khởi động** 118](#_Toc60531423)

[**Hình 118: Phiên bản PHP 7.3.12** 118](#_Toc60531424)

[**Hình 119: Giao diện trang web của composer** 119](#_Toc60531425)

[**Hình 120: Nhấn vào nút download composer** 119](#_Toc60531426)

[**Hình 121: Giao diện download composer** 120](#_Toc60531427)

[**Hình 122: Download file setup.exe** 121](#_Toc60531428)

[**Hình 123: Cửa sổ giao diện install** 121](#_Toc60531429)

[**Hình 124: Giao diện Installation Option** 122](#_Toc60531430)

[**Hình 125: Select Location composer setup** 122](#_Toc60531431)

[**Hình 126: Setting check your php** 123](#_Toc60531432)

[**Hình 127: Proxy setting composer** 124](#_Toc60531433)

[**Hình 128: Ready to install composer** 124](#_Toc60531434)

[**Hình 129: Hoàn tất cài composer** 125](#_Toc60531435)

[**Hình 130: Command Prompt và gõ composer** 126](#_Toc60531436)

[**Hình 131: Kết quả cài composer thành công** 126](#_Toc60531437)

[**Hình 132: Cửa sổ khi nhấn shift và chuột phải** 127](#_Toc60531438)

[**Hình 133: Giao diện Windows PowerShell** 127](#_Toc60531439)

[**Hình 134: Gõ câu lệnh tạo laravel** 128](#_Toc60531440)

[**Hình 135: Cài thành công laravel** 128](#_Toc60531441)

[**Hình 136: Folder demolaravel được tạo** 129](#_Toc60531442)

[**Hình 137: Giao diện laravel trên trình duyệt** 129](#_Toc60531443)

[**Hình 138: Nút tạo repo mới** 130](#_Toc60531444)

[**Hình 139: Nhập đầy đủ thông tin cần thiết và nhấn nút tạo** 131](#_Toc60531445)

[**Hình 140: Kho lưu trữ minh họa sau khi vừa tạo xong** 132](#_Toc60531446)

[**Hình 141: Lệnh clone một kho lưu trữ** 133](#_Toc60531447)

[**Hình 142: Copy project laravel có tên "blog" vào kho lưu trữ** 133](#_Toc60531448)

[**Hình 143: Đến đường dẫn của repo và gõ lệnh git add .** 133](#_Toc60531449)

[**Hình 144: Câu lệnh commit cam kết bắt buộc** 134](#_Toc60531450)

[**Hình 145: Câu lệnh push lên kho lưu trữ qua nhánh chính** 135](#_Toc60531451)

[**Hình 146: Kho lưu trữ đã được thêm thành công project laravel** 135](#_Toc60531452)

[**Hình 147: Form đăng ký tài khoản packagist** 136](#_Toc60531453)

[**Hình 148: Cách lấy API tooken để submit repo** 137](#_Toc60531454)

[**Hình 149: Tạo webhook** 137](#_Toc60531455)

[**Hình 150: Yêu cầu nhập lại mật khẩu** 138](#_Toc60531456)

[**Hình 151: Điều chỉnh thông tin cho webhook** 139](#_Toc60531457)

[**Hình 152: Webhook được tạo thành công** 140](#_Toc60531458)

[**Hình 153: Submit package** 140](#_Toc60531459)

[**Hình 154: Ckeck repo** 141](#_Toc60531460)

[**Hình 155: Lỗi chưa có file composer.json** 141](#_Toc60531461)

[**Hình 156: đặt file ở cùng cấp thư mục cha** 142](#_Toc60531462)

[**Hình 157: Ba câu lệnh để push lên git** 143](#_Toc60531463)

[**Hình 158: Check thành công** 143](#_Toc60531464)

[**Hình 159: Result submit** 144](#_Toc60531465)

[**Hình 160: Cài đặt thư viện jetstream** 145](#_Toc60531466)

[**Hình 161: Cài đặt gói hỗ trợ thư viện jetsteam** 146](#_Toc60531467)

[**Hình 162: Cài đặt npm** 147](#_Toc60531468)

[**Hình 163: Quá trình khởi chạy npm dev** 147](#_Toc60531469)

[**Hình 164: Hoàn tất khởi chạy npm dev** 148](#_Toc60531470)

[**Hình 165: Lỗi update table lên database** 148](#_Toc60531471)

[**Hình 166: Cập nhật số lượng ký tự cho email trong table create\_user** 149](#_Toc60531472)

[**Hình 167: Cập nhật số lượng ký tự email trong table create\_password\_resets** 149](#_Toc60531473)

[**Hình 168: Cập nhật số lượng ký tự cho table failed\_jobs** 150](#_Toc60531474)

[**Hình 169: Cập nhật số lượng ký tự cho table create\_session** 150](#_Toc60531475)

[**Hình 170: Cập nhật table create\_personal tránh xung đột** 151](#_Toc60531476)

[**Hình 171: Chọn tất cả các table và drop** 151](#_Toc60531477)

[**Hình 172: Chấp nhận drop tất cả các table** 151](#_Toc60531478)

[**Hình 173: Cập nhật tất cả table lên database** 152](#_Toc60531479)

[**Hình 174: Kích hoạt máy ảo laravel** 152](#_Toc60531480)

[**Hình 175: Chức năng login và register** 152](#_Toc60531481)

[**Hình 176: Thay đổi giá trị type và value** 153](#_Toc60531482)

[**Hình 177: Thông báo người dùng login thông qua username hoặc email** 153](#_Toc60531483)

[**Hình 178: Cập nhật giá trị đầu vào username hoặc email** 154](#_Toc60531484)

[**Hình 179: Kiểm điều kiện đầu vào** 155](#_Toc60531485)

[**Hình 180: Đường dẫn tới file web.php** 156](#_Toc60531486)

[**Hình 181: Route users** 156](#_Toc60531487)

[**Hình 182: Khai báo UserController** 156](#_Toc60531488)

[**Hình 183: Tạo UserController** 156](#_Toc60531489)

[**Hình 184: UserController** 157](#_Toc60531490)

[**Hình 185: Tạo hàm index trả về view** 157](#_Toc60531491)

[**Hình 186: đường dẫn tạo trang cho list user** 158](#_Toc60531492)

[**Hình 187: Giao diện cơ bản trang list users** 158](#_Toc60531493)

[**Hình 188: Cài đặt model Flight cho project** 159](#_Toc60531494)

[**Hình 189: Lấy tất cả thông tin uers** 159](#_Toc60531495)

[**Hình 190: Xuất thông tin của tất cả user ra trang list users** 160](#_Toc60531496)

[Hình 191: Kết quả trang list user 161](#_Toc60531497)

# **I. QUY TRÌNH LÀM VIỆC NHÓM**

Trong quá trình học tập, nhóm đã được nhận source code extension **autologin, mouse\_select, element\_customer, get\_cookie** tham khảo từ giáo viên và tiến hành viết lại báo cáo hướng dẫn cách tạo ra các extension đó ở phần trước và hiện tại tiến hành tìm hiểu, báo cáo về các lược đồ diagram cũng như áp dụng vào các project đã làm.

## **1. Bảng phân công công việc**

Tất cả các thành viên trong nhóm đều tham gia phân tích, tìm hiểu khái niệm về các lượt đồ diagram.

Ngoài ra, từ những khái niệm đã hiểu được, nhóm tiến hành vẽ sơ đồ diagram cho source code php-training hoặc các extension được giao từ giáo viên.

Các công việc được phân công như sau: (Chi tiết Sprint được nêu rõ ở tiến độ công việc)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Phân công** | **Sprint** | **Người thực hiện** | **Chức vụ** | **Tiến độ** |
| **Các khái niệm diagram** | - Chỉ ra khái niệm, nội dung, cách vẽ Deployment Diagram | 1 | Nguyễn Phương Tân | Nhóm trưởng |  |
| - Chỉ ra khái niệm, nội dung, cách vẽ Package Diagram | 1 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Communication Diagram | 2 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Sequence Diagram | 2 |
| **Áp dụng diagram** | - Cho ví dụ và áp dụng vào source code php-training với Deployment Diagram | 1 |
|  |  |
| **Extension** |  |  |
|  |  |
| **Các khái niệm diagram** | - Chỉ ra khái niệm, nội dung, cách vẽ Activity Diagram | 1 | Trần Phong Nhã | Thành viên |  |
| - Chỉ ra khái niệm, nội dung, cách vẽ Class Diagram | 1 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Use case Diagram | 2 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Interaction Overview Diagram | 2 |
| **Áp dụng diagram** | - Cho ví dụ và áp dụng vào extension Auto Login với Class Diagram | 1 |
|  |  |
| **Extension** |  |  |
|  |  |
| **Các khái niệm diagram** | - Chỉ ra khái niệm, nội dung, cách vẽ Object Diagram | 1 | Trần Triều Thuận | Nhóm phó |  |
| - Chỉ ra khái niệm, nội dung, cách vẽ State Diagram | 1 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Collaboration Diagram | 2 |
| - Chỉ ra khái niệm, nội dung, cách vẽ Component Diagram | 2 |
| **Áp dụng diagram** | - Cho ví dụ và áp dụng vào source code php-training với Object Diagram | 1 |
| - Cho ví dụ và áp dụng vào source code php-training với State Diagram | 2 |
| **Extension** |  |  |
|  |

## **2. Sprint Backlog**

Tiến hành cuộc họp 1**5 phút** hằng ngày để kiểm tra tiến độ thông qua **chat message online trên Discord** và yêu cầu sprint phải đạt được: Thực hiện đầy đủ về khái niệm cho các Diagram, nêu rõ nội dung áp dụng vào các source code hoặc extension tương ứng cho mỗi Diagram, áp dụng được extension điểm danh online.

Mục tiêu chuyển giao của sprint phải đáp ứng được các yêu cầu về tính năng của giảng viên.

Để đạt được điều đó:

1. Các thành viên phải hoàn thành các công việc đã được chuyển giao.
2. Tiến độ công việc chắc chắn được hoàn thành đúng tiến độ.
3. Công việc phải được thực hiện rõ ràng và ổn định.

## **3. Kết thúc một sprint**

* Sau khi kết thúc Sprint, Nhóm trưởng sẽ xác nhận phần công việc hoàn thành và chưa hoàn thành.
* Nhóm sẽ tổng kết lại những thuận lợi và khó khăn trong sprint vừa qua.
* Lập kế hoạch các công việc sẽ làm cho các sprint tiếp theo.

## **4. Tiến độ công việc**

**4.1 Sprint 1 (16/11/2020 – 22/11/2020)**

1. **Các chức năng cần thực hiện trong sprint này:**

* Nêu các khái niệm, định nghĩa và cách vẽ các Diagram: Deployment Diagram, Package Diagram, Activity Diagram, Class Diagram, Object Diagram, State Diagram.
* Đưa ra ví dụ, áp dụng vào source code php-training hoặc các extension mà giảng viên đã cung cấp theo các Diagram: Deployment Diagram, Class Diagram, Object Diagram.
* Hoàn thành extension **check-presence-student** (Dùng để cho sinh viên nhập ID tương ứng và lưu lên database áp dụng trong source code php-training).

1. **Thông qua các mục tiêu trên các thành viên được phân chia chức năng theo thời gian yêu cầu:**

* **Nguyễn Phương Tân**
  + Thực hiện làm khái niệm, mục đích sử dụng cho Deployment Diagram, Package Diagram trễ nhất ngày 18/11.
  + Áp dụng Deployment Diagram vào source code php-training và sẽ ra sơ đồ cho source code trễ nhất ngày 20/11.
  + Xuất trạng thái người dùng (Present hoặc Not present) đã gửi ID thông qua extension **check-presence-student** xuống danh sách user trễ nhất 23 giờ ngày 22/11.
* **Trần Triều Thuận**
  + Thực hiện làm khái niệm, mục đích sử dụng cho Object Diagram, State Diagram trễ nhất ngày 18/11.
  + Áp dụng Object Diagram vào source code php-training và sẽ ra sơ đồ cho source code trễ nhất ngày 20/11.
  + Gửi ID mà người dùng nhập vào thông qua extension **check-presence-student** lên database trễ nhất 15 giờ ngày 22/11.
* **Trần Phong Nhã**
  + Thực hiện làm khái niệm, các bước vẽ Class Diagram, Activity Diagram trễ nhất ngày 18/11.
  + Áp dụng Class Diagram vào extension Auto Login và sẽ ra sơ đồ cho extension trễ nhất ngày 20/11.
  + Làm giao diện cho extension **check-presence-student** và xuất ra trạng thái số tiết vắng của người dùng xuống trang danh sách user trễ nhất 15 giờ ngày 22/11.

**c) Sprint backlog**

Nhóm tiến hành họp nhóm online 15 phút vào 8 – 9 giờ tối hằng ngày thông qua Discord nhằm nêu rõ các khó khăn, kết quả đã làm trong ngày.

Giúp nhóm trưởng nắm được tình hình và trực tiếp hướng dẫn khắc phục các khó khăn bằng đàm thoại thông qua Stream chia sẻ màn hình với Discord.

**d) Kết thúc Sprint 1**

Kết thúc sprint 1. Kết quả hoàn thành của các thành viên trong nhóm được liệt kê:

* **Trần Triều Thuận**
  + Hoàn thành và chuyển giao khái niệm, mục đích sử dụng cho Object Diagram, State Diagram cho nhóm trưởng ngày 17/11.
  + Chuyển giao phần áp dụng Object Diagram vào source code php-training và sẽ ra sơ đồ cho source code ngày 19/11.
  + Chuyển giao chức năng gửi ID mà người dùng nhập vào thông qua extension **check-presence-student** lên database ngày 22/11.
  + **=> Kết quả: Đạt 100%**
* **Nguyễn Phương Tân**
  + Hoàn thành khái niệm, mục đích sử dụng cho Deployment Diagram, Package Diagram ngày 17/11.
  + Hoàn thành phần áp dụng Object Diagram vào source code php-training và sẽ ra sơ đồ cho source code ngày 19/11.
  + Hoàn thành xuất trạng thái người dùng (Present hoặc Not present) đã gửi ID thông qua extension **check-presence-student** xuống danh sách user ngày 22/11.
  + **=> Kết quả**: **Đạt 100%**
* **Trần Phong Nhã** 
  + Hoàn thành và chuyển giao khái niệm, các bước vẽ Class Diagram, Activity Diagram cho nhóm trưởng ngày 17/11.
  + Chuyển giao phần áp dụng Class Diagram vào extension Auto Login và sẽ ra sơ đồ cho extension ngày 19/11.
  + Hoàn thành giao diện cho extension **check-presence-student** và xuất ra trạng thái số tiết vắng của người dùng xuống trang danh sách user vào 21/11.
  + **=> Kết quả: Đạt 100%**

Đánh giá từ nhóm trưởng:

* Kết thúc sprint 1, tất cả các yêu cầu từ giảng viên hoàn thành với mức độ 99%.
* Tiến độ hoàn thành công việc của các thành viên làm tốt.

Lập kế hoạch cho Sprint 2:

* Nêu các khái niệm, định nghĩa và cách vẽ và áp dụng cho các Diagram: Communication Diagram, Collaboration Diagram, Use case Diagram.
* Xây dựng extension auto-check-user để tự động nhập số tiết vắng trong danh sách user của source code php-training.

# **II. DIAGRAMS**

## **1. Activity Diagram - Bản vẽ hoạt động**

### **1.1 Mở đầu**

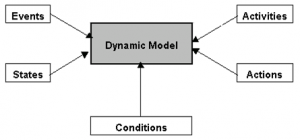
Theo UML thì hệ thống có thể được mô tả theo 2 mô hình tĩnh (Static Model) và mô hình động (Dynamic Model).

**Static Model**: mô tả cấu trúc của hệ thống bao gồm các bản vẽ Class Diagram, Object Diagram, Component Diagram và Deployment Diagram.

**Dynamic Model**: mô tả các hoạt động bên trong hệ thống bao gồm các bản vẽ Activity Diagram, State Diagram, Sequence Diagram, Collaboration Diagram.

Hai bản vẽ của mô hình thường xuyên được sử dụng trong thiết kế hệ thống phần mềm là **Activity Diagram** và **Sequence Diagram**.

### **1.2 Các thành phần của Dynamic Model**



Hình : Các thành phần của mô hình Dynamic Model

**Events**: là các sự kiện, mô tả một hoạt động bên ngoài tác động vào đối tượng và được đối tượng nhận biết và có phản hồi lại.

**Activities**: các mô tả một hoạt động trong hệ thống. Hoạt động có thể do một hoặc nhiều đối tượng thực hiện.

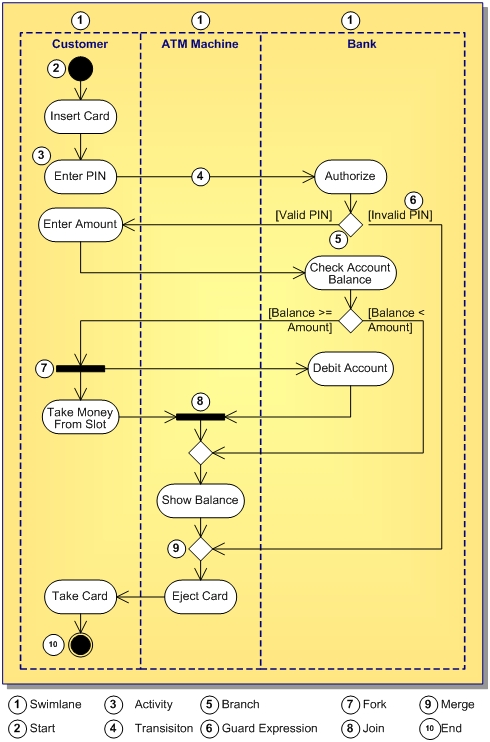
**States**: là các trạng thái của một đối tượng trong hệ thống, được mô tả bằng giá trị của một hoặc nhiều thuộc tính.

**Actions**: các hành động của đối tượng.

**Conditions**: mô tả các điều kiện.

### **1.3 Activity Diagram**

Activity Diagram là bản vẽ tập trung vào mô tả các hoạt động, luồng xử lý bên trong hệ thống. Dùng để mô tả các quy trình trong hệ thống, các luồng của một chức năng hoặc các hoạt động của một đối tượng.



Hình 2: Ví dụ về Activity Diagram của hoạt động rút tiền từ ATM

**Phân tích các ký hiệu:**

**Swimlane**: được dùng để xác định đối tượng nào tham gia hoạt động nào trong một quy trình. Ví dụ ở trên Customer thì Insert Card còn ATM Machine thì Show Balance.



Hình 3: Ký hiệu về Swimlane

**Nút Start, End**:

Start thể hiện điểm bắt đầu quy trình.

Start

Hình 4: Ký hiệu nút Start

End thể hiện điểm kết thúc quy trình.



Hình 5: Ký hiệu nút End

**Activity:** Activity mô tả một hoạt động trong hệ thống. Các hoạt động này do các đối tượng thực hiện.



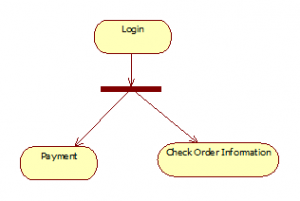
Hình 6: Ký hiệu Activity

**Branch:** thể hiện rẽ nhánh trong mệnh đề điều kiện.

Branch

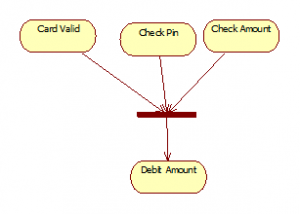
Hình 7: Ký hiệu Branch

**Fork:** thể hiện cho trường hợp thực hiện xong một hoạt động rồi sẽ rẽ nhánh thực hiện nhiều hoạt động tiếp theo.



Hình 8: Ký hiệu Fork

**Join**: Cùng ký hiệu với Fork nhưng thể hiện trường hợp phải thực hiện hai hay nhiều hành động trước rồi mới thực hiện hành động tiếp theo.



Hình 9: Ký hiệu Join

### **1.4 Cách xây dựng Activity Diagram**

**Bước 1**: Xác định các hoạt động cần mô tả. Xem xét bản vẽ Use Case để xác định yêu cầu nào cần mô tả.

**Bước 2**: Xác định trạng thái đầu tiên và trạng thái kết thúc.

**Bước 3**: Xác định các hoạt động tiếp theo.

Xuất phát từ điểm bắt đầu, phân tích để xác định các hoạt động tiếp theo cho đến khi gặp điểm kết thúc để hoàn tất bản vẽ.

### **1.5 Khi nào cần sử dụng bản vẽ Activity Diagram**

- Phân tích yêu cầu để hiểu rõ hệ thống.

- Phân tích Use Case.

- Cung cấp thông tin hỗ trợ để thiết kế bản vẽ Sequence Diagram.

### **1.6 Lợi ích của bản vẽ Activity Diagram**

Bản vẽ Activity diagram có rất nhiều lợi ích cho người dùng:

* Thể hiện được tính logic của một thuật toán.
* Mô tả được các bước thực hiện một UML use case.
* Minh họa lại quy trình làm việc giữa người dùng và hệ thống.
* Đơn giản hóa và cải thiện các quy trình thông qua việc mô phỏng và làm rõ các trường hợp của từng hành động.
* Mô hình hóa các yếu tố phần mềm. Ví dụ như phương pháp, chức năng, hoạt động.

### **1.7 Áp dụng vào ví dụ**

Ví dụ về sơ đồ hoạt động mô tả quy trình làm việc cho một quy trình tạo tài liệu sẽ thông qua các bước sau:

**Bước 1**: Mở gói ứng dụng xử lý văn bản.

**Bước 2**: Tạo một tệp mới.

**Bước 3**: Lưu tệp bằng một tên duy nhất.

**Bước 4**: Nhập tài liệu.

- Nếu cần có đồ họa.

+ Mở gói ứng dụng đồ họa hình ảnh, tạo phần đồ họa và dán phần đồ họa hình ảnh đó vào tài liệu.

- Nếu cần có bảng tính.

+ Mở gói ứng dụng bảng tính, tạo bảng tính và dán bảng tính vào tài liệu.

- Ngược lại thì sẽ tiếp tục các bước kế tiếp.

**Bước 5**: Lưu tập tin.

**Bước 6**: In tài liệu.

**Bước 7**: Thoát khỏi gói ứng dụng xử lý văn bản.

Như vậy ta có sơ đồ sau:



Hình 10: Sơ đồ hoạt động - Mô hình hóa một bộ xử lý tạo tài liệu văn bản

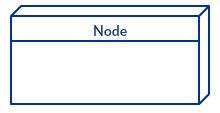
## **2. Deployment Diagram – Bản Vẽ Triển Khai**

**Deployment Diagram** – bản vẽ triển khai là một loại sơ đồ cấu trúc được sử dụng để mô hình hóa các khía cạnh của một hệ thống phần mềm hoặc đối tượng được triển khai như thế nào.

### **2.1 Những ký hiệu trong bản vẽ triển khai**

Để vẽ một sơ đồ triển khai, trước tiên bạn cần làm quen với các ký hiệu sơ đồ triển khai sau đây và các phần tử của sơ đồ triển khai.

**Nodes**



Hình 11: Ký hiệu của node

Một node, được biểu diễn dưới dạng khối lập phương, là một thực thể vật lý thực thi một hoặc nhiều thành phần, hệ thống con nào đó.

Một node có thể là một thiết bị phần cứng, hoặc một thành phần phần mềm nào đó.

**Artifacts**



Hình 12: Ký hiệu của Artifacts

Artifacts là những yếu tố cụ thể do một quá trình phát triển tạo nên. Ví dụ về Artifact như là thư viện, kho lưu trữ, tệp cấu hình hoặc tệp thực thi, v.v..

Artifacts sẽ được triển khai trên các node.

**Relationship**

Trong sơ đồ này có sự dụng 2 quan hệ **Association** và **Dependency** để thể hiện và liên kết mối quan hệ giữa các node với nhau.

Associate

Hình 13: Ký hiệu quan hệ Association

Component 2

Hình 14: Ký hiệu quan hệ Dependence

**Giải thích về hai loại quan hệ trên:**

* **Quan hệ Association**: Giữa hai đối tượng của hai class có sự ghép cặp với nhau (Ví dụ: vợ - chồng, thầy – trò,…). Ta tập hợp các kết nối cùng ý nghĩa giữa các đối tượng của hai lớp sẽ tạo thành liên kết Association.
* **Quan hệ Dependency:** Là quan hệ giữa 2 phần tử hoặc node trong một mô hình mà trong đó nếu thay đổi ở node, phần tử này (node, phần tử độc lập) có thể gây ra thay đổi ở node, phần tử kia (node, phần tử phụ thuộc)

**Devices**



Hình 15: Ký hiệu về một device

Device là một nút được sử dụng để đại diện cho phần cứng, tài nguyên vật lý trong hệ thống. Ví dụ về thiết bị là máy chủ ứng dụng.

### **2.2 Cách vẽ một sơ đồ triển khai**

Node và artifact là những yếu tố thiết kế của việc triển khai, trước khi thực hiện vẽ sơ đồ triển khai Deployment Diagram, tất cả các node và các mối quan hệ giữa các node đó phải được xác định rõ ràng.

**Bước 1:**

Xác định mục đích của sơ đồ triển khai của bạn. Và để làm như vậy, bạn cần xác định các node và device trong hệ thống mà bạn muốn hình dung ra sơ đồ.

Bạn nên cần có kinh nghiệm về kiến trúc hệ thống để làm được việc này.

**Bước 2:**

Tìm ra mối quan hệ giữa các nút và thiết bị. Khi bạn biết chúng được kết nối như thế nào, hãy tiến hành thêm các liên kết giao tiếp giữa chúng vào sơ đồ.

**Bước 3:**

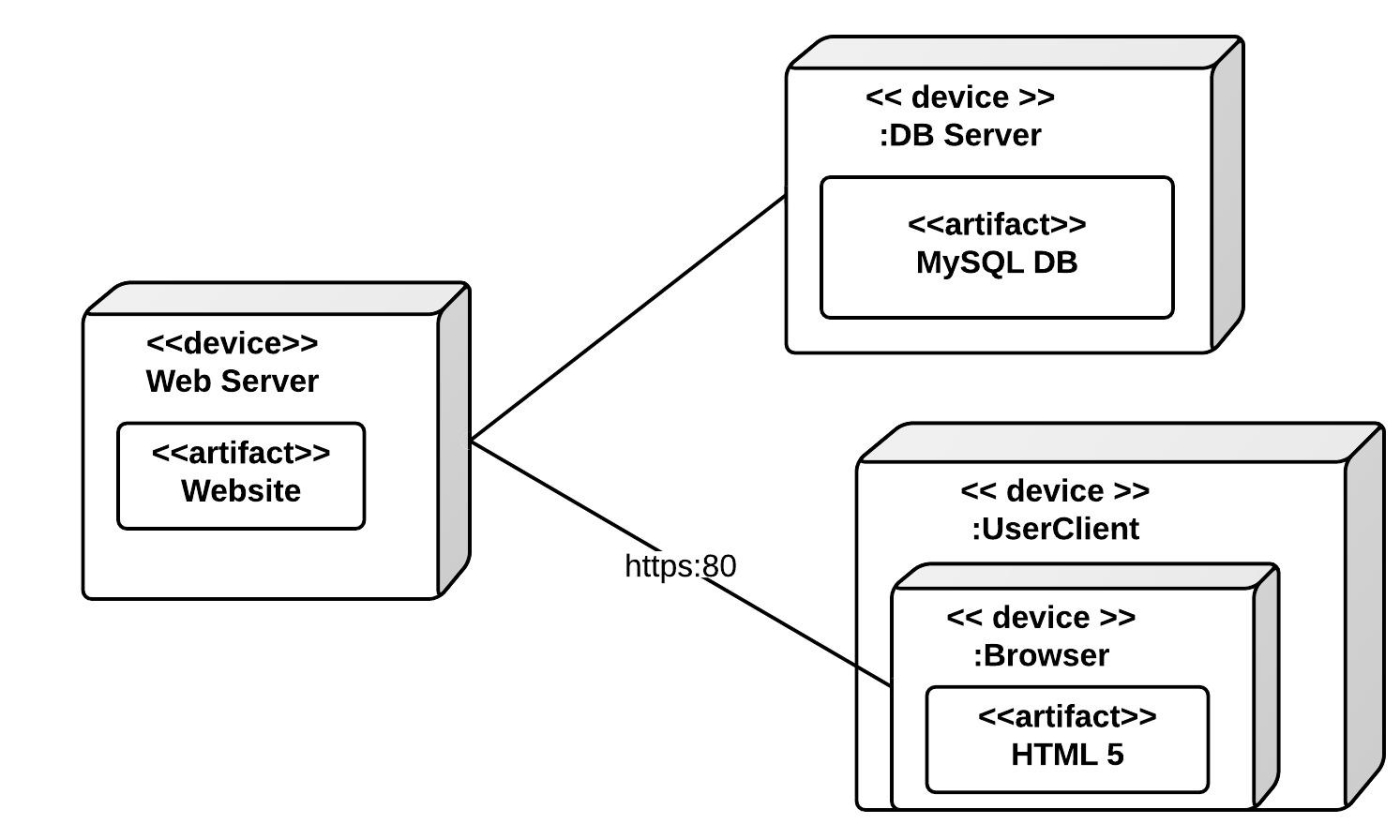
Xác định những yếu tố khác như các thành phần component, các đối tượng, các hoạt động mà bạn cần thêm để hoàn thành sơ đồ.

**Bước 4:**

Thêm phụ thuộc giữa các thành phần và đối tượng theo yêu cầu.

**Ví dụ:**

Sơ đồ triển khai sau thể hiện hoạt động của trình phát video HTML5 trong trình duyệt:



Hình 16: Ví dụ về Deployment Diagram với một người dùng Internet thông thường

Ví dụ này cho thấy một sơ đồ triển khai cơ bản. Ta có một máy chủ web, một máy chủ cơ sở dữ liệu và máy khách client mà người dùng xem trang web.

### **2.3 Mục đích của sơ đồ triển khai**

Sơ đồ triển khai (**Deployment Diagram**) hầu hết được sử dụng bởi quản trị viên hệ thống, kỹ sư mạng,… Các sơ đồ này được sử dụng với mục đích chỉnh là mô tả cách phần mềm được triển khai vào hệ thống phần cứng.

Để phần mềm hoạt động hiệu quả và tốc độ nhanh hơn, phần cứng cũng phải có chất lượng tốt. Nó phải được thiết kế hiệu quả để làm cho phần mềm hoạt động tốt và tạo ra kết quả chính xác trong thời gian ngắn nhất có thể.

* Sơ đồ triển khai cho thấy cấu trúc của hệ thống diễn ra như thế nào.
* Biết được phần cứng sẽ được sử dụng để triển khai hệ thống và các liên kết giữa các phần cứng khác nhau như thế nào.
* Mô hình hóa các thiết bị phần cứng và các đường giao tiếp giữa chúng.

**Link tham khảo và thực hiện vẽ online đặc biệt dành riêng cho Deployment Diagram:** <https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=DeploymentDiagram>

### **2.4 Áp dụng Deployment Diagram vào source code php-training**

**Phân tích source code php-training:**

Source code php-training xây dựng với mục đích sử dụng nhằm quản lý danh sách thành viên users cho một phòng ban, công ty hoặc lớp học,… Nó bao gồm các hoạt động quản lý thông tin người dùng, thêm, xoá, sửa, đổi mật khẩu người dùng.

* Người dùng phải đăng ký tài khoản đề được quản trị viên quản lý thông tin cá nhân cho các mục đích công việc, học tập. Thông tin người dùng bao gồm username, fullname, email, password.
* Các người dùng trong ứng dụng được quản lý thông tin theo từng ID, mỗi người dùng có một ID riêng khác nhau. Khi người dùng có thông tin cần thay đổi, các hành động xóa hoặc sửa tùy theo mục đích thay đổi sẽ được thực thi.
* Khi người dùng quên mật khẩu, người dùng phải nhập email hiện tại đang đăng ký và phải có xác thực thông qua email trên phiên làm việc hiện tại thì mới được phép tiếp tục thay đổi mật khẩu mới.
* Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

**Bước 1: Xác định các Node cần có và bố trí các thành phần lên node tương ứng.**

Dựa vào mục đích của hệ thống và kiến trúc hệ thông trên, ta xác định các node triển khai:

**Node 1:** Client – là người dùng thống thường, là người đăng ký tài khoản vào hệ thống để được quản lý bởi quản trị viên.

**Node 2:** Là máy chủ web server để tương tác giữa người dùng với ứng dụng web php-training

**Node 3:** Là một application server chứa một Artifact thư viện PHPMailer được sử dụng để giúp cho người dùng có thể xác thực tài khoản, đổi mật khẩu hoặc nhận thông báo mới từ ứng dụng web đến email người dùng.

**Node 4:** Database thì sẽ được đặt trên một server khác là Database server.

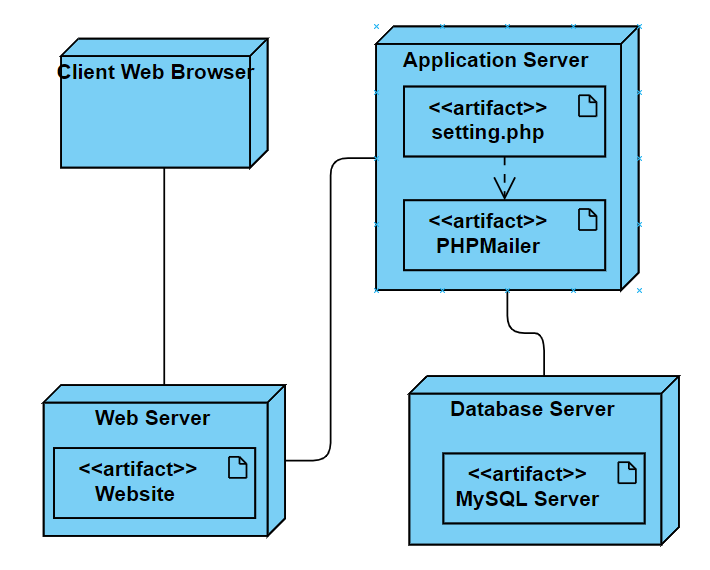
**Bước 2: Xác định mối quan hệ giữa các node**

Từ phần xác định nodes ở bước 1, ta tiến hành xác định mối quan hệ giữa chúng:

* Ta thấy **Node 1** với **client** sẽ yêu cầu truy cập đến Node 2 là **web server** để có thể trả về giao diện web cho người dùng.
* Nếu có yêu cầu liên quan đến tính năng gửi email như đổi mật khẩu, kích hoạt tài khoản, quên mật khẩu,… thì **Node 2** (web server) sẽ cần yêu cầu đến **Node 3** (sử dụng application server có một thư viện PHPMailer) để thực hiện hành động trên.
* Sau khi các hành động được thực hiện và thay đổi sẽ gửi dữ liệu đến và sử dụng đến **Node 4** là Database server.

Từ đó ta phân tích các thành phần cần có trong các node vào thực hiện **bước 3** và **bước 4** để hoàn thành sơ đồ:

Xem xét các thành phần trên với nhau để hoàn tất chức năng, chúng ta sẽ xác định các quan hệ của chúng. Biểu diễn lên bản vẽ chúng ta sẽ có Deployment Diagram như sau:



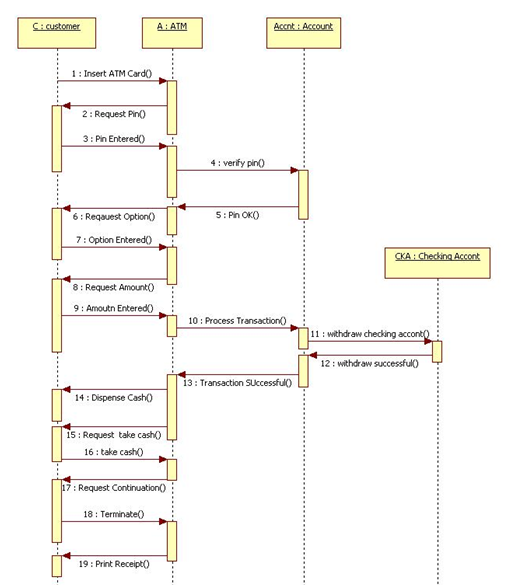
Hình 17: Deployment Diagram source code php-training

Theo mô hình trên, chúng ta thấy rằng hệ thống được triển khai trên 3 server là Web server, Application server và Database server. Ngoài ra còn bao gồm thiết bị truy cập đầu cuối Client.

## **3. Sequence Diagram - Bản vẽ tuần tự**

### **3.1 Khái niệm**

Sequence Diagram là bản vẽ mô tả sự tương tác của các đối tượng để tạo nên các chức năng của hệ thống. Bản vẽ mô tả sự tương tác theo thời gian nên phù hợp với việc sử dụng để thiết kế và cài đặt chức năng cho hệ thống phần mềm.



Hình 18: Ví dụ Sequence Diagram cho hoạt động rút tiền ở ATM

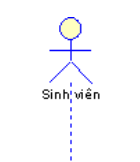
**Sequence Diagram Có 2 dạng:**

* Dạng tổng quát: thể hiện cả vòng lặp và rẽ nhánh
* Dạng cụ thể: miêu tả một kịch bản cụ thể

### **3.2 Các thành phần của Sequence Diagram**

**a. Actor:**

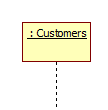
Là tác nhân bên ngoài tương tác với hệ thống.



Hình 19: Tác nhân tương tác với hệ thống

**b. Objects**

Object mô tả một đối tượng trong hệ thống. Để phân biệt với Class, Object có dấu “:” (dấu hai chấm) phía trước tên của nó.



Hình 20: Ký hiệu về đối tượng trong bản vẽ Sequence Diagram

**c. Object lifetime**

Là đường gạch chấm bên dưới đối tượng thể hiện thời gian sống của đối tượng.

Thời gian sống của mỗi đối tượng được mô tả theo một đường thẳng đứng. Thông thường thời gian trôi theo chiều từ trên xuống dưới.



Hình 21: Ký hiệu Obkect Lifetime

**d. Stimulus (message)**

Thể hiện tin nhắn/thông báo từ một đối tượng này tương tác với một đối tượng khác.

Thông tin gửi đi có thể là 1 yêu cầu thực thi hệ thống hay lời gọi hàm khởi tạo đối tượng, hủy đối tượng, cập nhật đối tượng,...

seq-Notation2

Hình 22: Ký hiệu về Stimulus trong bản vẽ Sequence Diagram

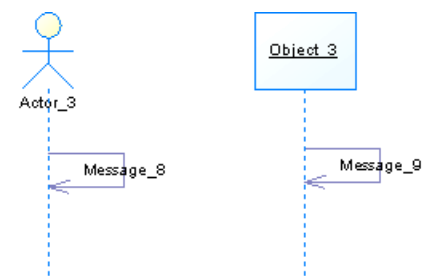
**e. Phân loại thông điệp**

* Message
* Self Message
* Call Message
* Self Call Message
* Return Message
* Self Return Message

**Self Message** là thông báo tin nhắn về một thông điệp nào đó được gởi từ đối tượng A đến cho chính nó, yêu cầu đối thực hiện một hành động hay một xử lý nào đó.

Ví dụ: Hệ thống kiểm tra thông tin đăng nhập,..

Self Message được biểu diễn như sau:

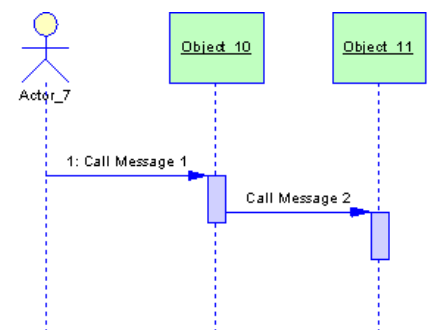


Hình 23: Cách biểu diễn một Self Message

**Call Message** được ý nghĩa là dùng để biểu diễn đối tượng A đang gọi thủ tục của đối tượng B.

Ví dụ: Màn hình chính gọi thủ tục Show (hiển thị) màn hình chi tiết sản phẩm,…

Call Message được biểu diễn như sau:



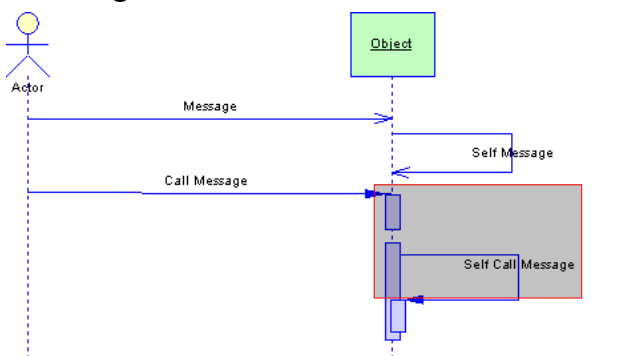
Hình 24: Cách biểu diễn Call Message

**Self Call Message** có ý nghĩa là thông báo tin nhắn về một thông điệp nào đó mà đối tượng gọi thủ tục/hàm của chính nó.

Ví dụ:

* Kiểm tra thông tin username/password
* Hiển thị thông tin thời khóa biểu lớp lên màn hình
* Xem thời khóa biểu lớp
* …

Self Call Message được biểu diễn như sau:

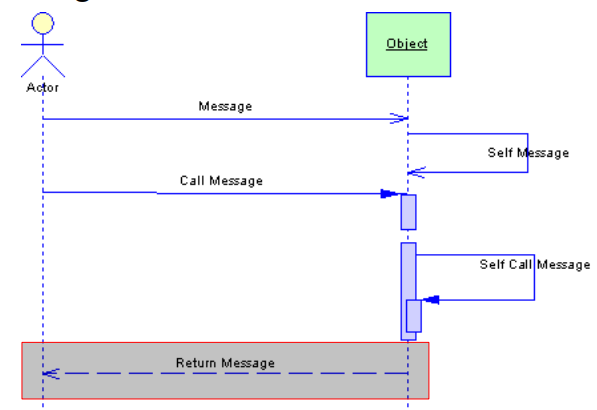


Hình 25: Cách biểu diễn một Self Call Mesage

**Return Message** có ý nghĩa rằng thông báo tin nhắn về một thông điệp nào đó gởi ngược về từ đối tượng nhận (kết quả ứng với thông điệp đối tượng A yêu cầu đối tượng B thực hiện).

Ví dụ: Thông điệp trả về kết quả kiểm tra đăng nhập hệ thống (thành công/thất bại).

Return Message được biểu diễn như sau:

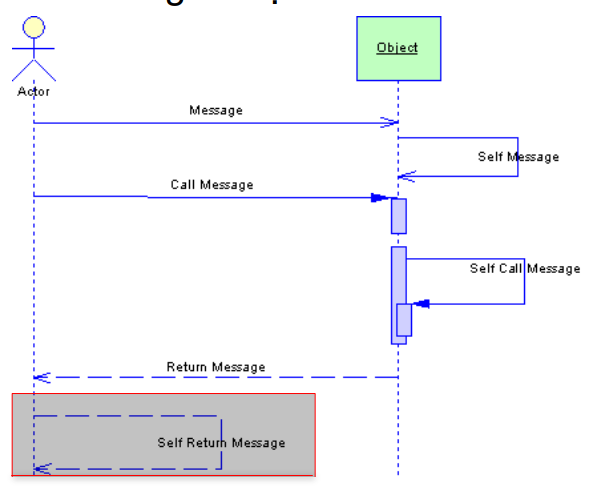


Hình 26: Cách biểu diễn Return mesage

**Self Return Message** là thông điệp trả kết quả từ “thông điệp yêu cầu” của chính đối tượng.

Ví dụ: Đối tượng người dùng kiểm tra thông tin đăng nhập hệ thống.

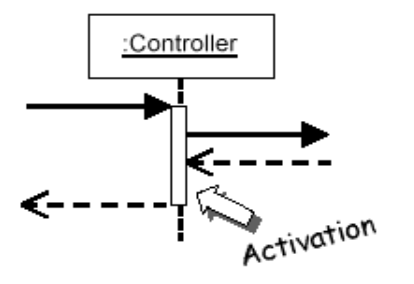
Self Return Message được biểu diễn như sau:



Hình 27: Cách biểu diễn Self Return message

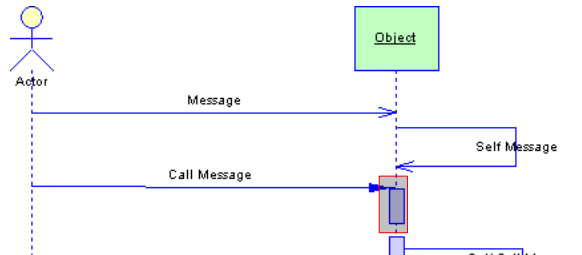
**f. Activation**:

**Activation** được sử dụng để mô tả thời gian cần để thực thi một hành động nào đó. Nó sẽ được tạo trong chu kỳ sống của một đối tượng.



Hình 28: Biểu diễn activation

Activation được biểu diễn như sau:



Hình 29: Biểu diễn Activation

**g. Guard Condition**

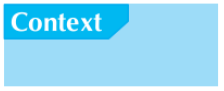
Guard Condition được dùng để thể hiện việc kiểm tra điều kiện trước khi gửi thông điệp.



Hình 30: Guard Condition

**h. Frame**

Dùng để thể hiện ngữ cảnh cho sơ đồ trình tự.

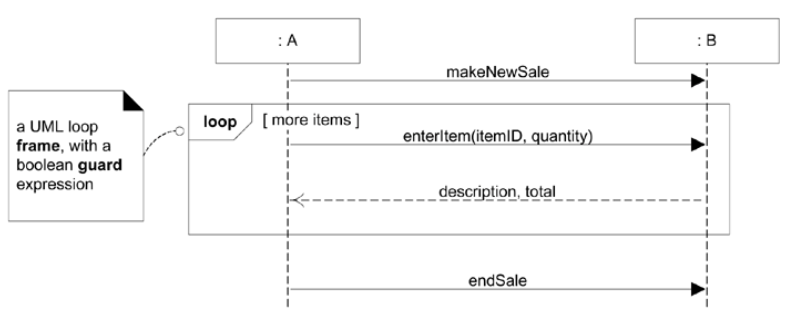


Hình 31: Frame

**i. Các loại frame**

1. Frame thể hiện vòng lặp.

Đối với frame thể hiện là một vòng lặp ta sẽ sử dụng với ký hiệu context là chữ **loop**.



Hình 32: Frame thể hiện là một vòng lặp ta sẽ sử dụng với ký hiệu context là chữ loop

1. Frame với ý nghĩa chỉ thực hiện khi điều kiện là **true**.

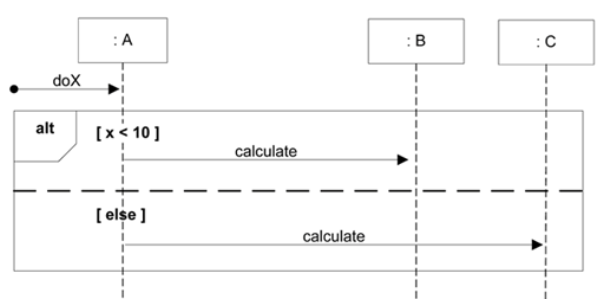
Đối với frame thể hiện với ý nghĩa chỉ thực hiện khi điều kiện là **true** thì ta sẽ sử dụng với ký hiệu context là chữ **opt**.



Hình 33: Frame thể hiện với ý nghĩa chỉ thực hiện khi điều kiện là true thì ta sẽ sử dụng với ký hiệu context là chữ opt

1. Frame thể hiện điều kiện loại trừ nhau.

Đối với frame thể hiện với ý nghĩa là 2 điều kiện loại trừ nhauthì ta sẽ sử dụng với ký hiệu context là chữ **alt**.



Hình 34: Frame thể hiện với ý nghĩa là 2 điều kiện loại trừ nhau thì ta sẽ sử dụng với ký hiệu context là chữ alt

**Như vậy ta có thể tổng kết lại các Selection trên:**

* **if -> (opt)**
* **if/else -> (alt)**
* **loop -> (loop)**

**k. Axes:** Trục tọa độ, trục ngang thể hiện các đối tượng, trục đứng thể hiện thời gian.

**l. Chú ý:** Thanh hình chữ nhật mô tả sự thực thi của một tác vụ để đáp ứng lại thông điệp gửi đến. Độ dài của thanh chữ nhật phản ánh thời gian thực thi của tác vụ và tính chất lồng nhau (nested) giữa chúng.

### **3.3 Các bước xây dựng Sequence Diagram**

**Bước 1**: Xác định ngữ cảnh

Xác định chức năng cần thiết kế. Dựa vào Use Case Diagram để xác định xem chức năng nào cần thiết kế. (Ngữ cảnh có thể là use case hoặc 1 trường hợp cụ thể của use case).

**Bước 2**: Dựa vào Activity Diagram để xác định actor, object tham gia vào lược đồ, các bước thực hiện theo các hoạt động, chức năng khác nhau.

**Bước 3**: Xác định lifeline cho các object.

**Bước 4:** Sắp xếp các thông điệp (operation).

**Bước 5**: Vẽ và kiểm tra lại Sequence Diagram.

### **3.4 Ứng dụng Sequence Diagram**

- Thiết kế các chức năng cho hệ thống. Mô tả tương tác giữa actor và các đối tượng hệ thống.

- Kiểm chứng và bổ sung method cho các Class.

- Sử dụng trong việc phân tích và đưa vào code các chức năng.

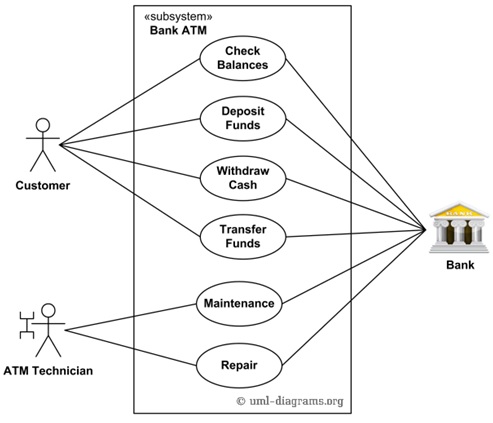
## **4. Use Case Diagram - Bản vẽ Use Case**

### **4.1 Khái niệm**

Use Case được mô tả trong ngôn ngữ UML qua sơ đồ Use Case (Use Case Diagram). Một sơ đồ Use Case chứa các phần tử mô hình biểu thị hệ thống và chỉ ra các mối quan hệ giữa các Use Case.

### **4.2 Các thành phần trong bản vẽ Use Case**

Ví dụ về Use Case Diagram:



Hình 35: Bản vẽ Use Case về ứng dụng ATM

Nhìn bản vẽ này chúng ta thấy có hai người dùng là Customer và ATM Technician và một đối tượng sử dụng hệ thống là Bank. Bên cạnh đó nó mô tả các chức năng của hệ thống và người dùng nào dùng chức năng gì.

Điều này giúp chúng ta hình dung được là chúng ta sẽ xây dựng hệ thống với những chức năng gì? Cho ai dùng?

#### **4.2.1 Actor**

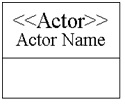
Actor được dùng để chỉ người sử dụng hoặc một đối tượng nào đó bên ngoài tương tác với hệ thống.

Actor được biểu diễn như sau:



Hình 36: Ký hiệu của Actor

Hoặc



Hình 37: Ký hiệu của Actor

#### **4.2.2 Use Case**

Use Case:là chức năng mà các Actor sẽ sử dụng. Nó được ký hiệu như sau:



Hình 38: Ký hiệu về Use Case

Việc xác định các chức năng mà Actor sử dụng bạn sẽ xác định được các Use Case cần có trong hệ thống.

#### **4.2.3 Relationship (Quan hệ)**

Relationship hay còn gọi là connector được sử dụng để kết nối giữa các đối tượng với nhau tạo nên bản vẽ Use Case.

Có các kiểu quan hệ cơ bản sau:

- Association

- Generalization

- Include

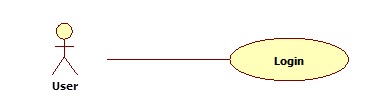
- Extend

**+ Quan hệ Association:**

Association

Hình 39: Ký hiệu quan hệ Association

**Association** thường được dùng để mô tả mối quan hệ giữa Actor và Use Case và giữa các Use Case với nhau.



Hình 40: Ví dụ thể hiện Actor User sử dụng Use Case Login

**+ Quan hệ Generalization:**

Generalization

Hình 41: Ký hiệu quan hệ Generalization

**Generalization** được sử dụng để thể hiện quan hệ kế thừa giữa các Actor hoặc giữa các Use Case với nhau.



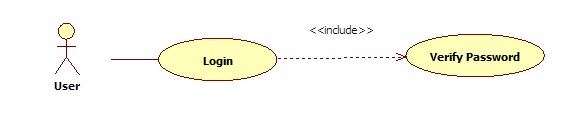
Hình 42: Ví dụ Actor User thừa kế toàn bộ quyền của Actor Guest

**+ Quan hệ Include:**



Hình 43: Ký hiệu quan hệ Include

**Include** là quan hệ giữa các Use Case với nhau, nó mô tả việc một Use Case lớn được chia ra thành các Use Case nhỏ để dễ cài đặt (module hóa) hoặc thể hiện sự dùng lại.



Hình 44: Ví dụ về quan hệ Include giữa các Use Case

Use Case “Verify Password” có thể gộp chung vào Use Case Login nhưng ở đây chúng ta tách ra để cho các Use Case khác sử dụng hoặc để module hóa cho dễ hiểu, dễ cài đặt.

**+ Quan hệ Extend**



Hình 45: Ký hiệu quan hệ Extend

**Extend** dùng để mô tả quan hệ giữa 2 Use Case. Quan hệ Extend được sử dụng khi có một Use Case được tạo ra để bổ sung chức năng cho một Use Case có sẵn và được sử dụng trong một điều kiện nhất định nào đó.

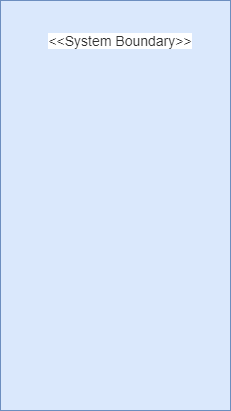


Hình 46: Ví dụ về quan hệ Extend giữa các Use Case

Trong ví dụ trên “Open Account” là Use Case cơ sở để cho khách hàng mở tài khoản. Tuy nhiên, có thêm một điều kiện là nếu khách hàng là công ty thì có thể thêm người sở hữu lên tài khoản này. **Add Account Holder** là chức năng mở rộng của Use Case “Open Account” cho trường hợp cụ thể nếu Actor là Công ty nên quan hệ của nó là quan hệ Extend.

#### **4.2.4 System Boundary**

System Boundary được sử dụng để xác định phạm vi của hệ thống mà chúng ta đang thiết kế. Các đối tượng nằm ngoài hệ thống này có tương tác với hệ thống được xem là các Actor.



Hình 47: Vùng System Boundary

System Boundary sẽ giúp chúng ta dễ hiểu hơn khi chia hệ thống lớn thành các hệ thống con để phân tích, thiết kế.

### **4.3 Đặc tả Use Case**

Để hiểu rõ hơn hệ thống chúng ta cần phải đặc tả các Use Case.

**Có 2 cách để đặc tả Use Case.**

**Cách 1:** Viết đặc tả cho các Use Case theo mẫu sau:

Tên Use Case // Account Details

Mã số Use Case // UCSEC35

Mô tả tóm tắt // Hiển thị thông tin chi tiết của Account

Các bước thực hiện // Liệt kê các bước thực hiện

Điều kiện thoát // Khi người dùng kích nút Close

Yêu cầu đặc biệt // Ghi rõ nếu có

Yêu cầu trước khi thực hiện // Phải đăng nhập

Điều kiện sau khi thực hiện // Ghi rõ những điều kiện nếu có sau khi thực hiện Use Case này

**Cách 2:** Sử dụng các bản vẽ để đặc tả.

Chúng ta có thể dùng các bản vẽ như **Activity Diagram**, **Sequence Diagram** để đặc tả Use case.

### **4.4 Mục đích sử dụng Use Case Diagram**

Use Case Diagram có một vai trò đặc biệt quan trọng trong quá trình phân tích, thiết kế và phát triển hệ thống. Dưới đây liệt kê một số ứng dụng tiêu biểu của Use Case Diagram.

- Phân tích và hiểu hệ thống

- Thiết kế hệ thống.

- Nắm bắt các yêu cầu của hệ thống.

- Thúc đẩy triển khai và tạo các trường hợp thử nghiệm

- Làm cơ sở cho việc phát triển, kiểm tra các bản vẽ như Class Diagram, Activity Diagram, Sequence Diagram, Component Diagram.

- Làm cơ sở để giao tiếp với khách hàng, các nhà đầu tư.

- Giúp cho việc kiểm thử chức năng, kiểm thử chấp nhận.

### **4.5 Các bước xây dựng Use Case Diagram**

**Bước 1:** Tìm các Actor

Trả lời các câu hỏi sau để xác định Actor cho hệ thống:

- Ai sử dụng hệ thống này?

- Hệ thống nào tương tác với hệ thống này?

**Bước 2:** Tìm các Actor

Trả lời câu hỏi các Actor sử dụng chức năng gì trong hệ thống? Chúng ta sẽ xác định được các Use Case cần thiết cho hệ thống.

**Bước 3:** Xác định các quan hệ

Phân tích và các định các quan loại hệ giữa các Actor và Use Case, giữa các Actor với nhau, giữa các Use Case với nhau, sau đó nối chúng lại chúng ta sẽ được bản vẽ Use Case.

### **4.6 Áp dụng Use Case Diagram vào extension AutoCheck và CheckAuto**

**Phân tích extensions:**

Với extensionAutoCheck, người dùng là các client (sinh viên) sẽ thực hiện nhập thông tin mã số sinh viên của bản thân vào ô nhập liệu và yêu cầu gửi thông qua button request. Server sẽ tiếp nhận thông tin và lưu lại vào database. Bên phía người sử dụng extension CheckAuto là các admin (Giảng viên) sẽ gội dữ liệu từ server về, kiểm tra với danh sách điểm danh hiện tại và thực hiện tự động điền số tiết vắng vào danh sách trong mỗi ô tương ứng

Extension AutoCheck và CheckAuto xây dựng với mục đích sử dụng nhằm giúp các admin lấy được danh sách do các client nhập dữ liệu điểm danh trên AutoCheck được chuyển đến server thông qua CheckAuto.

* Client khi dùng extension AutoCheck, bắt buộc phải nhập ID (MSSV).
* Các client trong ứng dụng được quản lý thông tin theo từng ID, mỗi client có một ID riêng khác nhau.

Khi admin muốn lấy dữ liệu điểm danh từ server thì chỉ cần nhấn vào button trên extension CheckAuto.

**Bước 1:** Tìm các Actor

Người sử dụng hệ thống: Client và Admin

**Bước 2:** Xác định các Use Case

Actor Client sử dụng chức năng auto check: tự động điểm danh khi nhập vào -> Use Case AutoCheck.

Actor Admin sử dụng chức năng check auto: tự động điền dữ liệu điểm danh khi nhấn vào nút Load -> Use Case CheckAuto.

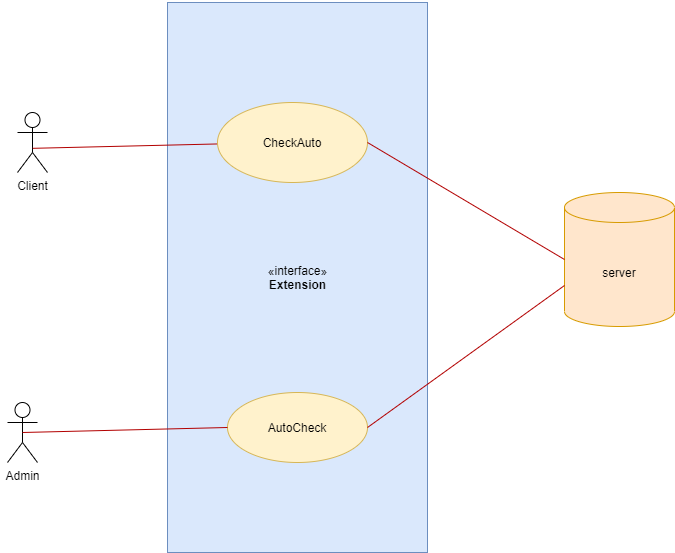
**Bước 3:** Xác định các quan hệ

Quan hệ giữa Client và Use Case AutoCheck: khi Client nhập dữ liệu vào và nhấn nút thì AutoCheck sẽ load dữ liệu lên server và lưu trữ trên đó.

Quan hệ giữa Admin và Use Case CheckAuto: khi Admin nhấn nút Load ở CheckAuto, CheckAuto sẽ load dữ liệu từ server đã lưu trữ trên đó và trả kết quả về cho admin.

Quan hệ giữa các Use Case: hai Use Case điều tương tác với Server. AutoCheck thì load dữ liệu lên server còn CheckAuto thì truy cập server và lấy dữ liệu để trả về.

Thông qua các bước trên chúng ta sẽ được bản vẽ Use Case.



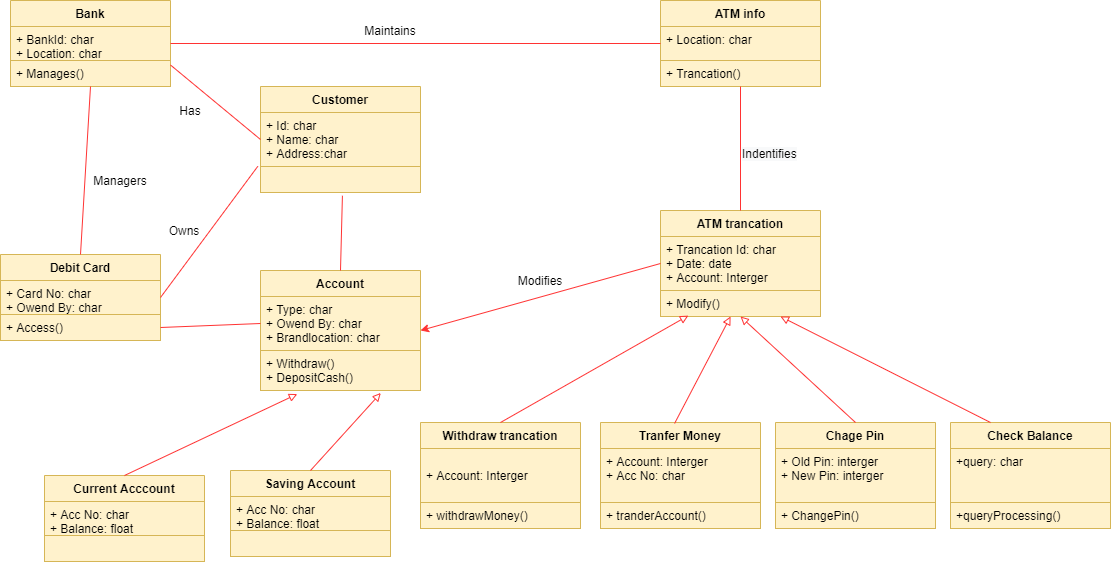
Hình 48: Use Case diagram extension autucheck & checkauto

## **5. Class Diagram - Bản vẽ về Class (lớp)**

### **5.1 Khái niệm**

Class Diagram là một trong những bản vẽ quan trọng nhất của thiết kế phần mềm, nó cho thấy cấu trúc và quan hệ giữa các thành phần tạo nên phần mềm. Trong quá trình xây dựng Class Diagram phải quyết định rất nhiều yếu tố về thiết kế nên nó là bản vẽ khó xây dựng nhất. Bản vẽ này sẽ cho thấy cấu trúc tĩnh của phần mềm.

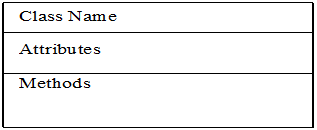
### **5.2 Các thành phần trong bản vẽ Class**



Hình 49: Ví dụ về Class Diagram của ATM

**Classes (Các lớp)**

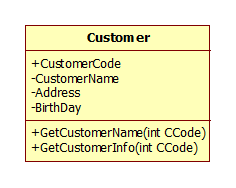
Class là thành phần chính của bản vẽ Class Diagram. Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống.



Hình 50: Ký hiệu về Class

**Trong đó:**

* Class Name: là tên của lớp.
* Attributes (thuộc tính): mô tả tính chất của các đối tượng. Ví dụ như khách hàng có Mã khách hàng, Tên khách hàng, Địa chỉ, Ngày sinh v.v…
* Method (Phương thức): chỉ các hành động mà đối tượng này có thể thực hiện trong hệ thống. Nó thể hiện hành vi của các đối tượng do lớp này tạo ra.



Hình 51: Ví dụ về một Class

### **5.3 Relationship (Quan hệ)**

Relationship thể hiện mối quan hệ giữa các Class với nhau. Trong UML 2.0 có các quan hệ thường sử dụng như sau:

- Association

- Aggregation

- Composition

- Generalization

+ **Quan hệ Association**

Association là quan hệ giữa hai lớp với nhau, thể hiện chúng có liên quan với nhau. Association thể hiện qua các quan hệ như “has: có”, “Own: sở hữu” v.v…

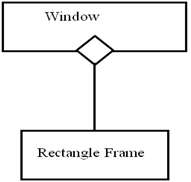
Association 1

Hình 52: Ví dụ về Association

Ví dụ quan hệ trên thể hiện Khách hàng nắm giữ Tài khoản và Tài khoản được sở hữu bởi Khách hàng.

+ **Quan hệ Aggregation**

Aggregation là một loại của quan hệ Association nhưng mạnh hơn. Nó có thể cùng thời gian sống (cùng sinh ra hoặc cùng chết đi)

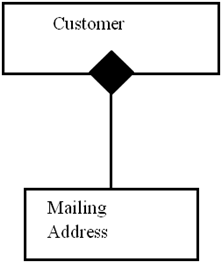


Hình 53: Ví dụ về Aggregation

Ví dụ quan hệ trên thể hiện lớp Window (cửa sổ) được lắp trên Khung cửa hình chữ nhật. Nó có thể cùng sinh ra cùng lúc.

+ **Quan hệ** **Composition**

Composition là một loại mạnh hơn của Aggregation thể hiện quan hệ class này là một phần của class kia nên dẫn đến cùng tạo ra hoặc cùng chết đi.

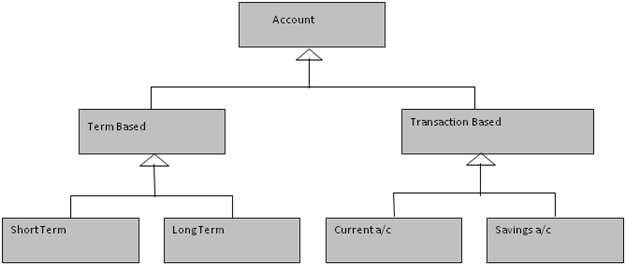


Hình 54: Ví dụ về Composition

Ví dụ trên class Mailing Address là một phần của class Customer nên chỉ khi nào có đối tượng Customer thì mới phát sinh đối tượng Mailing Address.

+ **Quan hệ** **Generalization**

Generalization là quan hệ thừa kế được sử dụng rộng rãi trong lập trình hướng đối tượng.



Hình 55: Ví dụ về Generalization

Các lớp ở cuối cùng như Short Term, Long Term, Current a/c, Savings a/c gọi là các lớp cụ thể (concrete Class). Chúng có thể tạo ra đối tượng và các đối tượng này thừa kế toàn bộ các thuộc tính, phương thức của các lớp trên.

Các lớp trên như Account, Term Based, Transaction Based là những lớp trừu tượng (Abstract Class), những lớp này không tạo ra đối tượng.

### **5.4 Đặc tả Class**

Nhìn vào Class Diagram có thể thấy cấu trúc của hệ thống gồm những lớp nào nhưng để cài đặt thì phải đặc tả chi tiết hơn nữa.

**Trong đó, cần mô tả:**

- Các thuộc tính: Tên, kiểu dữ liệu, kích thước,…

- Các phương thức:

+ Tên.

+ Mô tả.

+ Tham số đầu vào: Tên, kiểu dữ liệu, kích thước.

+ Kết quả đầu ra: Tên, kiểu dữ liệu, kích thước.

+ Luồng xử lý.

+ Điều kiện bắt đầu.

+ Điều kiện kết thúc.

### **5.5 Mục đích sử dụng bản vẽ Class**

Có thể tóm tắt một số ứng dụng của bản vẽ Class Diagram như sau:

- Hiểu cấu trúc của hệ thống

- Thiết kế hệ thống

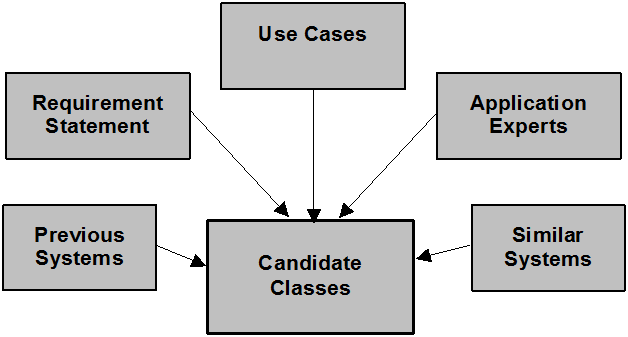
- Sử dụng để phân tích chi tiết các chức năng (Sequence Diagram, State Diagram v.v…)

- Sử dụng để cài đặt (coding)

### **5.6 Cách xây dựng bản vẽ Class**

**Bước 1:** Tìm các Classes dự kiến

Entity Classes (các lớp thực thể) là các thực thể có thật và hoạt động trong hệ thống, bạn dựa vào các nguồn sau để xác định chúng.



Hình 56: Các nguồn thông tin có thể tìm Class dự kiến

- Requirement statement: Các yêu cầu. Chúng ta phân tích các danh từ trong các yêu cầu để tìm ra các thực thể.

- Use Cases: Phân tích các Use Case sẽ cung cấp thêm các Classes dự kiến.

- Previous và Similar System: Có thể sẽ cung cấp thêm cho bạn các lớp dự kiến.

- Application Experts: Các chuyên gia ứng dụng cũng có thể giúp bạn.

**Bước 2:** Tìm các thuộc tính và phương thức cho lớp

- Tìm thuộc tính: phân tích thông tin từ các form mẫu có sẵn, bạn sẽ tìm ra thuộc tính cho các đối tượng của lớp.

- Tìm phương thức: phương thức là các hoạt động mà các đối tượng của lớp này có thể thực hiện.

**Bước 3:** Xây dựng các quan hệ giữa các lớp và phát hiện các lớp phát sinh

- Phân tích các quan hệ giữa các lớp và định nghĩa các lớp phát sinh do các quan hệ sinh ra. Chúng ta phân tích các thực thể ở trên và nhận thấy.

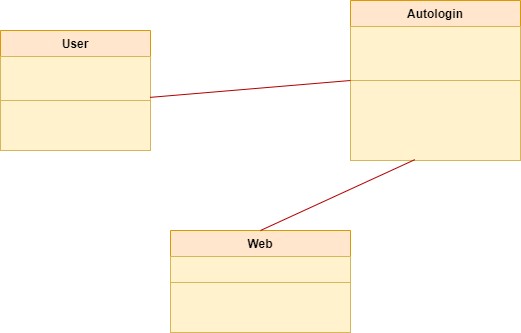
### **5.7 Áp dụng Class Diagram vào extension Auto Login**

**Bước 1:** Tìm các Class dự kiến

Xem xét Use Case Diagram của hệ thống:

Phân tích Use Case “**Auto Login**” chúng ta xác định thực thể autologin và có quan hệ với class User và class Web.

Tạm thời vẽ và xác định quan hệ sơ bộ chúng ta có bản vẽ Class dự kiến như sau:



Hình 57: Bản vẽ Class dự kiến

**Bước 2**: Xác định thuộc tính và quan hệ cho các lớp

- User: có các thuộc tính MSSV và Password

- Autologin: có các thuộc tính MSSV, Password và List Account

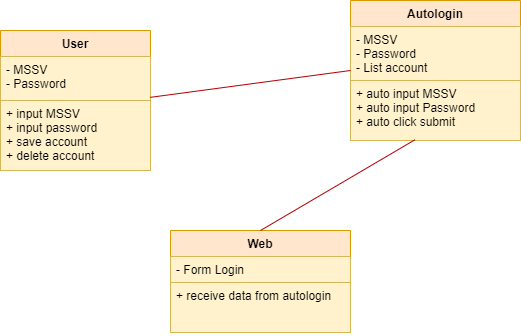
- Web: có thuộc tính Form Login



Hình 58: Xác định thuộc tính và quan hệ cho các lớp

**Bước 3**: Bổ sung phương thức cho các lớp:

Phương thức là các hành động mà đối tượng sinh ra từ lớp đó có thể thực hiện trong hệ thống.



Hình 59: Bổ sung phương thức cho các lớp

### **5.8 Áp dụng Class Diagram vào Extension AutoCheck và CheckAuto**

**Phân tích extensions:**

Với extensionAutoCheck, người dùng là các client (sinh viên) sẽ thực hiện nhập thông tin mã số sinh viên của bản thân vào ô nhập liệu và yêu cầu gửi thông qua button request. Server sẽ tiếp nhận thông tin và lưu lại vào database. Bên phía người sử dụng extension CheckAuto là các admin (Giảng viên) sẽ gội dữ liệu từ server về, kiểm tra với danh sách điểm danh hiện tại và thực hiện tự động điền số tiết vắng vào danh sách trong mỗi ô tương ứng

Extension AutoCheck và CheckAuto xây dựng với mục đích sử dụng nhằm giúp các admin lấy được danh sách do các client nhập dữ liệu điểm danh trên AutoCheck được chuyển đến server thông qua CheckAuto.

* Client khi dùng extension AutoCheck, bắt buộc phải nhập ID (MSSV).
* Các client trong ứng dụng được quản lý thông tin theo từng ID, mỗi client có một ID riêng khác nhau.
* Khi admin muốn lấy dữ liệu điểm danh từ server thì chỉ cần nhấn vào button trên extension CheckAuto.

**Bước 1: Xác định các class cần có và bố trí các thành phần lên class tương ứng.**

Dựa vào mục đích của hệ thống và kiến trúc hệ thống trên, ta xác định các class triển khai:

**Class 1:** Client – là người dùng thông thường, là người nhập vào extension AutoCheck.

**Class 2:** Extension AutoCheck.

**Class 3:** Admin – là người nắm quyền, là người có thể lấy dữ liệu từ server thông qua extension CheckAuto.

**Class 4:** Extension CheckAuto, lấy thông tin từ server và trả dữ liệu về cho admin.

**Server:** Database thì sẽ được đặt trên một Server.

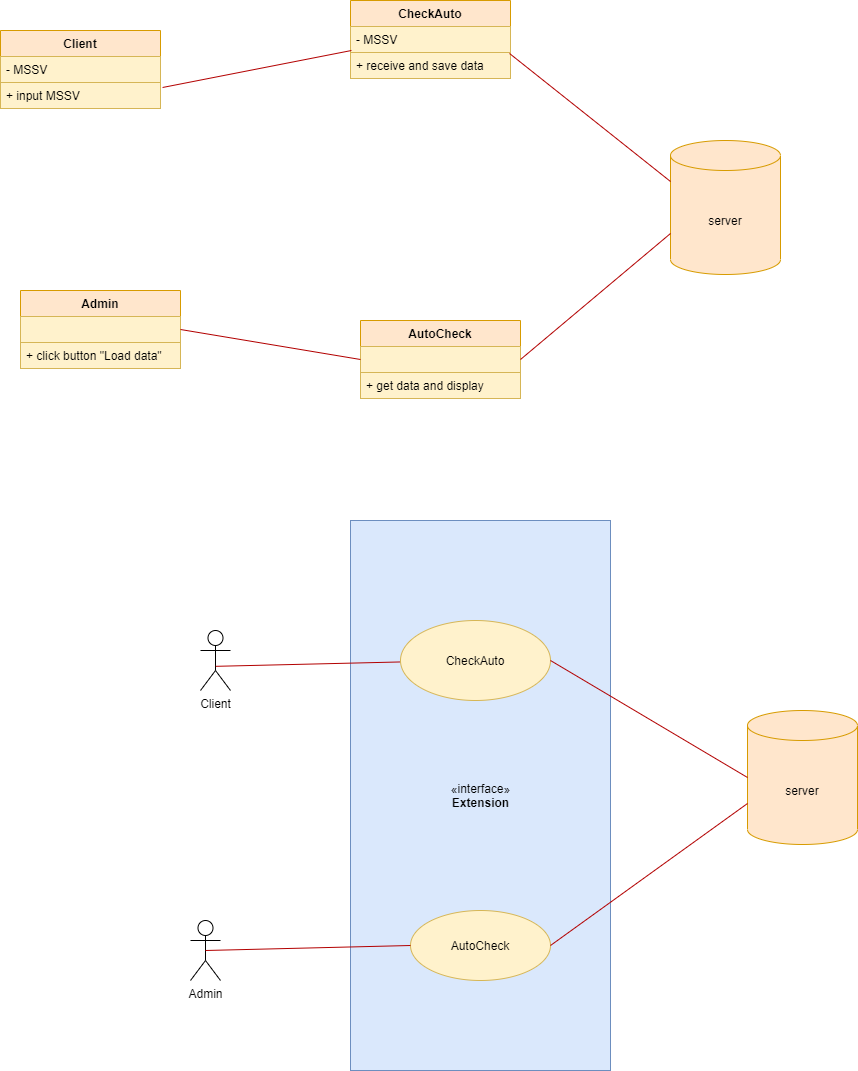
**Bước 2: Xác định mối quan hệ giữa các class**

Từ phần xác định class ở bước 1, ta tiến hành xác định mối quan hệ giữa chúng:

* Ta thấy **class 1** với **client** sẽ yêu cầu truy cập đến class 2 là Extension AutoCheck để có thể nhập thông tin.
* **Class 2** với extension AutoCheck sẽ tiếp nhận thông tin và truyền về server.
* Khi **class 3** với **admin** sẽ gửi một yêu cầu trả dữ liệu về danh sách thông qua class 4 extension **CheckAuto**.
* **Class 4** sẽ tiếp nhận yêu cầu và truy cập đến server lấy dữ liệu cần thiết và trả về cho admin.

Từ đó ta phân tích các thành phần cần có trong các class để hoàn thành sơ đồ:

Xem xét các thành phần trên với nhau để hoàn tất chức năng, chúng ta sẽ xác định các quan hệ của chúng. Biểu diễn lên bản vẽ chúng ta sẽ có Class Diagram như sau:



Hình 60: Class diagram Extension AutoCheck và CheckAuto

## **6. Package Diagram - Sơ đồ gói**

### **6.1 Khái niệm**

Sơ đồ gói, một loại sơ đồ cấu trúc, cho thấy sự sắp xếp và tổ chức của các phần tử mô hình trong dự án quy mô từ trung bình đến lớn. Sơ đồ gói có thể hiển thị cả cấu trúc và sự phụ thuộc giữa các hệ thống con hoặc mô-đun, hiển thị các quan điểm khác nhau của một hệ thống, ví dụ, như ứng dụng nhiều lớp (hay còn gọi là nhiều tầng) - mô hình ứng dụng nhiều lớp.

### **6.2 Mục đích của Sơ đồ gói**

Sơ đồ gói được sử dụng để cấu trúc các phần tử hệ thống cấp cao. Các gói được sử dụng để tổ chức hệ thống lớn chứa sơ đồ, tài liệu và các tài liệu phân phối chính khác.

Sơ đồ gói có thể được sử dụng để đơn giản hóa các sơ đồ lớp phức tạp, nó có thể nhóm các lớp thành các gói.

Một gói là một tập hợp các phần tử UML có liên quan về mặt logic.

Các gói được mô tả như các thư mục tệp và có thể được sử dụng trên bất kỳ sơ đồ UML nào.

**Vậy. Khi nào thì vẽ Sơ đồ gói?**

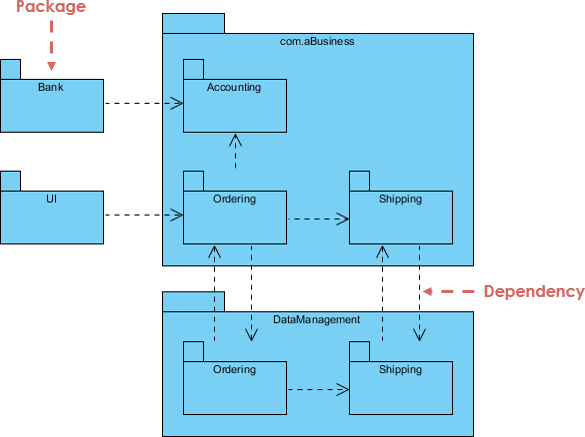
UML không coi các sơ đồ gói là một kỹ thuật riêng biệt, thường thì sẽ hữu ích khi kết hợp chúng bằng cách nhóm các phần tử mô hình khác lại với nhau thành các gói khác nhau trên cùng một sơ đồ.

Sơ đồ gói có thể hữu ích theo nhiều cách, chẳng hạn như:

* Khi muốn thấy tổng quan về một tập hợp lớn về các phần tử mô hình trên sơ đồ.
* Dùng để tổ chức, tạo một mô hình lớn.
* Dùng để nhóm các yếu tố liên quan.
* Dùng để tách không gian riêng biệt cho các tên phần tử.

### **6.3 Các thành phần trong bản vẽ Package Diagram**

Sơ đồ dưới đây là một mô hình hoạt động trong đó các lớp được nhóm thành các gói:



Hình 61: Ví dụ về Package Diagram

- Các gói xuất hiện dưới dạng hình chữ nhật với các tab nhỏ ở trên cùng.

- Tên gói nằm trên tab hoặc bên trong hình chữ nhật.

- Các mũi tên gạch đứt là biểu hiện cho sự phụ thuộc.

- Một gói A phụ thuộc vào gói B nếu có những thay đổi trong gói A có thể buộc những thay đổi trong gói B diễn ra tương ứng.

### **6.4 Các khái niệm cơ bản về Package Diagram**

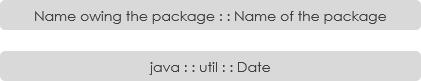
Sơ đồ gói tuân theo cấu trúc phân cấp của các gói lồng nhau.

Có một số ràng buộc trong khi sử dụng sơ đồ gói, chúng như sau:

- Tên gói không được giống nhau trong một hệ thống, tuy nhiên các lớp bên trong các gói khác nhau có thể có cùng tên.

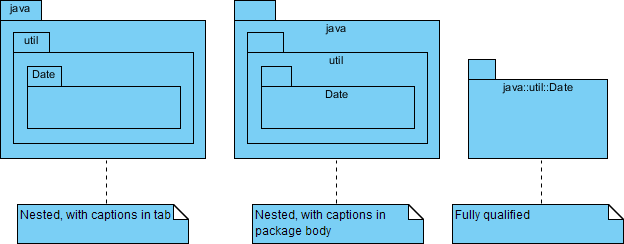
- Các gói có thể bao gồm toàn bộ sơ đồ.

- Tên đầy đủ điều kiện của một gói sẽ có cú pháp như sau:



Hình 62: Cú pháp của tên có đủ điều kiện của một gói

Các gói có thể được biểu thị bằng các ký hiệu với một số ví dụ được hiển thị bên dưới:



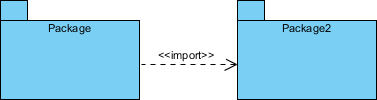
Hình 63: Các ký hiệu hiển thị các gói

**Ký hiệu phụ thuộc**

Có hai loại phụ liên quan đến sự phụ thuộc**: <<import>>** và **<<access>>.** Mặc dù có hai khuôn mẫu nhưng người dùng có thể sử dụng khuôn mẫu của riêng họ để biểu thị kiểu phụ thuộc giữa hai gói.

Ví dụ về sơ đồ gói - Nhập

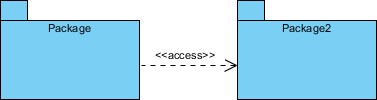
**<<import>>** - một gói nhập chức năng của gói khác.



Hình 64: Ví dụ về import package

Ví dụ về sơ đồ gói - Quyền truy cập

**<<access>>** - một gói yêu cầu trợ giúp từ các chức năng của gói khác.



Hình 65: Ví dụ về Access Package

### **6.5 Các bước vẽ Package diagram**

Khi muốn vẽ bất kỳ diagram nào ta luôn cần phải xác định các bước một cách cụ thể và sau đó mới bắt đầu phân tích cho bản vẽ. Sau đây là các bước để thực hành vẽ một Package diagram:

**Bước 1:** Xác định các gói có trong hệ thống.

**Bước 2:** Xác định các phụ thuộc có trong hệ thổng để đưa vào trong bản vẽ.

**Bước 3:** Áp dụng các gói và sự phụ thuộc giữa chúng để hoàn thành sơ đồ Package digram.

### **6.6 Ví dụ áp dụng**

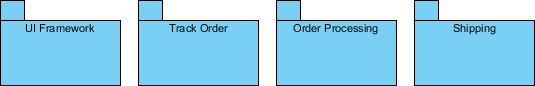
Ví dụ về sơ đồ gói - Hệ thống xử lý đơn hàng:

Trước tiên chúng ta nên xác định các gói trong hệ thống và sau đó liên kết chúng với nhau theo mối quan hệ:

**Bước 1:** Xác định các gói có trong hệ thống:

- Có một mô-đun theo dõi đơn đặt hàng, nó phải có liên hệ, giao tiếp đến với mô-đun khác để biết về chi tiết đơn hàng, chúng ta gọi nó là "Order Details".

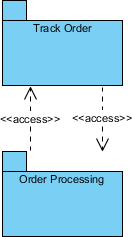
- Tiếp theo sau khi tìm nạp “Order Details”, nó phải biết về chi tiết vận chuyển, ta sẽ gọi đó là "Shipping".



Hình 66: Các package được xác định

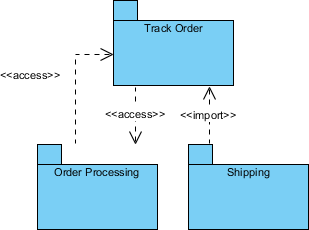
**Bước 2:** Xác định các phụ thuộc trong Hệ thống:

- Theo dõi đơn đặt hàng phải lấy chi tiết đơn hàng từ "Order Details" và đồng thời “Order Details" phải biết thông tin theo dõi do khách hàng cung cấp. Hai mô-đun đang truy cập lẫn nhau **<<access>>**.



Hình 67: Hai mô-đun đang truy cập lẫn nhau

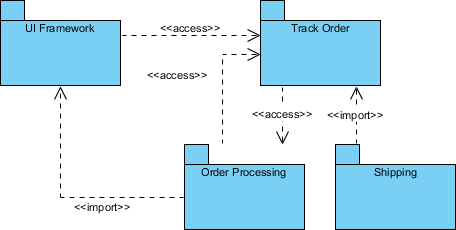
Để biết thông tin vận chuyển, "Shipping" có thể nhập "Track Order" (theo dõi đơn hàng) để điều hướng dễ dàng hơn.



Hình 68: Điều hướng thông tin vận chuyển

Cuối cùng, "Track Order" sẽ phụ thuộc vào “UI Framework” (Khung giao diện người dùng). Như vậy ta có sơ đồ sau:

**Bước 3:** Áp dụng các gói và sự phụ thuộc giữa chúng để hoàn thành sơ đồ Package digram.



Hình 69: Package xử lý đơn hàng

### **6.7 Áp dụng Package diagram vào extension Autologin**

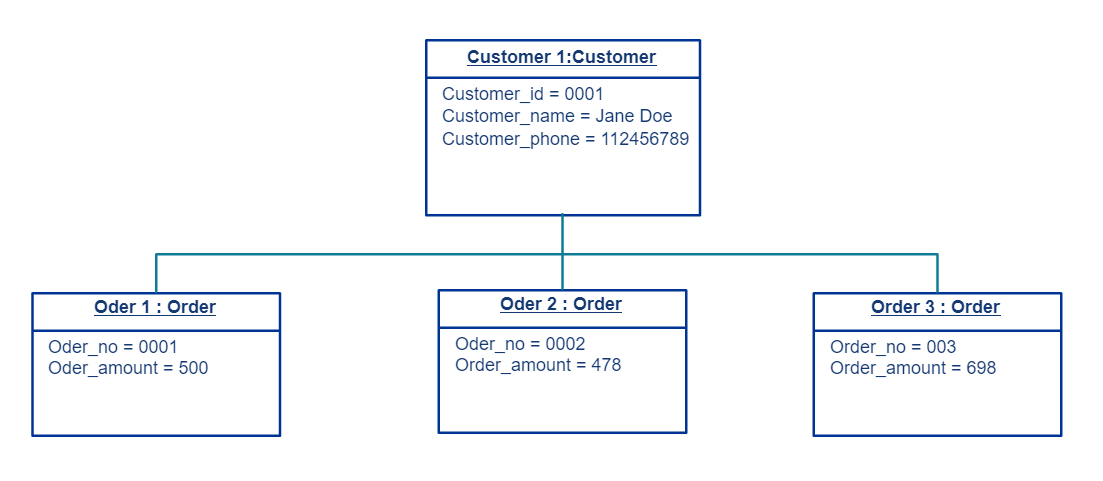
Trước tiên chúng ta nên xác định các gói trong hệ thống và sau đó liên kết chúng với nhau theo mối quan hệ:

**Bước 1:** Xác định các gói có trong hệ thống:

## **7. Object Diagram - Sơ đồ đối tượng**

### **7.1 Khái niệm**

Là một phiên bản khác của Class Diagram cũng được dùng các ký hiệu như Class Diagram. Sự khác biệt giữa hai lược đồ này là Object Diagram chỉ ra một loạt các đối tượng thay vì các class như Class Diagram.



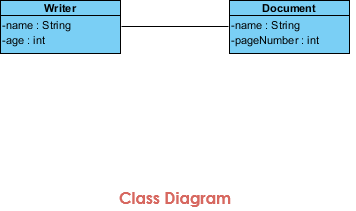
Hình 70: Object diagram

### **7.2 Mục đích sơ đồ đối tượng**

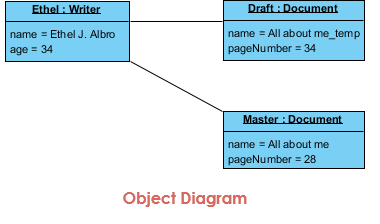
Sự khác biệt giữa sơ đồ lớp với sơ đồ đối tượng là: Sơ đồ lớp thì thể hiện một mô hình trừu tượng bao gồm các lớp và mối quan hệ giữa chúng. Tuy nhiên, sơ đồ đối tượng lại biểu diễn cho một cá thể tại một thời điểm cụ thể, bản chất cụ thể. Điều này có nghĩa sơ đồ đối tượng sẽ gần với hoạt động hệ thống thực tế hơn nhằm nắm bắt được kết cấu, cơ chế của một hệ thống tại một thời điểm cụ thể.

Object Diagram sẽ dùng chung các ký hiệu giống như Class Diagram nhưng sơ đồ Object Diagram cũng không thực sự quan trọng bằng Class Diagram. Chúng có thể dùng để minh họa một Class Diagram phức hợp chỉ ra được những thuộc tính và những mối quan hệ diễn ra như thế nào.

### **7.3 So sánh giữa Class Diagram và Object Diagram**



Hình 71: Class diagram



Hình 72: Object diagram

Dựa vào hai sơ đồ trên có thể thấy rằng việc dùng Object Diagram chỉ ra các đối tượng với tên được gạch chân và các thuộc tiính của đối tượng được đưa ra một cách minh bạch hơn so với Class Diagram. Nó sẽ bao quát hơn về kiểu dữ liệu và tên của class.

### **7.4 Áp dụng object diagram vào source code php-training**

**Phân tích source code:**

Source code php-training xây dựng với mục đích sử dụng nhằm quản lý danh sách thành viên users cho một phòng ban, công ty hoặc lớp học,… Nó bao gồm các hoạt động quản lý thông tin người dùng, thêm, xoá, sửa, đổi mật khẩu người dùng.

* Người dùng phải đăng ký tài khoản đề được quản trị viên quản lý thông tin cá nhân cho các mục đích công việc, học tập. Thông tin người dùng bao gồm username, fullname, email, password.
* Các người dùng trong ứng dụng được quản lý thông tin theo từng ID, mỗi người dùng có một ID riêng khác nhau. Khi người dùng có thông tin cần thay đổi, các hành động xóa hoặc sửa tùy theo mục đích thay đổi sẽ được thực thi.
* Khi người dùng quên mật khẩu, người dùng phải nhập email hiện tại đang đăng ký và phải có xác thực thông qua email trên phiên làm việc hiện tại thì mới được phép tiếp tục thay đổi mật khẩu mới.

Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

Như vậy ta sẽ tiến hành theo các bước sau:

**Bước 1:** Xác định số lượng đối cần có trong source code php-training, vì đây là một ứng dụng nhằm quản lý danh sách thành viên users nên sẽ có 2 đối tượng chính là **User** và **Admin.**

**Bước 2:** Xác định các thuộc tính có trong một đối tượng. Ứng dụng chỉ là quản lý danh sách thành viên ở mức cơ bản vậy nên sẽ có các thuộc tính chính cho đối tượng như sau:

Đối với User:

* Username
* Fullname
* Email

Đối với Admin:

* Username
* Fullname
* Email

**Bước 3: Áp dụng vào sơ đồ đối tượng.**

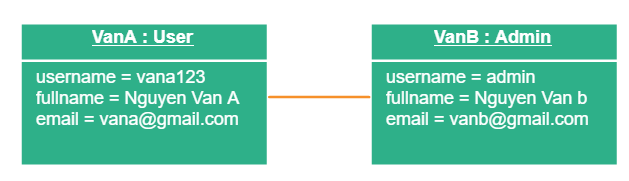
Với VanA thuộc đối tượng User với thuộc tính:

* username dùng để đăng nhập có giá trị là vana123.
* fullname dùng để phân biệt nhiều người dùng có giá trị là Nguyen Van A
* email cho việc liên hệ cũng như kích hoạt tài khoản có giá trị là vana@gmail.com

Với VanB thuộc đối tượng Admin với thuộc tính:

* username dùng để đăng nhập, quản lý các users khác có giá trị là admin.
* fullname dùng để phân biệt nhiều admin có giá trị là Nguyen Van B
* email cho việc liên hệ cũng như kích hoạt tài khoản cho admin với quyền quản trị có giá trị là vanb@gmail.com.

Từ 2 đối tượng User và Admin với các thuộc tính được đưa ra tiến hành áp dụng vào sơ đồ Object Diagram sẽ được kết quả như sau:

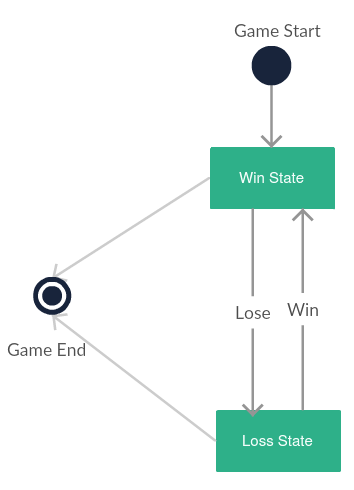


Hình 73: Object diagram áp dụng vào source code php-training

## **8. State Diagram - Sơ đồ trạng thái**

### **8.1 Khái niệm**

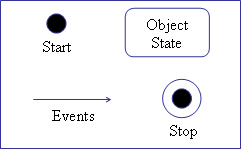
Đây là một sơ đồ trạng thái được dùng bổ sung cho **Class Diagram**. State Diagram chỉ ra tất cả các trạng thái mà đối tượng của Class có thể có và các event gây ra sự thay đổi của trạng thái. Event làm thay đổi của trạng thái xảy ra khi một đối tượng tự gửi thông tin dữ liệu đến cho State.



Hình 74: State diagram

### **8.2 Các thành phần trong State Diagram**

State Diagram gồm:



Hình 75: Ký hiệu chi tiết UML

**Start**: Trạng thái ban đầu được ký hiệu bằng **một điểm tròn**.

**Event**: Biến đổi giữa các trạng thái được ký hiệu bằng **mũi tên**.

**Object State**: Là đối tượng trong state diagram. Một số trạng thái ở giữa được ký hiệu bằng **hình chữ nhật**.

**Stop**: Một hoặc nhiều trạng thái kết thúc được ký hiệu bằng **một điểm tròn và vòng tròn bao ngoài**.

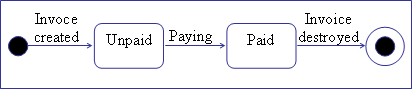
Những sự kiện gây nên sự biến đổi từ một trạng thái này sang một trạng thái khác.

### **8.3 Mục đích sơ đồ trạng thái**

State Diagram chuyển đổi trạng thái cũng có thể là một hành động liên quan, xác định điều gì phải được thực hiện trước khi chuyển đổi trạng thái (State Transition) được diễn ra. State Diagram không được dùng để vẽ cho tất cả các Class Diagram, mà chỉ riêng những Class Diagram có số lượng trạng thái được định nghĩa từ trước và các hành động của Class cũng bị ảnh hưởng và thay đổi sang các trạng thái khác nhau. State Diagram còn có thể vẽ được cho **hệ thống tổng thể**.

Ví dụ:

Sơ đồ thể hiện sự bắt đầu, kết thúc, sự kiện và trạng thái



Hình 76: Sơ đồ state diagram hoàn chỉnh

### **8.4 Áp dụng State Diagram vào source code php-training**

**Phân tích source code:**

Source code php-training xây dựng với mục đích sử dụng nhằm quản lý danh sách thành viên users cho một phòng ban, công ty hoặc lớp học,… Nó bao gồm các hoạt động quản lý thông tin người dùng, thêm, xoá, sửa, đổi mật khẩu người dùng.

* Người dùng phải đăng ký tài khoản đề được quản trị viên quản lý thông tin cá nhân cho các mục đích công việc, học tập. Thông tin người dùng bao gồm username, fullname, email, password.
* Các người dùng trong ứng dụng được quản lý thông tin theo từng ID, mỗi người dùng có một ID riêng khác nhau. Khi người dùng có thông tin cần thay đổi, các hành động xóa hoặc sửa tùy theo mục đích thay đổi sẽ được thực thi.
* Khi người dùng quên mật khẩu, người dùng phải nhập email hiện tại đang đăng ký và phải có xác thực thông qua email trên phiên làm việc hiện tại thì mới được phép tiếp tục thay đổi mật khẩu mới.

Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

Như vậy ta sẽ tiến hành theo các bước sau:

**Bước 1:** Xác định chức năng cần thực hiện, ở đây để có thể cho một ví dụ dễ hiểu nhất nên sẽ dùng chức năng **Login** và xác định nên dùng sơ đồ trạng thái chỉ ra cách vận hành của chức năng.

**Bước 2:** Xác định điểm bắt đầu, hành động đúng, hành động sai và điểm kết thúc:

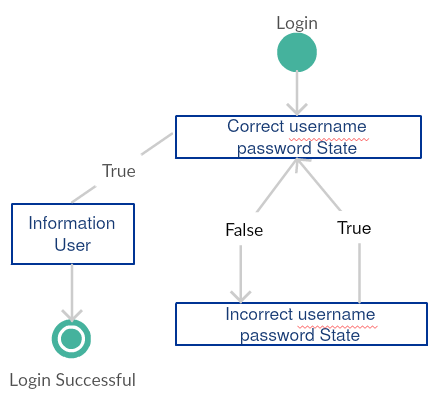
Trạng thái 1:

* Đúng username và password.
* Vào trang thông tin cá nhân.
* Kết thúc.

Trạng thái 2:

* Sai username và password.
* Đăng nhập lại.
* Đúng username và password.
* Vào trang thông tin cá nhân.
* Kết thúc.

**Bước 3: Áp dụng vào sơ đồ trạng thái.**



Hình 77: State diagram áp dụng vào source code php-training

**Trạng thái 1:**

Tiến hành thực hiện chức năng đăng nhập khi người dùng nhập đúng **username**, **password** sẽ trả về cho người dùng trang thông tin cá nhân của người dùng kèm thông báo đăng nhập thành công.

**Trạng thái 2:**

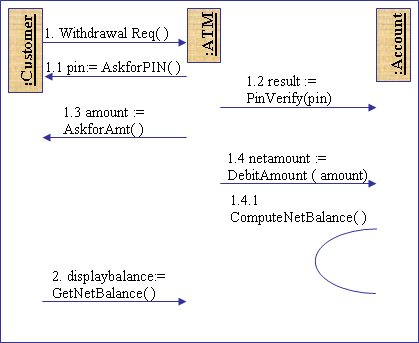
Tiến hành thực hiện chức năng đăng nhập khi người dùng nhập sai username, password sẽ yêu cầu người dùng nhập lại **username**, **password** đúng với **username**, **password** đăng ký, sau khi người dùng nhập đúng quay lại trường hợp 1 và trả về người dùng trang thông tin cá nhân kèm thông báo đăng nhập thành công.

## **9. Collaboration Diagram - Sơ đồ cộng sinh**

### **9.1 Khái niệm**

Collaboration Diagram hay còn được gọi là sơ đồ cộng tác nó tương tự như State Diagram nhưng lại tâm trung hết toàn bộ vào các sự kiện tương tác giữa các đối tượng với nhau.

Trong một sơ đồ cộng tác một hoặc nhiều đối tượng được biểu diễn bằng ký hiệu lớp. Thứ tự các event trong sơ đồ cộng tác được thể hiện bằng việc đánh số. Tuy nhiên việc đánh số có thể có phần phức tạp và khó hiểu hơn so với kỹ thuật dùng mũi tên trong sơ đồ tuần tự (State Diagram).



Hình 78: Collaboration diagram

### **9.2 Các thành phần trong Collaboration Diagram**

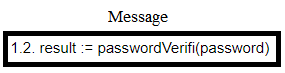
Component Diagram gồm:

**Object:** Là ký hiệu bởi một đối tượng được hiển thị **tên của đối tượng bằng dấu gạch chân**.



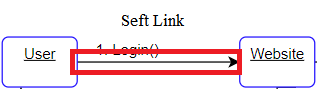
Hình 79: Object

**Message:** Được dùng để đưa ra thông điệp giao tiếp giữa các đối tượng cần truyền tải thông tin, dữ liệu với mong muốn hành động sẽ diễn ra sau đó được ký hiệu là **tên phía trên đường mũi tên**.



Hình 80: Message

**Seft Links(Link):** Là liên kết các đối tượng và dùng để chỉ ra các liên kết tương ứng và mỗi liên kết tương ứng với một liên kết khác trong sơ đồ lớp được ký hiệu là đường **mũi tên.**



Hình 81: Seft link

**Backend:** Được dùng nội bộ bên trong một Đối Tượng hoặc Lớp bất kỳ thường là một thủ tục riêng(private) được ký hiệu là **hình vòng cung vào trong**.



Hình 82: Backend

### **9.3 Mục đích sơ đồ cộng sinh**

Như hình minh họa trên:

1. Khách hàng yêu cầu rút tiền (Withdrawal Req()) được gọi từ class Customer.

1.1. Lúc này quy trình sẽ gọi hàm AskforPIN() được gọi ra từ class ATM. Thể hiện giá trị trả về là mã pin được khách hàng cung cấp, được viết dưới dạng pin:=AskforPIN().

1.2. Sau khi có được mã pin khách hàng gọi hàm PinVerify() từ class Customer và truyền mã pin được viết dưới dạng result:= PinVerify(pin).

1.3. Xác minh thành công mã pin từ khách hàng lúc này khách hàng gọi hàm AskforAmt() từ class Account yêu số tiền cần rút được viết dưới dạng amount:= AskforAmt().

1.4. Sau khi khách hàng yêu cầu số tiền rút thì lúc này hàm DebitAmount() được gọi ra từ class Customer truyền vào giá trị của khách hàng yêu cầu rút tiền được viết dưới dạng: netamount: = DebitAmount(amount).

1.4.1. Hình vòng cung trong class Account biểu hiện cho việc hàm ComputeNetBalance() được gọi và xử lý nội bộ thường sẽ là một thủ tục private của class Account.

2. Sau khi được xử lý nội bộ lúc này màn hình sẽ hiển thị số dư còn lại của khách hàng thông qua displaybalance:=GetNetBalance().

Sơ đồ trên giúp cho có cái nhìn tổng quan về các đối tượng tương tác và gọi các chức năng liên quan đến sau giữ vai trò cho ứng dụng web được hoạt động một cách ổn định nhất

### **9.4 Áp dụng Collaboration Diagram vào source code php-training**

**Phân tích source code:**

Source code php-training xây dựng với mục đích sử dụng nhằm quản lý danh sách thành viên users cho một phòng ban, công ty hoặc lớp học,… Nó bao gồm các hoạt động quản lý thông tin người dùng, thêm, xoá, sửa, đổi mật khẩu người dùng.

* Người dùng phải đăng ký tài khoản đề được quản trị viên quản lý thông tin cá nhân cho các mục đích công việc, học tập. Thông tin người dùng bao gồm username, fullname, email, password.
* Các người dùng trong ứng dụng được quản lý thông tin theo từng ID, mỗi người dùng có một ID riêng khác nhau. Khi người dùng có thông tin cần thay đổi, các hành động xóa hoặc sửa tùy theo mục đích thay đổi sẽ được thực thi.
* Khi người dùng quên mật khẩu, người dùng phải nhập email hiện tại đang đăng ký và phải có xác thực thông qua email trên phiên làm việc hiện tại thì mới được phép tiếp tục thay đổi mật khẩu mới.

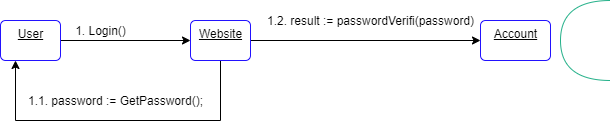
Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

**Bước 1:** Xác định chức năng cần thực hiện, ở đây để có thể cho một ví dụ dễ hiểu nhất nên sẽ dùng chức năng **Login** và dùng sơ đồ cộng sinh để nắm được quá trình tương tác giữa ba đối tượng: User, Website, Account.

**Bước 2:** Xác định đối tượng bắt đầu, đối tượng bắt đầu thực hiện hành động là user với mong muốn đăng nhập vào trang web.

* User muốn đăng nhập vào Website nên sẽ gọi hàm().
* Website lấy password của user thông qua hàm getPassword().
* Sau khi website lấy được password từ người dùng thì account sẽ gọi hàm passwordVerifi() truyền tham số là passwrod được website lấy từ user.
* Xử lý ngầm dưới server và tương tác với database.

**Bước 3:** Áp dụng và vẽ sơ đồ cộng sinh.

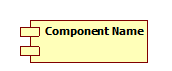


Hình 83: State diagram áp dụng vào source code php-training

## **10. Component Diagram - Sơ đồ thành phần**

### **10.1 khái niệm**

Component Diagram là một bản vẽ cho thấy được cấu trúc hệ thống theo từng thành phần của phần mềm. Component được đóng gói độc lập do là một thành phần của phần mềm, nó có thể được độc lập triển khai trên hệ thống và có khả năng tương tác với các thành phần khác khi thực hiện các chức năng của một hệ thống nào đó.



Hình 84: Ký hiệu của component

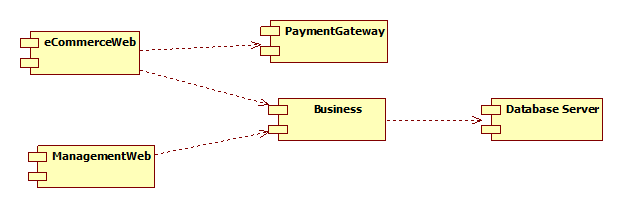
Ngoài ra còn có Component Dependency thể hiện quan hệ giữa các thành phần lại với nhau, các thành phần của phần mềm luôn luôn cần sử dụng một hoặc nhiều chức năng ở các thành phần khác trong hệ thống nên việc quan hệ Component Dependency được sử dụng thường xuyên.



Hình 85: Ký hiệu của component dependency

### **10.2 Mục đích sơ đồ Component Diagram**

Dựa vào bản vẽ Component Diagram cho hệ thống eCommerce như hình dưới đây tiến hành phân tích như sau:



Hình 86: Component diagram

Chia nhỏ hệ thống thành các SubSystem:

1. Website phục vụ cho đối tượng ngoài trông ty là customer và người dùng ẩn danh ra một component riêng dễ dàng cho việc triển khai và bảo mật. Component này được gọi là **eCommerceWeb**.

2. Phần website phục phụ cho nội bộ công ty cũng như phát triển **Component** **eCommerceWeb** được gọi là **ManagementWeb**.

3. **Component** **Business** được dùng, tương tác qua lại giữa web và database dùng để xử lý các hoạt động khác.

4. **Component PaymentGateway** dùng cho khách hàng thanh toán trực tuyến.

5. **Component Database Server** được tách riêng ra để tiện cho việc bảo mật cũng như nâng cấp database phục vụ phát triển ứng dụng.

Sau khi chia nhỏ các **Component** sẽ giúp cho việc quá trình thiết kế, phất triển, triển khai cũng như bảo mật ứng dụng một cách hiệu quả. Ngoài ra có một giải pháp triển khai các **Component** trên một server riêng để tăng khả năng bảo mật cũng như chịu tải cho hệ thống.

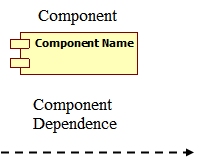
**Component** được dùng vào các công việc như:

Thể hiện rõ ràng minh bạch cấu trúc của một hệ thống.

Cung cấp đầu vào cho bản vẽ Deployment.

Hỗ trợ cho việc thiết kế các kiến trúc của một phần mềm.

### **10.3 Các thành phần trong Component Diagram**



Hình 87: Các kí hiệu trong Component diagram

Component diagram gồm:

* **Component**: Thành phần củ một phần mềm được đóng gói một cách độc lập, Component có thể được triển khai độc lập trên một hệ thống và đồng thời cũng có khả năng tương tác với các Component khác khi thực hiện chức năng trong cùng một hệ thống. Component được ký hiệu: **mảnh ghép hình chữ nhật**.
* **Component Dependence**: Thể hiện liên kết giữa các component với nhau và được dùng thường xuyên trong hệ thống, ký hiệu: **mũi tên nét đứt**.

### **10.4 Áp dụng Component Diagram vào extension AutoCheck và CheckAuto**

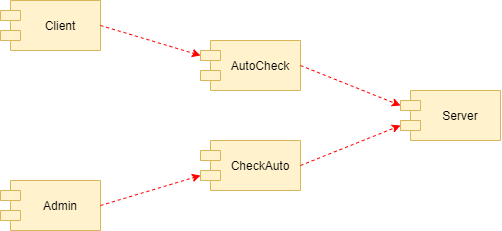
**Phân tích các hệ thống cho một Component**:

- Extension phục vụ cho hai đối tượng là client và admin -> Ta sẽ có được hai component là client và admin.

- Client và Admin sẽ tương tác với hai extension là AutoCheck và CheckAuto -> Ta có được hai component là AutoCheck và CheckAuto.

- Các extension tương tác qua lại với database dùng để xử lý các hoạt động.

Sau khi chia nhỏ các componenttasẽ được bảng vẽ:



Hình 88: Component diagram for extension

### **10.5 Áp dụng Component Diagram vào source code php-training**

**Phân tích source code:**

Source code php-training xây dựng với mục đích sử dụng nhằm quản lý danh sách thành viên users cho một phòng ban, công ty hoặc lớp học,… Nó bao gồm các hoạt động quản lý thông tin người dùng, thêm, xoá, sửa, đổi mật khẩu người dùng.

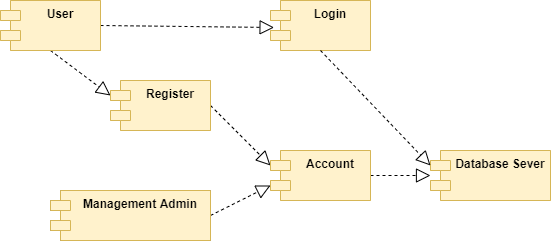
* Người dùng phải đăng ký tài khoản đề được quản trị viên quản lý thông tin cá nhân cho các mục đích công việc, học tập. Thông tin người dùng bao gồm username, fullname, email, password.
* Các người dùng trong ứng dụng được quản lý thông tin theo từng ID, mỗi người dùng có một ID riêng khác nhau. Khi người dùng có thông tin cần thay đổi, các hành động xóa hoặc sửa tùy theo mục đích thay đổi sẽ được thực thi.
* Khi người dùng quên mật khẩu, người dùng phải nhập email hiện tại đang đăng ký và phải có xác thực thông qua email trên phiên làm việc hiện tại thì mới được phép tiếp tục thay đổi mật khẩu mới.

Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

Như vậy ta tiến hành phân tích theo diagram:

* Source code phục vụ cho hai đối tượng là user và admin -> Ta sẽ có được hai component là **user** và **management admin**.
* Đối với user sẽ tương tác với một số chức năng như login, register nên ta có thêm hai component là **login** và **register**. Ngoài ra, đối với người dùng admin sẽ có tương tác với các account để quản lý nên ta có component **account**.
* Các chức năng đều tương tác đến database server nên ta có thêm một component là **database server**.

Như vậy ta sẽ có một bản vẽ đơn giản như sau:



Hình 89: Component diagram for source code php-training

## **11. Interaction Diagram - Sơ đồ tương tác**

### **11.1 Khái niệm**

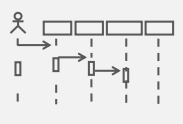
Interaction Diagram là sơ đồ được sử dụng để mô tả một số loại tương tác giữa các yếu tố khác nhau trong mô hình. Tương tác này là một phần của hành vi năng động của hệ thống.

Sơ đồ trình tự nhấn mạnh về chuỗi thời gian của các thông điệp và sơ đồ cộng tác nhấn mạnh vào việc tổ chức cấu trúc của các đối tượng.

Các loại sơ đồ tương tác:

− Sơ đồ tuần tự (Sequence diagram):

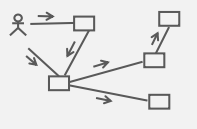
Một cách nhìn hướng về trình tự thời gian tương tác giữa các đối tượng.



Hình 90: Sơ đồ tuần tự

− Sơ đồ giao tiếp (Communication diagram):

Một cách nhìn thông điệp giữa các đối hướng về cấu trúc của quá trình truyền thông điệp giữa các đối tượng.



Hình 91: Sơ đồ giao tiếp

### **11.2 Mục đích Interaction Diagram**

Mục đích của sơ đồ tương tác là:

- Để nắm bắt hành vi năng động của một hệ thống.

- Mô tả luồng thông điệp trong hệ thống.

- Mô tả tổ chức cấu trúc của vật thể.

- Để mô tả sự tương tác giữa các đối tượng.

**11.3 Cách vẽ Interaction Diagram**

Những điều sau đây cần được xác định rõ ràng trước khi vẽ sơ đồ tương tác:

- Các đối tượng tham gia vào sự tương tác.

- Các luồng thông báo giữa các đối tượng.

- Trình tự trong đó các thông báo đang chảy.

- Tổ chức đối tượng.

**11.4 Sử dụng Interaction Diagram**

Sơ đồ tương tác có thể được sử dụng:

- Để mô hình luồng điều khiển theo trình tự thời gian.

- Mô hình luồng kiểm soát bởi các tổ chức cơ cấu.

- Đối với kỹ thuật chuyển tiếp.

- Đối với kỹ thuật đảo ngược.

## **12. Communication Diagram - Sơ đồ giao tiếp**

### **12.1 Khái niệm**

Sơ đồ giao tiếp (Communication Diagram) còn gọi là sơ đồ cộng tác, là một sơ đồ tương tác hiển thị thông tin tương tự như sơ đồ tuần tự. Nó tập trung vào việc mô tả các đối tượng tham gia tương tác như thế nào.

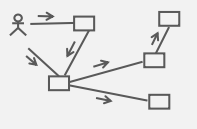
Nó thường được dùng để biểu diễn một kịch bản tương tác của một Use case. Ta có thể tạo nhiều Communication Diagram cho một Use case

**Sơ đồ chỉ ra:**

- Các đối tượng tham gia tương tác.

- Đường liên kết giữa các đối tượng.

- Thông điệp trao chuyển giữa các đối tượng.



Hình 92: Sơ đồ giao tiếp

### **12.2 Mục đích sơ đồ Communication Diagram**

- Thể hiện mối quan hệ rõ ràng trong quá trình tương tác.

- Thể hiện tốt hơn quá trình cộng tác.

- Thể hiện rõ hơn hiệu quả của quá trình tương tác trên từng đối tượng.

### **12.3 Các thành phần trong Communication Diagram**

Các ký hiệu được dùng trong sơ đồ giao tiếp tương tự nhưng ký hiệu của sơ đồ trình tự.

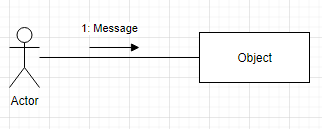
Sơ đồ giao tiếp có 4 thành phần chính:

- **Object**: Đối tượng tham gia quá trình tương tác giữa người dùng và hệ thống tạo nên ứng dụng.

- **Actor**: Tác nhân bên ngoài tương tác với hệ thống.

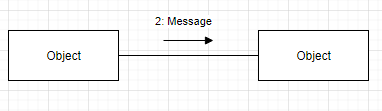
- **Instance Link**: Biểu diễn liên kết thông báo giữa 2 đối tượng.

+ Liên kết giữa actor và object:



Hình 93: Liên kết giữa actor và object

+ Liên kết giữa 2 object:



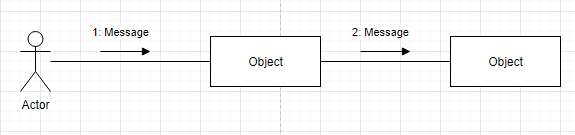
Hình 94: Liên kết giữa 2 object

- **Message**:

+ Thông điệp dùng cho mô tả tương tác giữa các đối tượng.

+ Thông điệp sẽ được gởi từ đối tượng này sang đối tượng khác.

+ Thông điệp có thể là 1 yêu cầu thực thi hệ thống, lời gọi hàm khởi tạo đối tượng, hủy đối tượng, cập nhật đối tượng,...



Hình 95: Mô tả tương tác giữa các đối tượng

**Các thuộc tính củ message:**

* **Action**
* **Control Flow**
* **Operation**
* **Arguments**
* **Return Value**
* **Predecessor list**
* **Condition**

Đối với riêng thuộc tính **Action** ta có:

* **Create**: đối tượng gửi thông điệp để gọi hàm khởi tạo đối tượng nhận thông điệp.
* **Destroy**: đối tượng gửi thông điệp để gọi hàm hủy đối tượng nhận thông điệp.
* **Self Destroy**: đối tượng gửi thông điệp sẽ bị hủy sau khi gởi thông điệp đến đối tượng nhận.

Đối với riêng thuộc tính **Control Flow** ta có:

* **Undefined**: không được định nghĩa
* **Asynchronous**: thông điệp không đồng bộ, đối tượng gửi thông điệp không cần đợi kết quả trả về từ đối tượng nhận, các thông điệp có thể thực hiện đồng thời.
* **Procedure Call**: thông điệp đồng bộ, đối tượng gửi thông điệp phải đợi kết quả trả về từ đối tượng nhận.
* **Return**: thông điệp return thường liên kết với thông điệp loại “Procedure Call”

### **12.4 Các bước vẽ Communication Diagram**

**Bước 1**: Xác định ngữ cảnh

Xác định chức năng cần thiết kế. Dựa vào Use Case Diagram để xác định xem chức năng nào cần thiết kế. (Ngữ cảnh có thể là use case hoặc 1 trường hợp cụ thể của use case).

**Bước 2**: Xác định actor, object tham gia vào lược đồ, xác định các bước thực hiện theo các hoạt động, chức năng khác nhau.

**Bước 3**: Xác định các liên kết association.

**Bước 4:** Sắp xếp các actor và object cho phù hợp.

**Bước 5**: Thêm các message.

**Bước 6**: Vẽ và kiểm tra lại Sequence Diagram.

### **12.5 So sánh Communication Diagram & Sequence Diagram**

**Biểu đồ trình tự - Sequence Diagram**

* Thể hiện rõ trình tự của quá trình tương tác
* Thể hiện tốt hơn luồng công việc
* Thể hiện tốt hơn quá trình mô tả các luồng sự kiện phức tạp trên phương diện thời gian thực

**Biểu đồ giao tiếp - Communication Diagram**

* Thể hiện mối quan hệ rõ ràng trong quá trình tương tác
* Thể hiện tốt hơn quá trình cộng tác
* Thể hiện rõ hơn hiệu quả của quá trình tương tác trên từng đối tượng.

### **12.6 Áp dụng Communication Diagram vào extension AutoCheck và CheckAuto**

**Phân tích extensions:**

Với extensionAutoCheck, người dùng là các client (sinh viên) sẽ thực hiện nhập thông tin mã số sinh viên của bản thân vào ô nhập liệu và yêu cầu gửi thông qua button request. Server sẽ tiếp nhận thông tin và lưu lại vào database. Bên phía người sử dụng extension CheckAuto là các admin (Giảng viên) sẽ gội dữ liệu từ server về, kiểm tra với danh sách điểm danh hiện tại và thực hiện tự động điền số tiết vắng vào danh sách trong mỗi ô tương ứng

Extension AutoCheck và CheckAuto xây dựng với mục đích sử dụng nhằm giúp các admin lấy được danh sách do các client nhập dữ liệu điểm danh trên AutoCheck được chuyển đến server thông qua CheckAuto.

* Client khi dùng extension AutoCheck, bắt buộc phải nhập ID (MSSV).
* Các client trong ứng dụng được quản lý thông tin theo từng ID, mỗi client có một ID riêng khác nhau.

Khi admin muốn lấy dữ liệu điểm danh từ server thì chỉ cần nhấn vào button trên extension CheckAuto

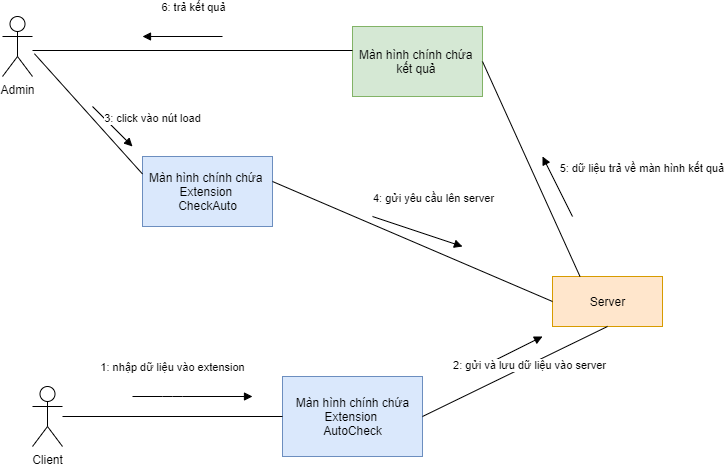
**Xác định các thành phần:**

**Actor**: gồm client và admin

**Object**: màn hình chính, màn hình chứa extension AutoCheck, màn hình chứa extension CheckAuto, Server.

**Các mũi tên liên kết**: đại diện cho liên kết thông báo giữa actor và object, admin và object.

**Message**: gồm 6 message.



Hình 96: Communication Diagram for extensions

### **12.7 Áp dụng Communication Diagram vào extension Autologin**

**Phân tích extensions:**

Với extensionAutologin, người dùng là các client sẽ thực hiện nhập thông tin bao gồm tài khoản và mật khẩu cho một trang web nào đó vào trong form của extension. Sau đó nhấn lưu, tài khoàn và mật khẩu này sẽ được lưu trên **Cache** của chrome thông qua **sync storage**. Khi cần dùng đến dữ liệu thì dữ liệu sẽ được lấy về thông qua extension.

Tiếp theo các client sẽ vào một trang web (ví dụ: el.tdc.edu.vn) nhấn vào Login.

Ở đây có hai trường hợp diễn ra:

- Trường hợp 1: Nếu client nhập dữ liệu vào extension và không nhấn vào nút tự động đăng nhập thì kết quả trả về đó là tài khoản và mật khẩu sẽ được điền vào form login nhưng không thực hiện việc tự nhấn submit dăng nhập và khi này người dùng phải tụ thực hiện việc nhấn submit.

- Trường hợp 2: Nếu client nhập dữ liệu vào extension và nhấn vào nút tự động đăng nhập thì kết quả trả về đó là tài khoản và mật khẩu sẽ được điền vào form login và sẽ tự động thực hiện việc nhấn submit đăng nhập.

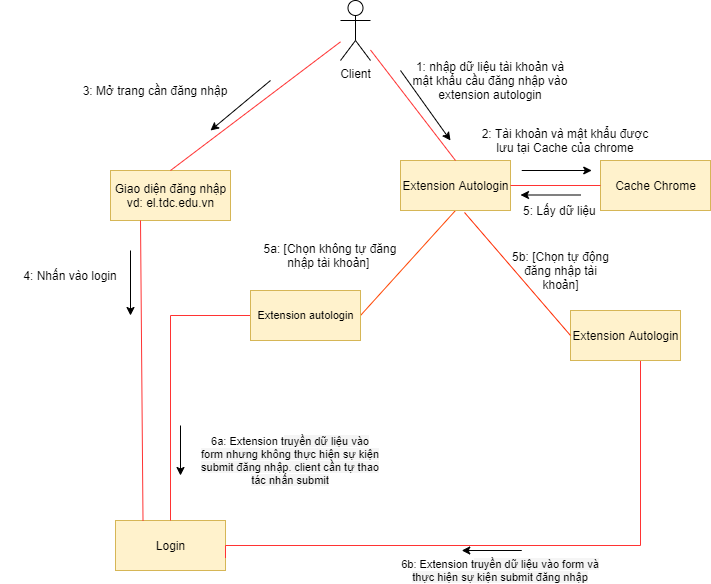
**Xác định các thành phần:**

**Actor**: gồm Client.

**Object**: Giao diện trang web, extension autologin, cache chrome, Login

**Các liên kết**: liên kết giữa client với object và giữa các object với nhau.

**Message**: gồm 6 message chính.



Hình 97: Communication Diagram for extension Autologin

# **III. EXTENSION DIỂM DANH ONLINE**

## **1. Giới thiệu**

Extension: Là một trang web được xây dựng và liên kết vào chrome dưới dạng tiện ích mở rộng, nó có thể là một extension hữu ích cho công việc, giải trí hoặc cũng là một extension cực kỳ nguy hiểm được các hacker tạo ra với mục đích xấu, tuy nhiên khi nhìn về mặt tích cực của extension sẽ giúp cho việc trải nghiệm công việc hoặc giải trí một cách thoải mái và tiện dụng nhất có thể. Ví dụ: một extension có thể được dùng để làm thay đổi thành phần bố cục của một trang web chẳn hạn như xóa quảng cáo

Extension autologin và checkauto: Extension này tạo ra giúp cho việc điểm danh của giáo viên và sinh viên trong một lớp học.

## **2. Hướng dẫn code và cách cài đặt sử dụng các extension**

### **2.1 Hướng dẫn code extension Autocheck**

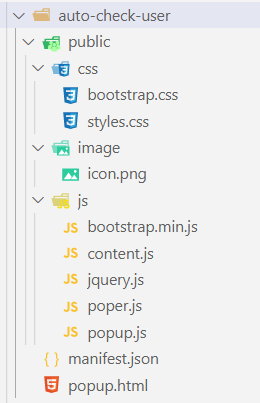
Extension **Autocheck** giúp giáo viên lấy được dữ liệu điểm danh từ sinh viên thông qua server.

**Các file cần có trong extension autocheck:**

- public (css, images, js).

- mainfest.json.

- popup.html.



**Hình 98: Cây thư mục extension autocheck**

**2.1.1** Thiết kế giao diện cho extension trong file **mainfest.json**

{

    "manifest\_version": 2,

    "name": "Autocheck presence student",

    "description": "Tự động điểm danh Sinh Viên",

    "version": "1.0",

 "permissions": ["http://localhost/\*", "https://localhost/\*", "storage", "tabs"],

    "content\_scripts": [{

        "matches": [

            "<all\_urls>"

        ],

        "js": ["public/js/jquery.js" , "public/js/content.js"]

    }],

    "browser\_action": {

        "default\_title": "Autocheck presence student",

        "default\_popup": "popup.html",

        "default\_icon": "public/image/icon.png"

    }

}

**2.1.2** Thiết kế giao diện cho extension trong file **popup.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link rel="stylesheet" type="text/css" href="./public/css/bootstrap.css">

    <link rel="stylesheet" type="text/css" href="./public/css/styles.css">

    <script src="./public/js/jquery.js"></script>

</head>

<body>

    <header>

        <h3>AutoCheck Student</h3>

    </header>

    <div class="container" style="text-align: center;">

        <button id="load" class="btn btn-outline-success" type="submit">Load data</button>

        <button id="clean" class="btn btn-outline-danger" type="submit">Clean</button>

        <button id="reload" class="btn btn-outline-info" type="submit">Reload</button>

    </div>

    <div id="responsecontainer" hidden>

    </div>

    <script src="./public/js/popup.js"></script>

    <script src="./public/js/poper.js"></script>

    <script src="./public/js/bootstrap.min.js"></script>

</body>

</html>

**2.1.3** Viết các sự kiện cho file **popup.js**

$(document).ready(function () {

  /\*the first load database to extension \*/

  $.ajax({

    //create an ajax request to display.php

    type: "GET",

    url: "http://localhost/php-training/for-extension/auto-check-presence.php",

    dataType: "html", //expect html to be returned

    success: function (response) {

      $("#responsecontainer").html(response);

      //alert(response);

    },

  });

  /\*-------------------------------------------\*/

  /\* re-initialize storage mssv every time it is loaded\*/

  chrome.storage.sync.get(function () {

    user = { mssv: [] };

    chrome.storage.sync.set({

      list: user,

    });

  });

  chrome.storage.sync.get(["list"], function (result) {

    console.log(result["list"]["mssv"]);

  });

  /\*-------------------------------------------\*/

  /\* initialize storage for the list presence \*/

  chrome.storage.sync.get(function () {

    user = { mssv\_user: [] };

    chrome.storage.sync.set({

      list\_online: user,

    });

  });

  chrome.storage.sync.get(["list\_online"], function (result) {

    console.log(result["list\_online"]["mssv\_user"]);

  });

  /\*-------------------------------------------\*/

  /\* event handling click load button \*/

  $("#load").click(function () {

    // alert($('#save').html());

    if ($("#load").html() == "Load data") {

      $("#load").html("Load Done");

      $("#load").attr("disabled", "true");

    }

  });

  $("#load").click(function () {

    list\_user\_presenced();

    $("#clean").show();

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "start" });

    });

  });

  /\*-------------------------------------------\*/

  /\* event handling click clean button \*/

  $("#clean").hide();

  $("#clean").click(function () {

    $("#load").hide();

    $("#reload").show();

    $("#responsecontainer").html();

    /\* load database to extension \*/

    $.ajax({

      //create an ajax request to display.php

      type: "GET",

      url: "http://localhost/php-training/for-extension/auto-check-presence.php",

      dataType: "html", //expect html to be returned

      success: function (response) {

        $("#responsecontainer").html(response);

        //alert(response);

      },

    });

    /\*-------------------------------------------\*/

    chrome.storage.sync.get(function () {

      user = { mssv: [] };

      chrome.storage.sync.set({

        list: user,

      });

    });

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "clean" });

    });

  });

  /\*-------------------------------------------\*/

  /\* event handling click reload button \*/

  $("#reload").hide();

  $("#reload").click(function () {

    /\* load database to extension \*/

    $.ajax({

      //create an ajax request to display.php

      type: "GET",

      url: "http://localhost/php-training/for-extension/auto-check-presence.php",

      dataType: "html", //expect html to be returned

      success: function (response) {

        $("#responsecontainer").html(response);

        //alert(response);

      },

    });

    /\*-------------------------------------------\*/

    chrome.storage.sync.get(function () {

      user = { mssv: [] };

      chrome.storage.sync.set({

        list: user,

      });

    });

    list\_user\_presenced();

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "reload" });

    });

  });

});

function list\_user\_presenced() {

  // console.log($('#mssv\_user2').val());

  var get\_input\_mssv;

  var total\_user\_mssv = $("#total\_user\_mssv").val();

  console.log(total\_user\_mssv);

  //show data storage : https://chrome.google.com/webstore/detail/storage-area-explorer/ocfjjjjhkpapocigimmppepjgfdecjkb

  //save list mssv from server file to chrome storage

  chrome.storage.sync.get(["list"], function (result) {

    for (var i = 1; i <= total\_user\_mssv; i++) {

      if ($("#inlineCheckbox" + i).is(":checked") == true) {

        get\_input\_mssv = $("#mssv\_user" + i).val();

        var get\_list = result["list"];

        get\_list["mssv"].push(get\_input\_mssv);

        chrome.storage.sync.set({

          list: get\_list,

        });

      }

    }

    console.log(result["list"]);

  });

}

**2.1.4** Viết các chức năng cho các nút tại file **content.js**

$(document).ready(function () {

    chrome.runtime.onMessage.addListener(function (

        request,

    ) {

        if (request.message === "start") {

            load\_presence();

            chrome.storage.sync.get(function () {

                user = { mssv: [] };

                chrome.storage.sync.set({

                    list: user,

                });

            });

        }

        if (request.message === "clean") {

            console.log('clean');

            clean\_status();

        }

        if (request.message === "reload") {

            reload();

            chrome.storage.sync.get(function () {

                user = { mssv: [] };

                chrome.storage.sync.set({

                    list: user,

                });

            });

        }

    });

});

function load\_presence() {

    var get\_mssv\_server;

    chrome.storage.sync.get(["list\_online"], function (result) {

        //Attribute Starts With Selector: https://api.jquery.com/attribute-starts-with-selector/

       var total\_user\_server = $('[id^="grvListStudents\_lblStudentID\_"]').length;

        /\*save list mssv from "main server online presence" to chrome storage\*/

        for (var i = 0; i < total\_user\_server; i++) {

            if ($('#grvListStudents\_lblStudentID\_' + i)) {

                get\_mssv\_server = $('#grvListStudents\_lblStudentID\_' + i).html();

                var get\_list = result['list\_online'];

                get\_list['mssv\_user'].push(get\_mssv\_server);

                chrome.storage.sync.set({

                    "list\_online": get\_list

                });

            } else;

        }

        var list\_user\_server = result['list\_online'];

        console.log("so user tren server: " + total\_user\_server + ", " + list\_user\_server['mssv\_user'].length);

        /\* Test mssv users post up and auto-fill tabs \*/

        chrome.storage.sync.get(["list"], function (result\_sync) {

            var list\_user\_presenced = result\_sync['list'];

            console.log(list\_user\_server['mssv\_user']);

            console.log(list\_user\_presenced['mssv']);

            for (let i = 0; i < list\_user\_server['mssv\_user'].length; i++) {

                if (jQuery.inArray(list\_user\_server['mssv\_user'][i], list\_user\_presenced['mssv']) !== -1) {

                    $('#grvListStudents\_txtVangCP\_' + i).val(0);

                    $('#inputnumber' + i).val("");

                } else {

                    $('#inputnumber' + i).val(5);

                    $('#grvListStudents\_txtVangKP\_' + i).val(5);

                }

            }

        });

    });

}

function clean\_status() {

    var total\_user\_server = $('[id^="grvListStudents\_lblStudentID\_"]').length;

    for (var i = 0; i < total\_user\_server; i++) {

        $('#grvListStudents\_txtVangCP\_' + i).val("");

        $('#inputnumber' + i).val("");

        $('#grvListStudents\_txtVangKP\_' + i).val("");

    }

}

function reload() {

    chrome.storage.sync.get(["list\_online"], function (result) {

        var list\_user\_server = result['list\_online'];

        console.log("so user tren server: " + list\_user\_server['mssv\_user'].length);

        /\* Test mssv users post up and auto-fill tabs \*/

        chrome.storage.sync.get(["list"], function (result\_sync) {

            var list\_user\_presenced = result\_sync['list'];

            console.log(list\_user\_presenced['mssv']);

            console.log(list\_user\_server['mssv\_user']);

            for (let i = 0; i < list\_user\_server['mssv\_user'].length; i++) {

                if (jQuery.inArray(list\_user\_server['mssv\_user'][i], list\_user\_presenced['mssv']) !== -1) {

                    $('#grvListStudents\_txtVangCP\_' + i).val(0);

                } else {

                    $('#inputnumber' + i).val(5);

                    $('#grvListStudents\_txtVangKP\_' + i).val(5);

                }

            }

        });

    });

}

**2.1.5** Viết css cho extention tại file **style.css** theo đường dẫn **public/css/style.css**

header h3{

    text-align: center;

}

header #go-to-options{

    float: right;

}

.container{

  padding-bottom: 0.5em;

  width: 300px;

  border-radius: 5px;

}

.input-group-text{

    background: #fff;

}

#save{

    margin-top: 15px;

    width: 32.6%;

    border: 1px solid #ccc;

    color: #000;

}

#save:hover{

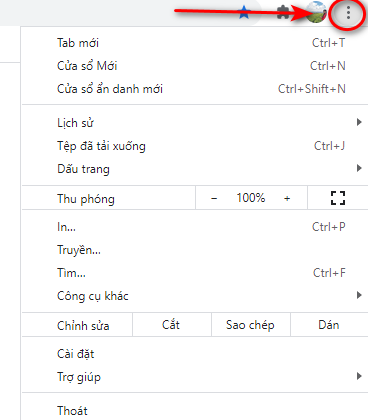
    border: 1px solid #fff;

}

Các file còn lại như: **bootstrap.min.css**, **bootstrap.min.js**, **poper.js**, **jquery.js** có thể tải từ trên mạng về.

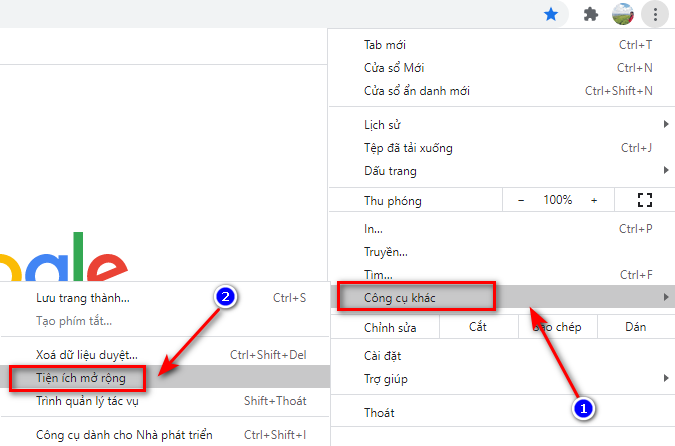
### **2.2 Hướng dẫn cài đặt extension Autocheck cho chrome**

Các bạn mở chrome và nhấn vào nút tùy chỉnh và điều khiển chrome bên góc phải màn hình:



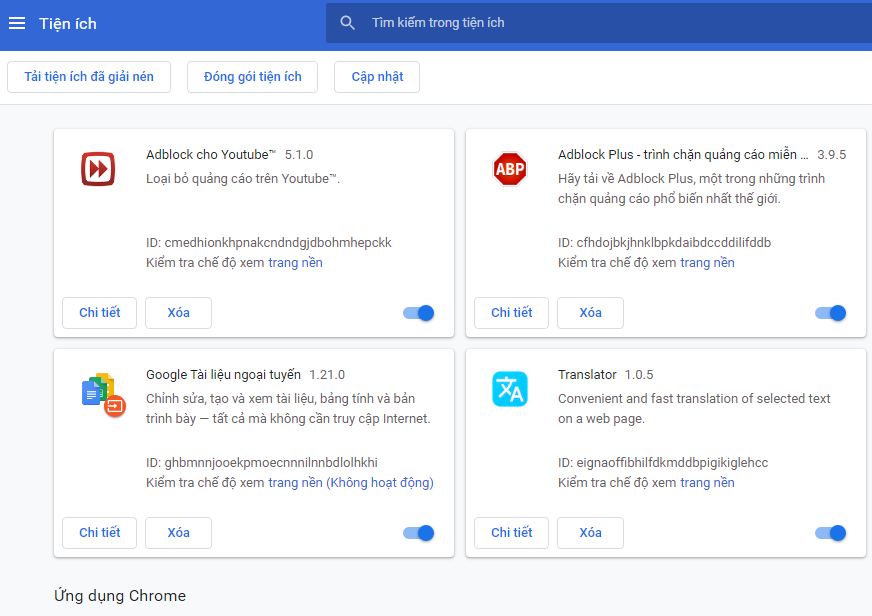
**Hình 99: Giao diện nút tùy chỉnh và điều khiển chrome**

Tiếp theo chọn “**Công cụ khác**” -> “**Tiện ích mở rộng**” được giao diện như sau:



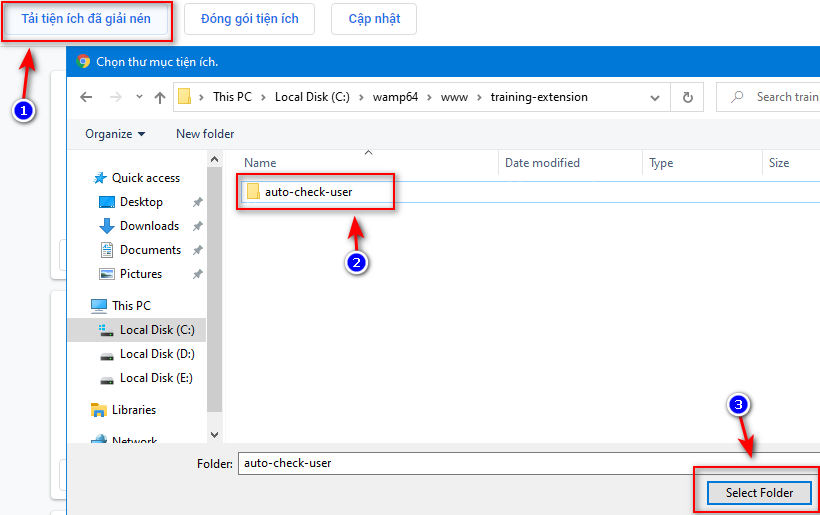
**Hình 100: Giao diện công cụ khác**

Sau khi nhấn vào tiện ích mở rộng bạn sẽ được giao diện như sau:



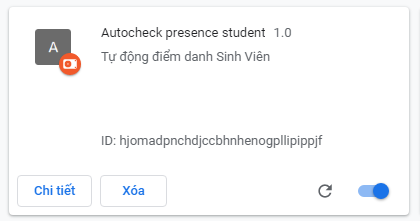
**Hình 101: Giao diện khi nhấn vào tiện ích mở rộng**

Tiếp theo nhấn vào mục: “**Tải tiện ích đã giải nén**” -> chọn folder “**auto-check-user**” -> chọn select



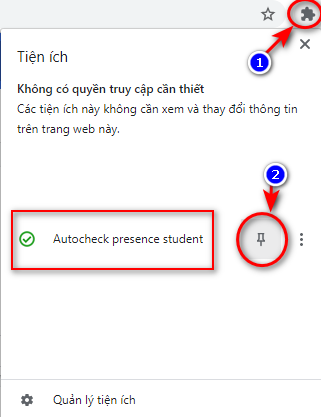
**Hình 102: Tải tiện ích đã giải nén**

Sau khi tải lên xong ta sẽ có một extension như hình:



**Hình 103: extension autocheck đã được cài**

Tiếp theo các bạn nhấn vào biểu tượng Tiện ích -> chọn đến extension vừa được cài -> nhấn vào biểu tượng nút ghim:



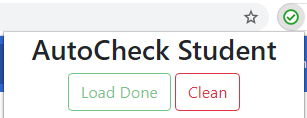
**Hình 104: Ghim extension vừa cài trong tiện ích**

Sau khi ghim extension sẽ hiển thị bên ngoài kế bên tiện ích. Nhằm mục đích cho người dùng dễ sử dụng:

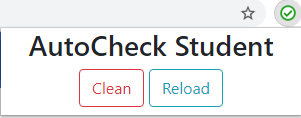


**Hình 105: Biểu tượng extension cạnh tiện ích**

Giao diện làm việc của extension autocheck:



**Hình 106: Giao diện extension autocheck chứa nút Load và Clean**



**Hình 107: Giao diện extension autocheck chứa nút Clean và Reload**

**Trong đó:**

+ Nút **Load Data**: lấy dữ liệu từ server trả vào danh sách.

+ Nút **Clean**: xóa dữ liệu đã điền vào danh sách.

+ Nút **Reload**: lặp lại hành động của chức năng **Load Data**.

### **2.3 Hướng dẫn code extension Checkauto**

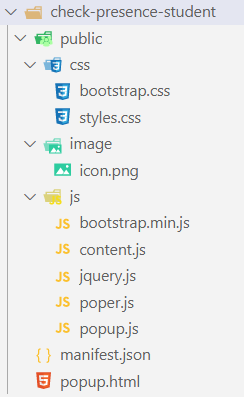
Extension checkauto giúp sinh viên điểm danh trực tiếp tại chổ ngồi khi đến lớp.

**Các file cần có trong extension autocheck:**

- public (css, images, js).

- mainfest.json.

- popup.html.



**Hình 108: Cây thư mục extension checkauto**

**2.3.1** Thiết kế giao diện cho extension trong file **mainfest.json**

{

    "manifest\_version": 2,

    "name": "Check presence student",

    "description": "Sinh Viên Điểm danh",

    "version": "1.0",

    "permissions": ["http://localhost/\*", "https://localhost/\*"],

    "content\_scripts": [{

        "matches": [

            "<all\_urls>"

        ],

        "js": ["public/js/jquery.js" , "public/js/content.js"]

    }],

    "browser\_action": {

        "default\_title": "Check presence student",

        "default\_popup": "popup.html",

        "default\_icon": "public/image/icon.png"

    }

}

**2.3.2** Thiết kế giao diện cho extension trong file **popup.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <link rel="stylesheet" type="text/css" href="./public/css/bootstrap.css">

    <link rel="stylesheet" type="text/css" href="./public/css/styles.css">

    <script src="./public/js/jquery.js"></script>

</head>

<body>

    <header>

        <h3>AutoCheck Student</h3>

    </header>

    <div class="container" style="text-align: center;">

        <button id="load" class="btn btn-outline-success" type="submit">Load data</button>

        <button id="clean" class="btn btn-outline-danger" type="submit">Clean</button>

        <button id="reload" class="btn btn-outline-info" type="submit">Reload</button>

    </div>

    <div id="responsecontainer" hidden>

    </div>

    <script src="./public/js/popup.js"></script>

    <script src="./public/js/poper.js"></script>

    <script src="./public/js/bootstrap.min.js"></script>

</body>

</html>

**2.3.3** Viết các sự kiện cho file **popup.js**

$(document).ready(function () {

  /\*the first load database to extension \*/

  $.ajax({

    //create an ajax request to display.php

    type: "GET",

    url: "http://localhost/php-training/for-extension/auto-check-presence.php",

    dataType: "html", //expect html to be returned

    success: function (response) {

      $("#responsecontainer").html(response);

      //alert(response);

    },

  });

  /\*-------------------------------------------\*/

  /\* re-initialize storage mssv every time it is loaded\*/

  chrome.storage.sync.get(function () {

    user = { mssv: [] };

    chrome.storage.sync.set({

      list: user,

    });

  });

  chrome.storage.sync.get(["list"], function (result) {

    console.log(result["list"]["mssv"]);

  });

  /\*-------------------------------------------\*/

  /\* initialize storage for the list presence \*/

  chrome.storage.sync.get(function () {

    user = { mssv\_user: [] };

    chrome.storage.sync.set({

      list\_online: user,

    });

  });

  chrome.storage.sync.get(["list\_online"], function (result) {

    console.log(result["list\_online"]["mssv\_user"]);

  });

  /\*-------------------------------------------\*/

  /\* event handling click load button \*/

  $("#load").click(function () {

    // alert($('#save').html());

    if ($("#load").html() == "Load data") {

      $("#load").html("Load Done");

      $("#load").attr("disabled", "true");

    }

  });

  $("#load").click(function () {

    list\_user\_presenced();

    $("#clean").show();

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "start" });

    });

  });

  /\*-------------------------------------------\*/

  /\* event handling click clean button \*/

  $("#clean").hide();

  $("#clean").click(function () {

    $("#load").hide();

    $("#reload").show();

    $("#responsecontainer").html();

    /\* load database to extension \*/

    $.ajax({

      //create an ajax request to display.php

      type: "GET",

      url: "http://localhost/php-training/for-extension/auto-check-presence.php",

      dataType: "html", //expect html to be returned

      success: function (response) {

        $("#responsecontainer").html(response);

        //alert(response);

      },

    });

    /\*-------------------------------------------\*/

    chrome.storage.sync.get(function () {

      user = { mssv: [] };

      chrome.storage.sync.set({

        list: user,

      });

    });

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "clean" });

    });

  });

  /\*-------------------------------------------\*/

  /\* event handling click reload button \*/

  $("#reload").hide();

  $("#reload").click(function () {

    /\* load database to extension \*/

    $.ajax({

      //create an ajax request to display.php

      type: "GET",

      url: "http://localhost/php-training/for-extension/auto-check-presence.php",

      dataType: "html", //expect html to be returned

      success: function (response) {

        $("#responsecontainer").html(response);

        //alert(response);

      },

    });

    /\*-------------------------------------------\*/

    chrome.storage.sync.get(function () {

      user = { mssv: [] };

      chrome.storage.sync.set({

        list: user,

      });

    });

    list\_user\_presenced();

    //Use the chrome.tabs API to interact with the browser's tab system

    chrome.tabs.query({ active: true, currentWindow: true }, function (tabs) {

      var activeTab = tabs[0];

      chrome.tabs.sendMessage(activeTab.id, { message: "reload" });

    });

  });

});

function list\_user\_presenced() {

  // console.log($('#mssv\_user2').val());

  var get\_input\_mssv;

  var total\_user\_mssv = $("#total\_user\_mssv").val();

  console.log(total\_user\_mssv);

  //show data storage : https://chrome.google.com/webstore/detail/storage-area-explorer/ocfjjjjhkpapocigimmppepjgfdecjkb

  //save list mssv from server file to chrome storage

  chrome.storage.sync.get(["list"], function (result) {

    for (var i = 1; i <= total\_user\_mssv; i++) {

      if ($("#inlineCheckbox" + i).is(":checked") == true) {

        get\_input\_mssv = $("#mssv\_user" + i).val();

        var get\_list = result["list"];

        get\_list["mssv"].push(get\_input\_mssv);

        chrome.storage.sync.set({

          list: get\_list,

        });

      }

    }

    console.log(result["list"]);

  });

}

**2.3.4** Viết các chức năng cho các nút tại file **content.js**

$(document).ready(function () {

    chrome.runtime.onMessage.addListener(function (

        request,

    ) {

        if (request.message === "start") {

            load\_presence();

            chrome.storage.sync.get(function () {

                user = { mssv: [] };

                chrome.storage.sync.set({

                    list: user,

                });

            });

        }

        if (request.message === "clean") {

            console.log('clean');

            clean\_status();

        }

        if (request.message === "reload") {

            reload();

            chrome.storage.sync.get(function () {

                user = { mssv: [] };

                chrome.storage.sync.set({

                    list: user,

                });

            });

        }

    });

});

function load\_presence() {

    var get\_mssv\_server;

    chrome.storage.sync.get(["list\_online"], function (result) {

        //Attribute Starts With Selector: https://api.jquery.com/attribute-starts-with-selector/

        var total\_user\_server = $('[id^="grvListStudents\_lblStudentID\_"]').length;

        /\*save list mssv from "main server online presence" to chrome storage\*/

        for (var i = 0; i < total\_user\_server; i++) {

            if ($('#grvListStudents\_lblStudentID\_' + i)) {

                get\_mssv\_server = $('#grvListStudents\_lblStudentID\_' + i).html();

                var get\_list = result['list\_online'];

                get\_list['mssv\_user'].push(get\_mssv\_server);

                chrome.storage.sync.set({

                    "list\_online": get\_list

                });

            } else;

        }

        var list\_user\_server = result['list\_online'];

        console.log("so user tren server: " + total\_user\_server + ", " + list\_user\_server['mssv\_user'].length);

        /\* Test mssv users post up and auto-fill tabs \*/

        chrome.storage.sync.get(["list"], function (result\_sync) {

            var list\_user\_presenced = result\_sync['list'];

            console.log(list\_user\_server['mssv\_user']);

            console.log(list\_user\_presenced['mssv']);

            for (let i = 0; i < list\_user\_server['mssv\_user'].length; i++) {

                if (jQuery.inArray(list\_user\_server['mssv\_user'][i], list\_user\_presenced['mssv']) !== -1) {

                    $('#grvListStudents\_txtVangCP\_' + i).val(0);

                    $('#inputnumber' + i).val("");

                } else {

                    $('#inputnumber' + i).val(5);

                    $('#grvListStudents\_txtVangKP\_' + i).val(5);

                }

            }

        });

    });

}

function clean\_status() {

    var total\_user\_server = $('[id^="grvListStudents\_lblStudentID\_"]').length;

    for (var i = 0; i < total\_user\_server; i++) {

        $('#grvListStudents\_txtVangCP\_' + i).val("");

        $('#inputnumber' + i).val("");

        $('#grvListStudents\_txtVangKP\_' + i).val("");

    }

}

function reload() {

    chrome.storage.sync.get(["list\_online"], function (result) {

        var list\_user\_server = result['list\_online'];

        console.log("so user tren server: " + list\_user\_server['mssv\_user'].length);

        /\* Test mssv users post up and auto-fill tabs \*/

        chrome.storage.sync.get(["list"], function (result\_sync) {

            var list\_user\_presenced = result\_sync['list'];

            console.log(list\_user\_presenced['mssv']);

            console.log(list\_user\_server['mssv\_user']);

            for (let i = 0; i < list\_user\_server['mssv\_user'].length; i++) {

                if (jQuery.inArray(list\_user\_server['mssv\_user'][i], list\_user\_presenced['mssv']) !== -1) {

                    $('#grvListStudents\_txtVangCP\_' + i).val(0);

                } else {

                    $('#inputnumber' + i).val(5);

                    $('#grvListStudents\_txtVangKP\_' + i).val(5);

                }

            }

        });

    });

}

**2.3.5** Viết css cho extention tại file **style.css** theo đường dẫn **public/css/style.css**

header h3{

    text-align: center;

}

header #go-to-options{

    float: right;

}

.container{

  padding-bottom: 0.5em;

  width: 300px;

  border-radius: 5px;

}

.input-group-text{

    background: #fff;

}

#save{

    margin-top: 15px;

    width: 32.6%;

    border: 1px solid #ccc;

    color: #000;

}

#save:hover{

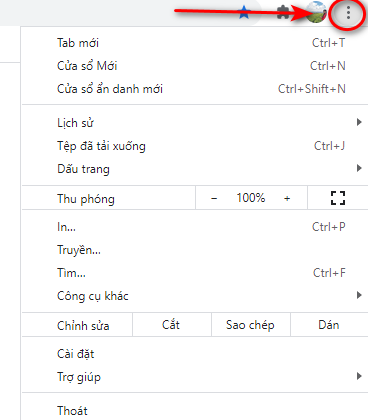
    border: 1px solid #fff;

}

Các file còn lại như: **bootstrap.min.css**, **bootstrap.min.js**, **poper.js**, **jquery.js** có thể tải từ trên mạng về.

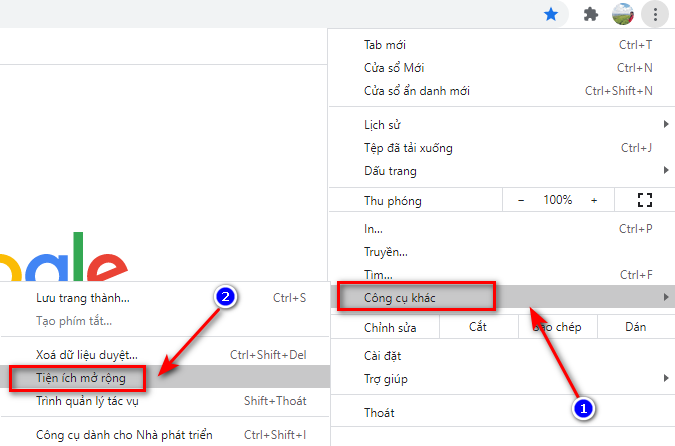
### **2.4. Hướng dẫn cài đặt extension Checkauto cho chrome**

Các bạn mở chrome và nhấn vào nút tùy chỉnh và điều khiển chrome bên góc phải màn hình:



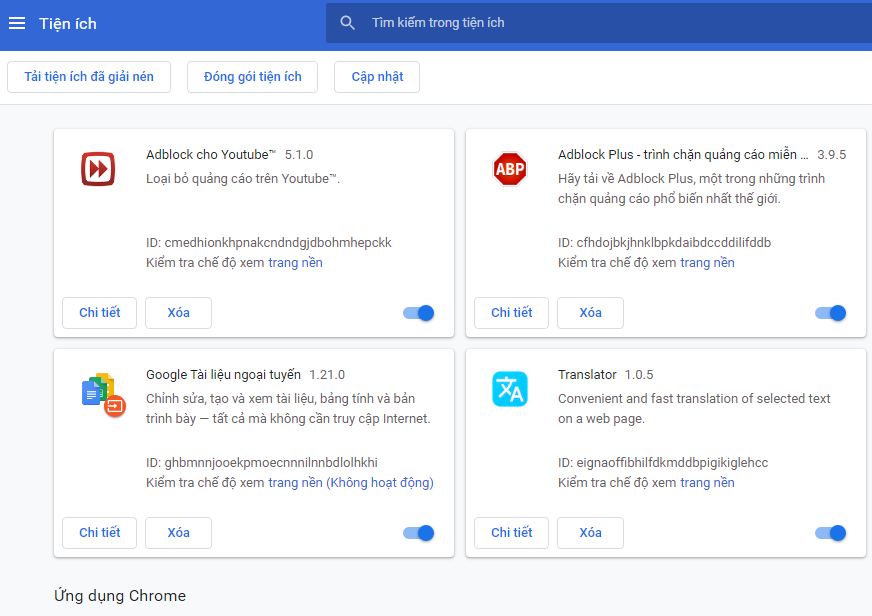
**Hình 109: Giao diện nút tùy chỉnh và điều khiển chrome**

Tiếp theo chọn công cụ khác -> tiện ích mở rộng được giao diện như sau:



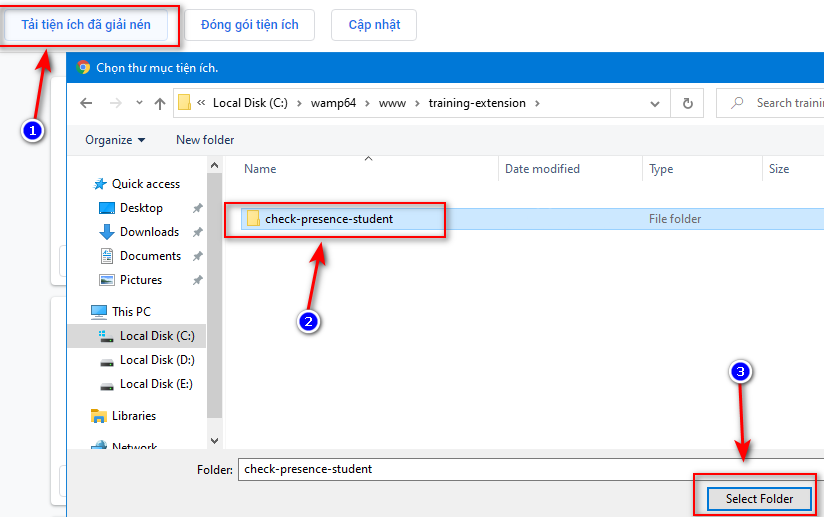
**Hình 110: Giao diện công cụ khác**

Sau khi nhấn vào tiện ích mở rộng bạn sẽ được giao diện như sau:



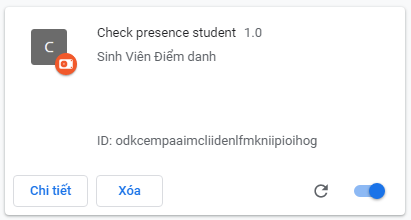
**Hình 14: Giao diện khi nhấn vào tiện ích mở rộng**

Tiếp theo nhấn vào mục: tải tiện ích đã giải nén -> chọn file check-presence-student -> chọn select:



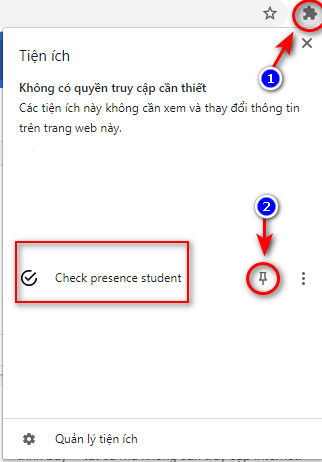
**Hình 111: Tải tiện ích đã giải nén**

Sau khi tải lên xong ta sẽ có một extension như hình:



**Hình 112: extension checkauto đã được cài**

Tiếp theo các bạn nhấn vào biểu tượng : Tiện ích -> chọn đến extension vừa được cài -> nhấn vào biểu tượng nút ghim:



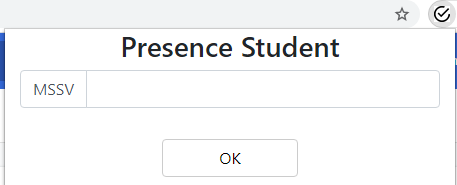
**Hình 113: Ghim extension vừa cài trong tiện ích**

Sau khi ghim extension sẽ hiển thị bên ngoài kế bên tiện ích. Nhằm mục đích cho người dùng dễ sữ dụng:



**Hình 114: Biểu tượng extension cạnh tiện ích**

Giao diện làm việc của extension autocheck:



**Hình 115: Giao diện extension checkauto**

**Trong đó:**

+ MSSV: nơi các bạn sinh viên nhận mssv của mình vào.

+ Nút OK: gửi dữ liệu lên server.

# **III. LARAVEL**

## **1. Hướng Dẫn Cài Laravel 8.x**

**1.1 Giới thiệu**

Laravel hiện nay là một trong những PHP framework mã nguồn mở và hoàn toàn miễn phí, tuy vậy cách cài đặt Laravel có phần hơi khó khăn với người dùng cơ bản, đặc biệt với những người mới vào học về lập trình PHP.

**1.2 Hướng dẫn cài đặt và tạo project laravel 8.x**

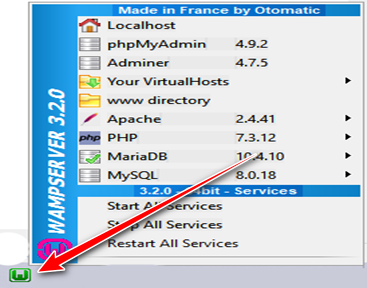
***1.2.1 Cài đặt Wamp Server***

Nếu các bạn chưa có Wampserver sẵn trong máy thì có thể vào link: <https://www.wampserver.com/en/download-wampserver/> để tải và cài đặt. Sau khi tải và cài đặt thành công vào click vào biểu tượng W (wampserver) để khởi động Wampserver.



**Hình 116: Biểu tượng Wamp Server**

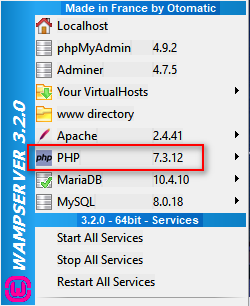
Sau khi click vào khởi động, bạn phải chờ ít phút. Trong lúc Wampserver chạy sẽ xuất hiện biểu tượng **W** ở phía dưới bên góc phải màn hình.



**Hình 117: Biểu tượng Wampserver khi khởi động**

Khi khởi động biểu tượng sẽ thay đổi màu từ Đỏ -> Cam ->Xanh, khi biểu tượng màu xanh như hình 8 là wamp đã khởi động thành công.

Ở đây các bạn chú ý phiên bản PHP các bạn phải sử dụng phiên bản >= 7.3.



**Hình 118: Phiên bản PHP 7.3.12**

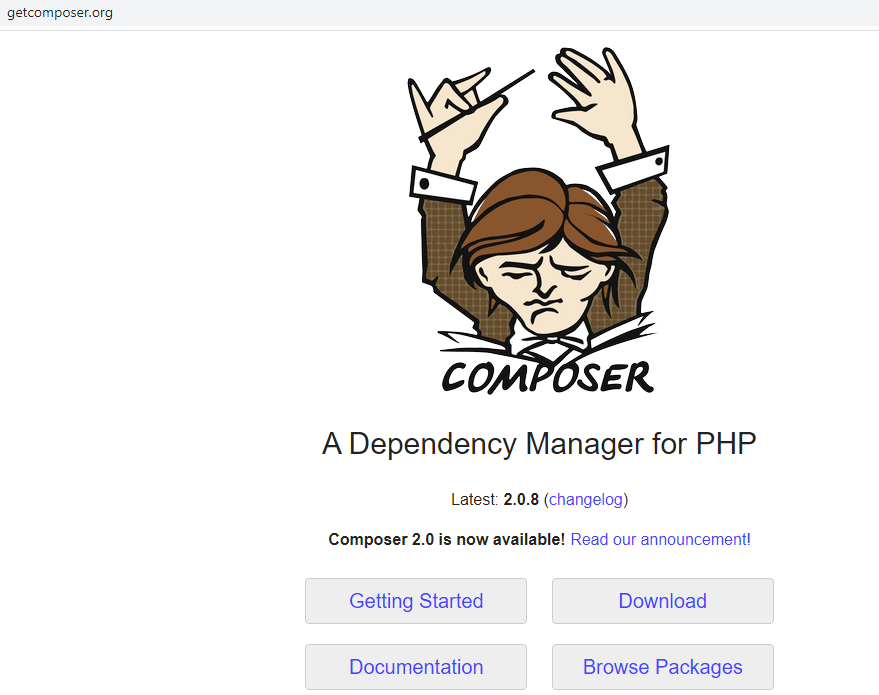
***1.2.2 Cài đặt Composer***

Composer là một trình quản lý cấp gói ứng dụng, thư viện cho ngôn ngữ lập trình PHP.

Trước tiên chúng ta phải tải về cài đặt composer trước rồi mới tiến hành cài đặt và tạo một project laravel.

Bạn có thể tải composer qua link sau: <https://getcomposer.org/>

Khi các bạn truy cập vào đường link trên các bạn sẽ thấy giao diện như hình sau:



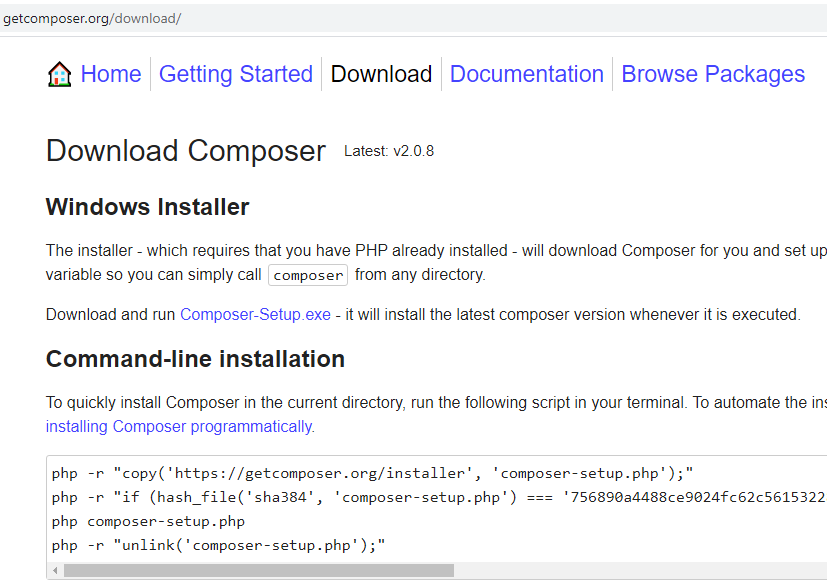
**Hình 119: Giao diện trang web của composer**

Sau đó nhấn vào nút **Download**



**Hình 120: Nhấn vào nút download composer**

Khi nhấn vào sẽ dẫn bạn đến trang web download, ở đây bạn có thể tải file **setup.exe** hoặc gõ lệnh.



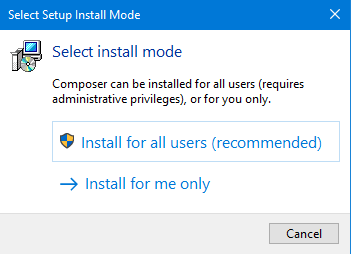
**Hình 121: Giao diện download composer**

Để tải file cài đặt bạn nhấn vào **Composer-Setup.exe**. Sau khi nhấn thì file cài đặt sẽ được tải xuống.



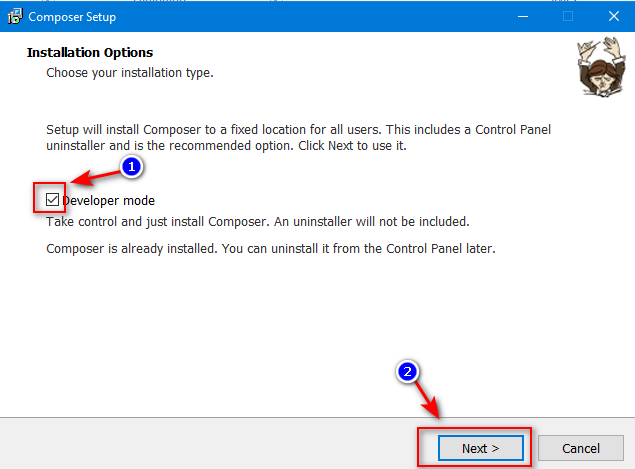
**Hình 122: Download file setup.exe**

Sau đó nhấn vào file đã download sẽ hiện ra cửa sổ cài đặt của composer:



**Hình 123: Cửa sổ giao diện install**

Ở đây các bạn nhấn vào “**Install for all users”** để bắt đầu cài đặt.



**Hình 124: Giao diện Installation Option**

Các bạn có thể click vào ô trống **Developer mode** hoặc để trống điều được. Sau đó nhấn **Next**.

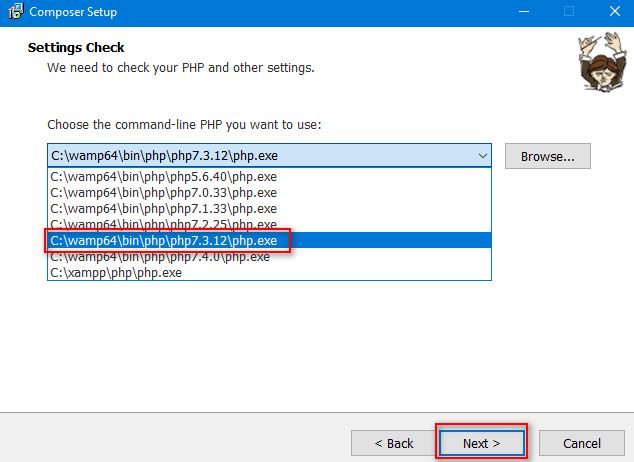


**Hình 125: Select Location composer setup**

Tùy chọn **Select Destination Location**, composer sẽ được cài theo đường dẫn.

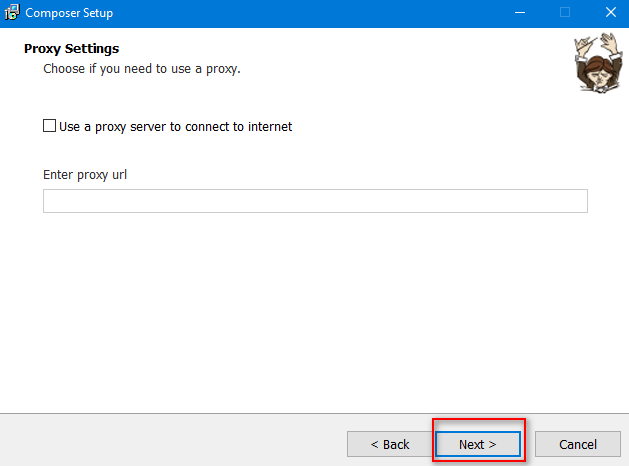
**Ví dụ:** theo hình thì composer sẽ cài theo đường dẫn: **C:\composer**.

Tiếp tục nhấn next để tiếp tục cài đặt.



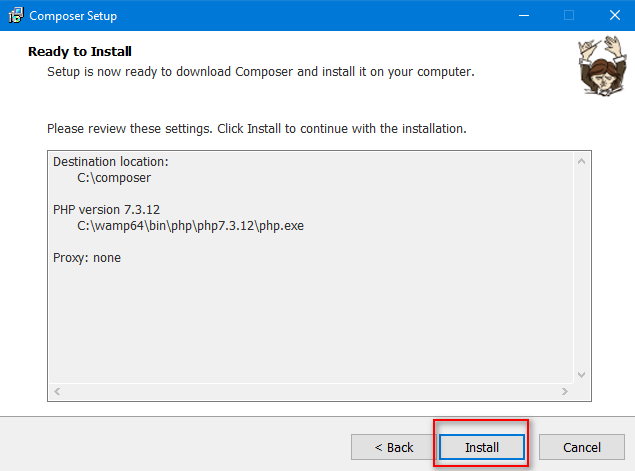
**Hình 126: Setting check your php**

Các bạn sẽ chọn phiên bản **php7.3.12** trở lên. Vì laravel 8.x trở lên **chỉ có thể** chạy trên nền **php >= 7.3**. Và tiếp tục nhấn next.



**Hình 127: Proxy setting composer**

Ở giao diện proxy setting các bạn để trống và tiếp tục nhấn next.

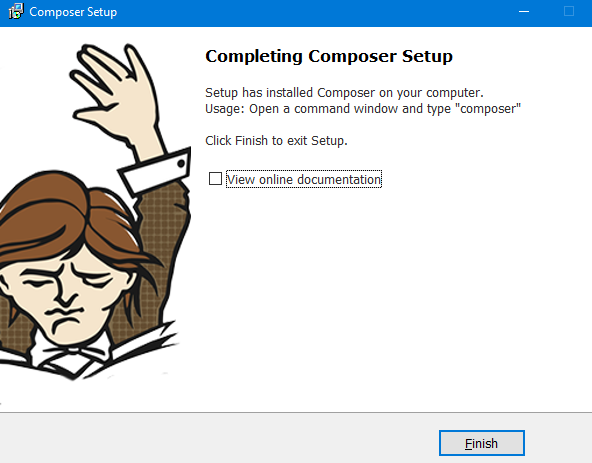


**Hình 128: Ready to install composer**

Ở giao diện **Ready to install** cho người cài đặt check lại các phần đã cài trước đó. Nếu không có thay đổi gì thì bạn nhấn **Install**.

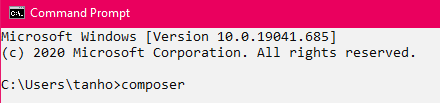
Hệ thống sẽ tự download composer cho bạn và yêu cầu bạn phải kết nối internet để cài đặt.

Sau khi cài xong như hình dưới và bạn nhấn **Finish** là hoàn tất việc cài đặt composer.



**Hình 129: Hoàn tất cài composer**

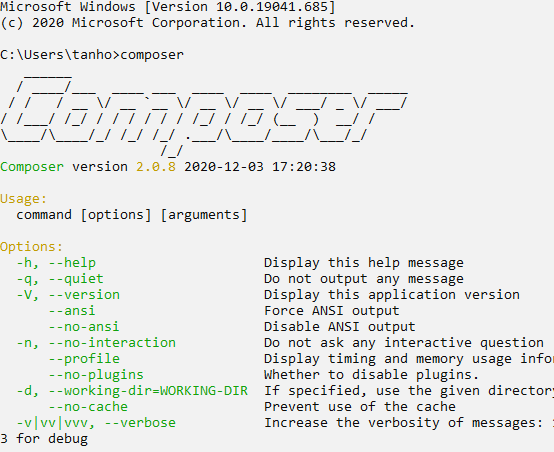
Để check xem composer đã được cài chưa bạn có thể gọi **Command Prompt** lên và gõ lệnh composer.



**Hình 130: Command Prompt và gõ composer**

Sau đó nhấn Enter.

Nếu cài thành công bạn sẽ thấy như hình 13 bên dưới.



**Hình 131: Kết quả cài composer thành công**

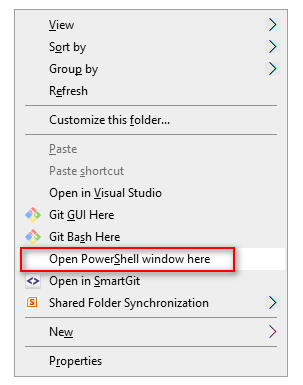
***1.2.3 Cách cài project laravel 8.x đơn giản***

Để cài một project laravel 8.x bạn có thể tham khảo qua trang web : <https://laravel.com/docs/8.x>

Ỡ đây chúng ta sẽ cài project laravel bằng cách gõ lệnh:

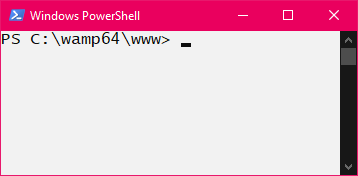
**Bước 1:**

Các bạn vào thư mục theo đường dẫn **C:\wamp64\www**. Nhấn **Shift + click** **chuột phải**. Sau đó sẽ hiện ra một cửa sổ các bạn chọn **Open PowerShell window here**.



**Hình 132: Cửa sổ khi nhấn shift và chuột phải**

Sau khi nhấn thì bạn sẽ nhận được một giao diện như sau:



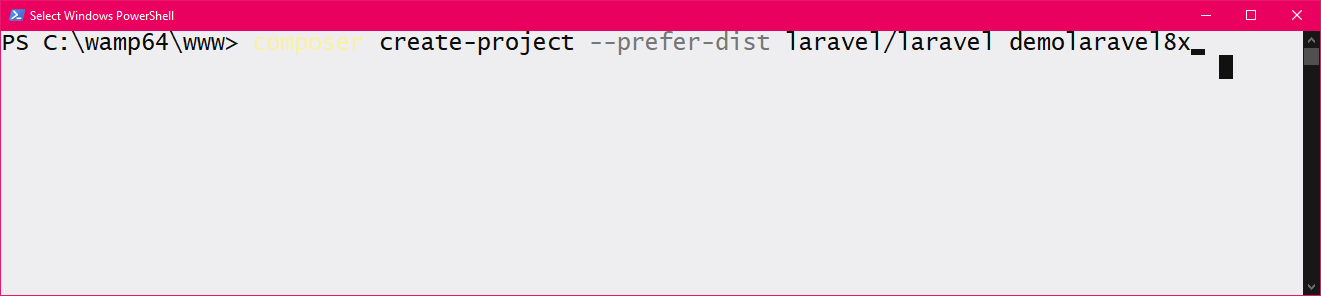
**Hình 133: Giao diện Windows PowerShell**

Để cài tự động laravel về thì các bạn gõ câu lệnh như sau vào **Window PowerShell**:

composer create-project --prefer-dist laravel/laravel blog

Trong đó bạn để ý chữ “**blog**” ở cuối dòng. Các bạn có thể đặt tên cho project theo ý mình (vd: demolaravel8x)

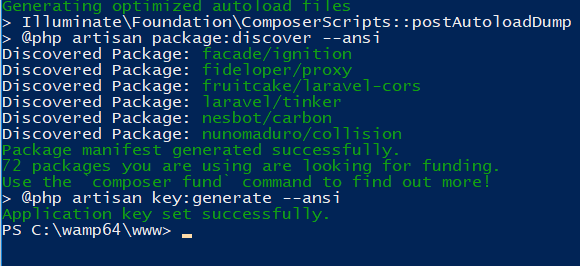
composer create-project --prefer-dist laravel/laravel demolaravel8x



**Hình 134: Gõ câu lệnh tạo laravel**

Sau đó nhấn Enter. Thì hệ thống sẽ tự động download các file, package cần thiết cho laravel.

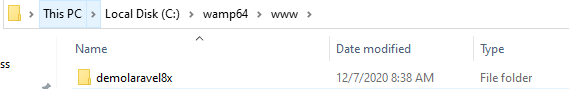
**Yêu cầu:** Các bạn phải kết nối internet trước khi tải laravel.



**Hình 135: Cài thành công laravel**

Khi nào các bạn thấy cửa sổ **Window PowerShell** như hình trên thì bạn đã thành công cài đặt một project laravel rồi.

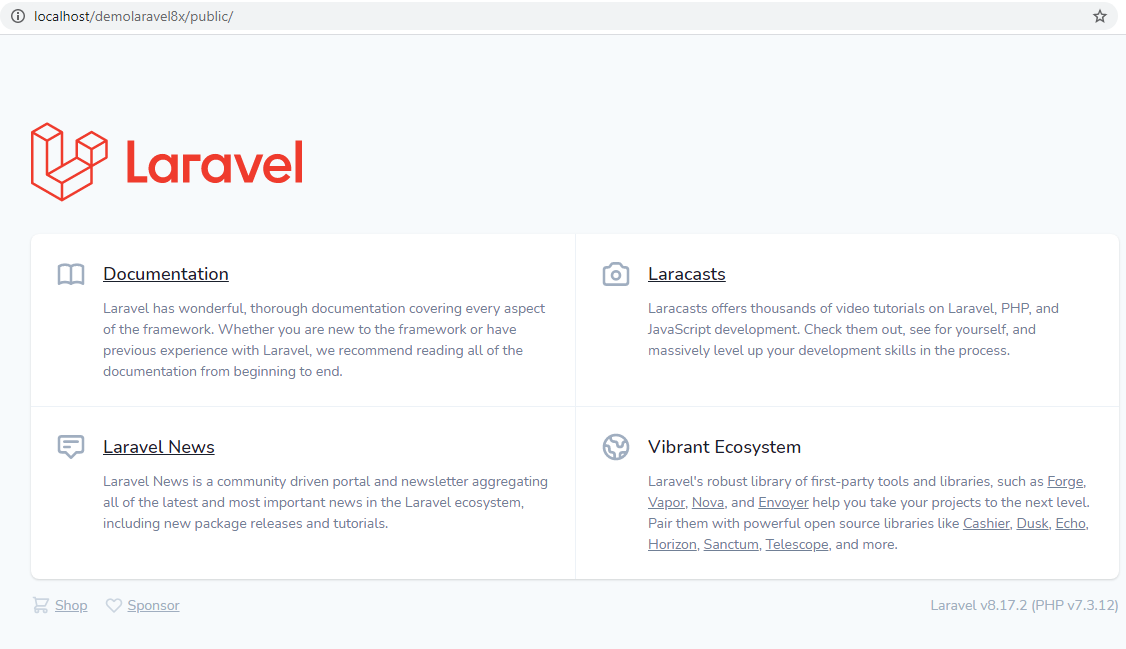
Bạn vào đường dẫn **C:\wamp64\www** sẽ thấy 1 folder có tên **demolaravel8x** đã được tạo và bên trong là các file cần thiết cho laravel 8.x.



**Hình 136: Folder demolaravel được tạo**

Để kiểm tra các bạn vào trình duyệt internet gõ đường dẫn như sau: **localhost/demolaravel8x/public**

Nếu bạn đến được giao diện như hình sau:



**Hình 137: Giao diện laravel trên trình duyệt**

Vậy là bạn đã cài thành công một project đơn giản với laravel 8.x.

**2. Tạo một gói Composer/Packagist cơ bản**

Sau khi tạo một package, bạn có thể tạo một dự án mới hoặc cập nhật một dự án đã tồn tại để sử dụng Composer. Với phương pháp này, bạn có thể tạo các dự án phức tạp và chia sẻ chúng với các lập trình viên khác trên toàn thế giới.

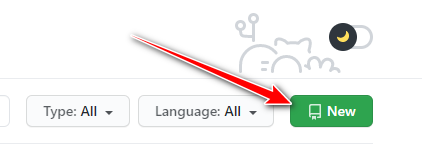
**2.1 Tạo một kho lưu trữ với github**

Trước tiên, chúng ta sẽ tạo một **kho lưu trữ rỗng** trên **github** và sau đó ta sẽ đưa source code của **project laravel 8.x** mà ta đã tạo ở phần trên vào.

Ở bài hướng dẫn này sẽ chỉ cách tạo một **kho lưu trữ** trực tiếp trên github website.

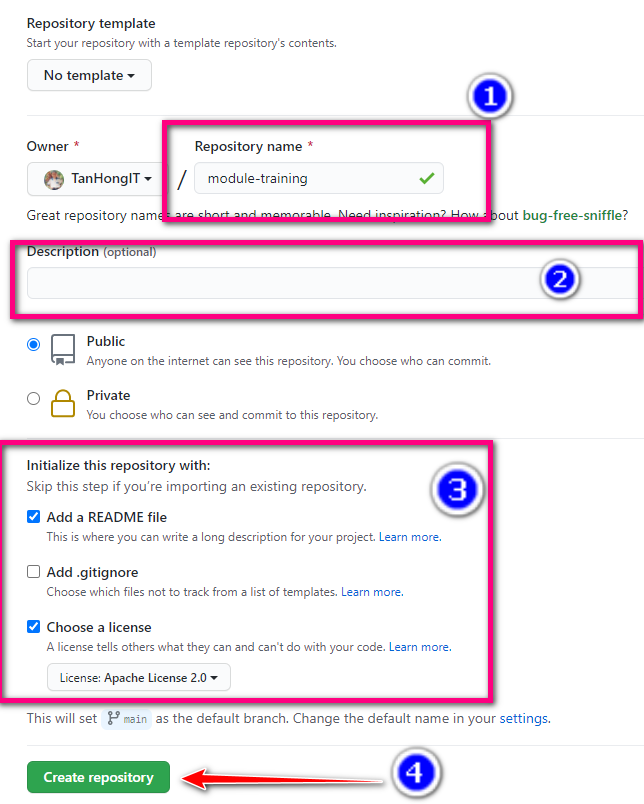
Các bạn hãy đăng nhập tài khoản git của mình và truy cập vào đường dẫn sau: <https://github.com/new>

Hoặc bạn có thể nhấn vào nút **New** có background màu xanh tại trang danh sách các **repositories** để tạo một project mới.



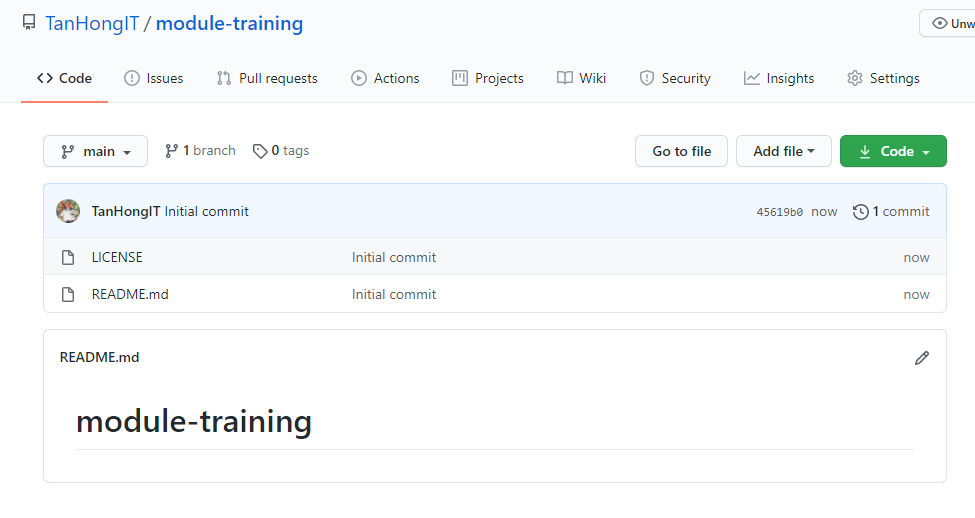
**Hình 138: Nút tạo repo mới**

Tiếp theo bạn hãy điền thông tin tương ứng và nhấn vào button Create repository.



**Hình 139: Nhập đầy đủ thông tin cần thiết và nhấn nút tạo**

Sau khi chờ đợi bạn sẽ có một kho lưu trữ rông như hình sau:



**Hình 140: Kho lưu trữ minh họa sau khi vừa tạo xong**

Hình trên là ví dụ về một kho lưu trữ mà mình đã tạo với nhánh chính có tên là “**main**”.

Bước tiếp theo chúng ta sẽ Clone kho lưu trữ trên về máy tính.

**Lưu ý:**

Trước khi thực hiện clone về máy bạn cần phải cài Git vào máy tính. Các bạn có thể tải về và cài đặt theo đường link sau: <https://git-scm.com/downloads>

Các bạn vào thư mục theo đường dẫn **C:\wamp64\www**. Nhấn **Shift + click** **chuột phải**. Sau đó sẽ hiện ra một cửa sổ các bạn chọn **Open Command Prompt here**.

Sau khi mở lên ban hãy thực hiện theo lệnh:

**git clone link\_project**

Trong đó **link\_project** là đường dẫn đến kho lưu trữ mà bạn vừa mới tạo xong.

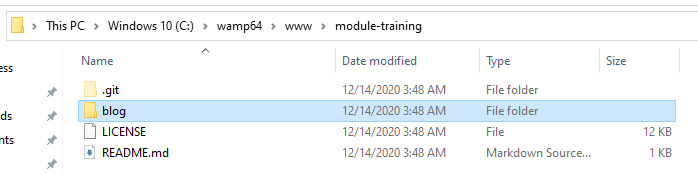
**Ví dụ:** git clone https://github.com/TanHongIT/module-training



**Hình 141: Lệnh clone một kho lưu trữ**

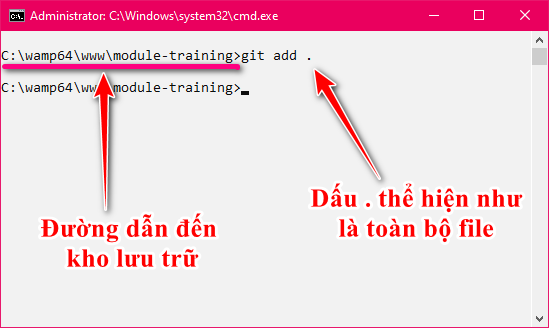
Sau khi đã clone về bạn mở hãy copy source code **project laravel 8.x** mà ta đã tạo vào.

Ở hình dưới mình đã copy project có tên là blog vào kho lưu trữ.



**Hình 142: Copy project laravel có tên "blog" vào kho lưu trữ**

Sau khi copy xong, quay lại màn hình của sổ **Command Prompt**, bạn hãy thực hiện câu lệnh **git add .** để thêm toàn bộ thay đổi vào bộ đệm của git dưới máy tính.

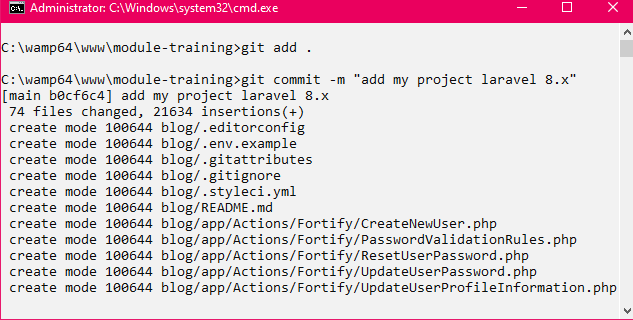


**Hình 143: Đến đường dẫn của repo và gõ lệnh git add .**

Tiếp theo thực hiện câu lệnh **git commit -m “your add message”**

Phần ký tự nằm trong dấu ngoặc kép là phần tin nhắn cam kết mà bạn muốn upload lên kho lưu trữ. Bạn thay đổi lại theo ý muốn và sau đó thì nhấn **Enter**.

**Ví dụ:** git commit -m “add my project laravel 8.x”



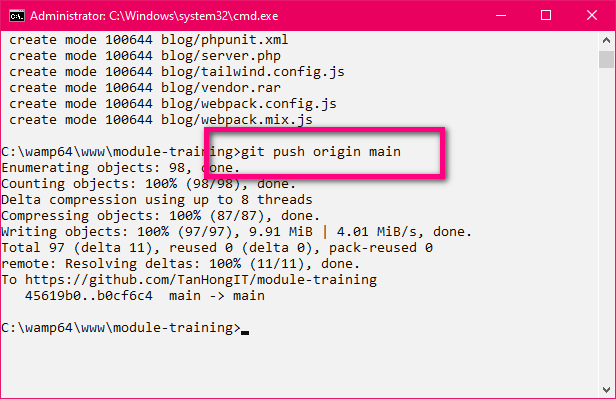
**Hình 144: Câu lệnh commit cam kết bắt buộc**

Tiếp theo chúng ta sẽ thực hiện câu lệnh push lên github bằng câu lệnh:

**git push origin main**

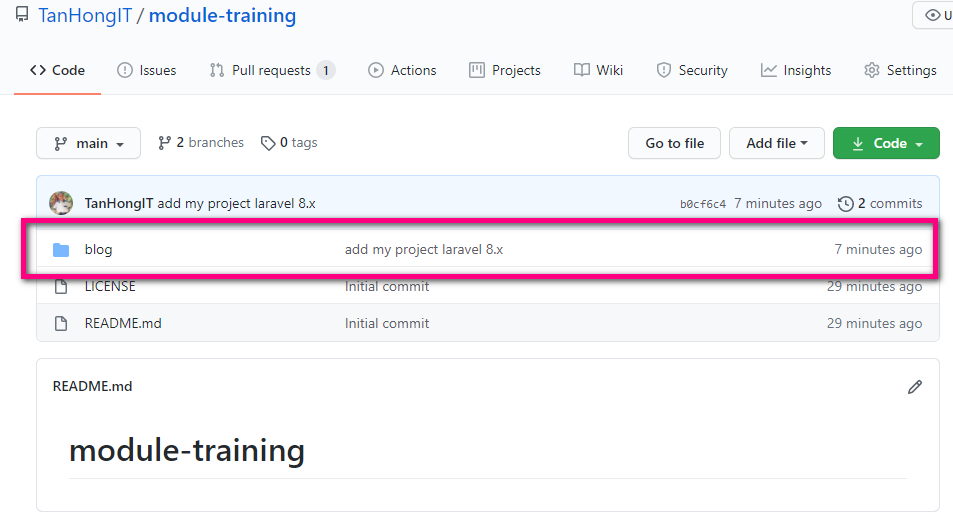
**Trong đó:** từ khóa “**main**” là tên nhánh chính của kho lưu trữ.

Bạn phải kiểm tra tên nhánh chính của kho lưu trữ và thay đổi lại cho phù hợp nhé. Khi đã thay đổi đúng tên thì nhấn Enter để tiến hành push. Điều quan trọng là máy tính của bạn phải có kết nối internet nhé.



**Hình 145: Câu lệnh push lên kho lưu trữ qua nhánh chính**

Sau đó chúng ta hãy lên đường dẫn của kho lưu trữ trên trình duyệt để kiểm tra lại.



**Hình 146: Kho lưu trữ đã được thêm thành công project laravel**

Chúng ta thấy rằng source code đã được upload lên github thành công với dòng commit là “add my project laravel 8.x”

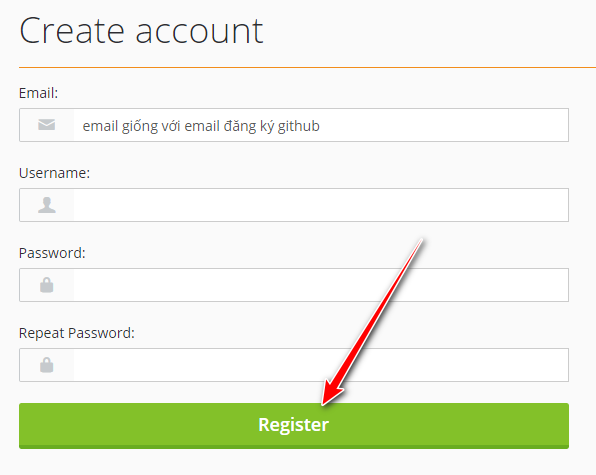
**2.2 Tạo Package**

Chúng ta sẽ tạo package với **github** kết nối với **packagist**

Chúng ta cần có một tài khoản trên trang **packagist.org**. Nếu bạn chưa có hãy bắt đầu tạo một tài khoản mới với email tạo phải trùng với email đã đăng ký tài khoản github.

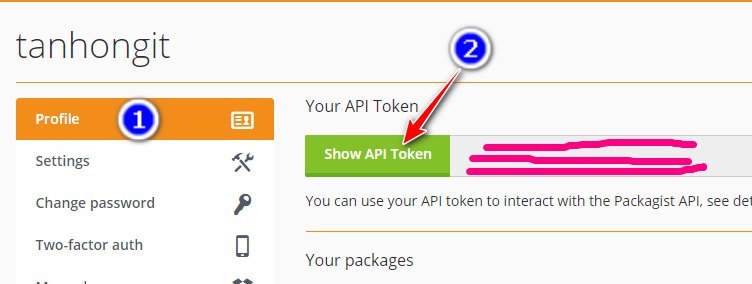
Vài link <https://packagist.org/register/> để tiến hành đăng ký tài khoản.

Nhập đầy đủ thông tin và sau đó nhấn vào button **Register**.



**Hình 147: Form đăng ký tài khoản packagist**

Sau khi tạo xong tài khoản, bạn hãy vào phần **Profile** và nhấn vào **Show API Token**.

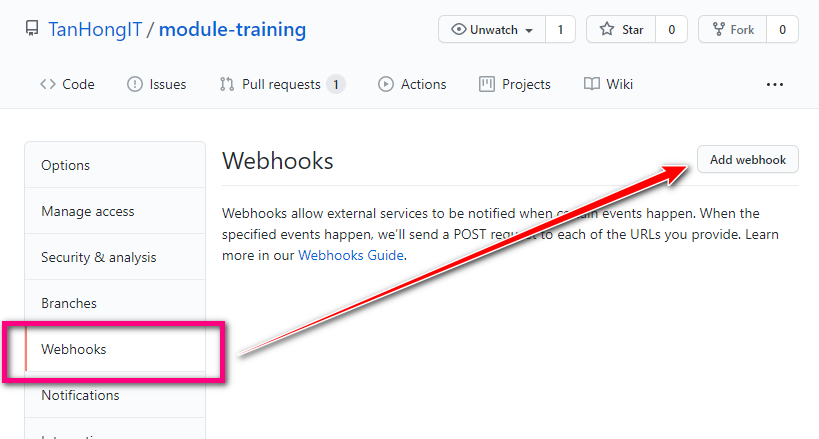


**Hình 148: Cách lấy API tooken để submit repo**

Khi này API sẽ được hiển thị ở thẻ input kế bên cạnh, bạn hãy bảo mật cẩn thận API này đừng cho bất ý ai biết.

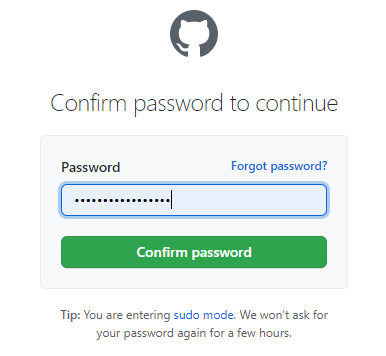
Quay lại trang chứa kho lưu trữ của bạn và đi qua tab **Setting**s.

Sau đó vào mục **Webhooks** và nhấn vào button “**Add webhook”**.



**Hình 149: Tạo webhook**

Bước tiếp theo github sẽ yêu cầu bạn nhập lại mật khẩu để xác nhận bảo mật hành động.



**Hình 150: Yêu cầu nhập lại mật khẩu**

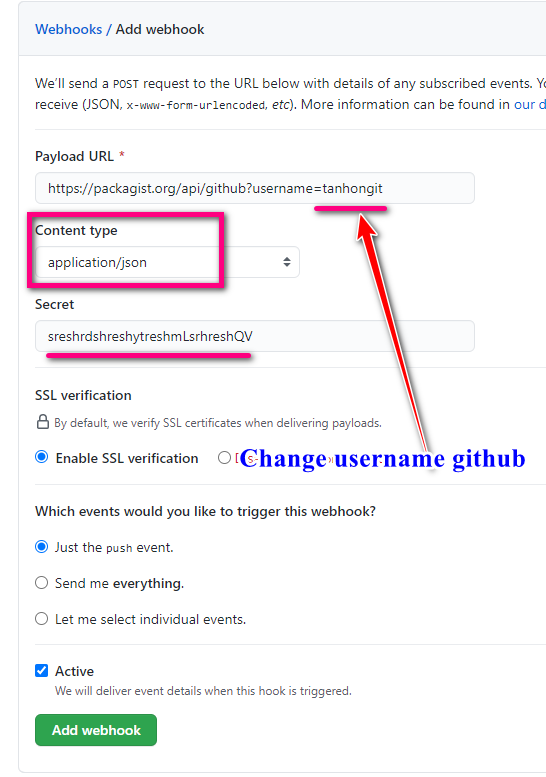
Tại đây bạn hãy thay đổi lại **Payload URL** là https://packagist.org/api/github?username=tanhongit

Bạn nhớ đổi usernane lại thành username tài khoản github của các bạn nhé.

Về **Content type**, bạn chuyển nó về: application/json

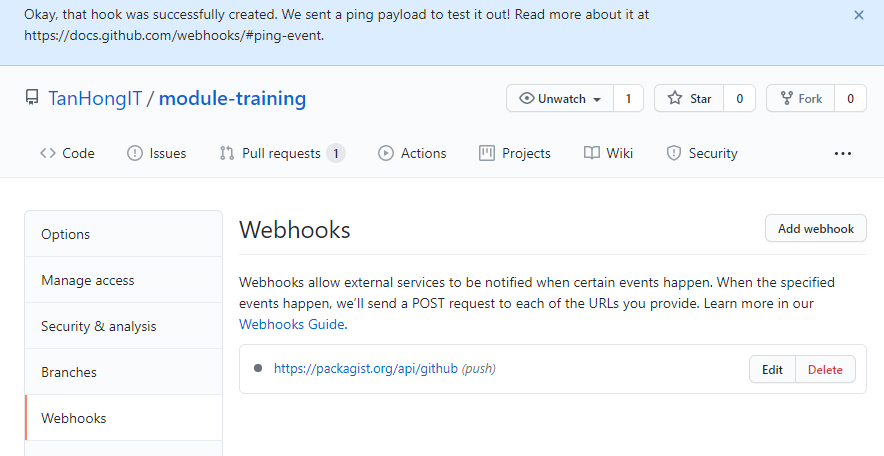
Ở trường nhập **Secret**, bạn hãy dán đoạn API token mà nãy chúng ta vừa lấy bên **packagist** vào.

Sau khi đã hoàn tất các yêu cầu trên thì ta nhấn vào nút **Add webhook**.



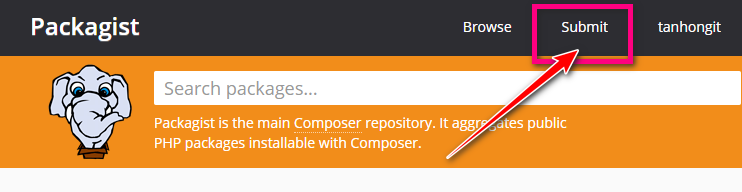
**Hình 151: Điều chỉnh thông tin cho webhook**

Chúng ta sẽ được một webhook trong kho lưu trữ như sau:



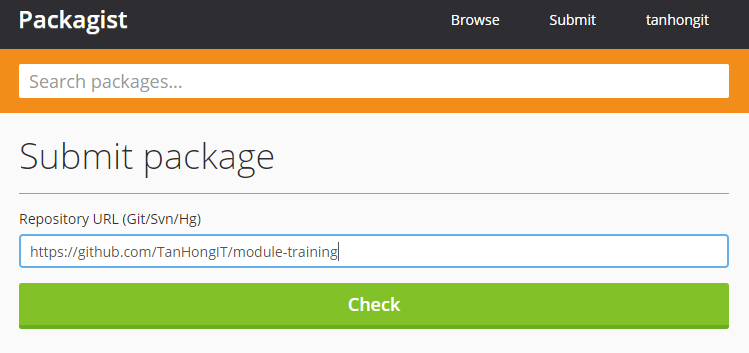
**Hình 152: Webhook được tạo thành công**

Giờ chúng ta lại quay sang **packagist.org** và nhấn vào tab **Submit**.



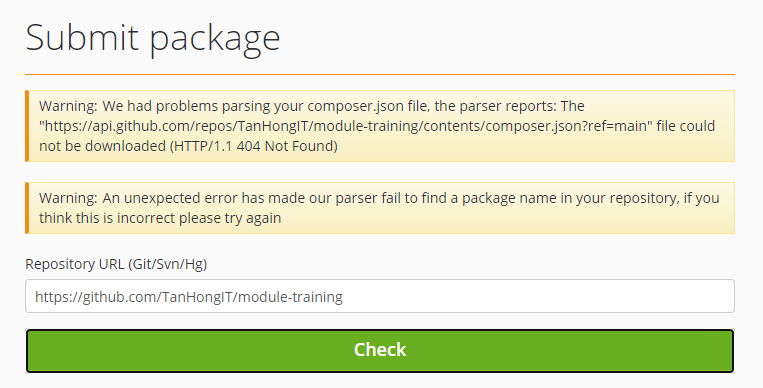
**Hình 153: Submit package**

Tiến hành paste đường dẫn của kho lưu trữ vào trường nhập liệu và nhấn nút Check.



**Hình 154: Ckeck repo**

Chắc chắn khi này chúng ta sẽ thấy thông báo lỗi bởi bì chúng ta chưa tạo file **composer.json** để cho **packagist.org** hiểu được.



**Hình 155: Lỗi chưa có file composer.json**

Tiến hành tạo một file **composer.json** có nọi dung như sau:

{

    "require": {

        "vendor/package": "1.3.2",

        "vendor/package2": "1.\*",

        "vendor/package3": "^2.0.3",

        "php": ">=7.3",

        "another-vendor/package": "1.\*"

    },

    "name": "tanhongit/laravel-training-module",

    "description": "Training Project Laravel Module",

    "keywords": [

        "tanhongit"

    ],

    "authors": [

        {

            "name": "tanhongit",

            "email": "phuongtan12357nguyen@gmail.com"

        }

    ],

    "license": "MIT",

    "autoload": {

        "psr-4": {

            "tanhongit\\module\\": "/"

        }

    },

    "minimum-stability": "dev"

}

Bạn hãy sửa lại thông tin trong đoạn code cho phù hợp với kho lưu trữ của bạn.

Bạn lưu file này nằm ở cấp thư mục cha trong kho lưu trữ.



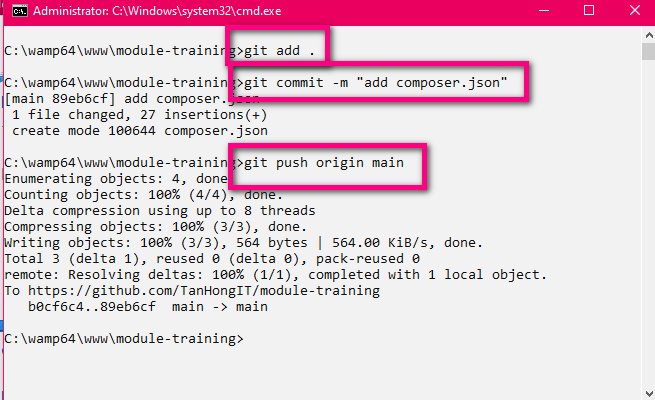
**Hình 156: đặt file ở cùng cấp thư mục cha**

Sau đó ta thực hiện 3 câu lệnh huyền thoại lần lượt là:

git add .

git commit -m “tin nhắn cam kết”

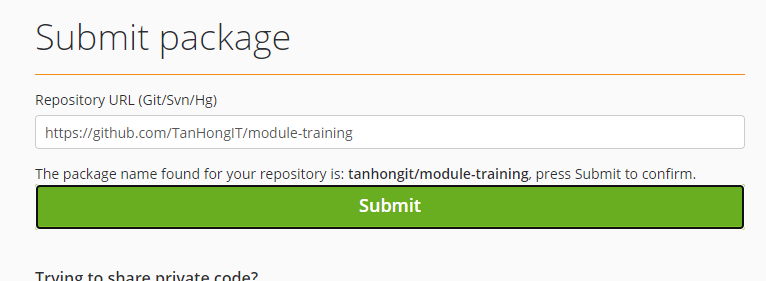
git push origin nhánh\_chính



**Hình 157: Ba câu lệnh để push lên git**

Cuối cùng, sau khi đã push lên github xong, ta tiến hành check lại một lần nữa.

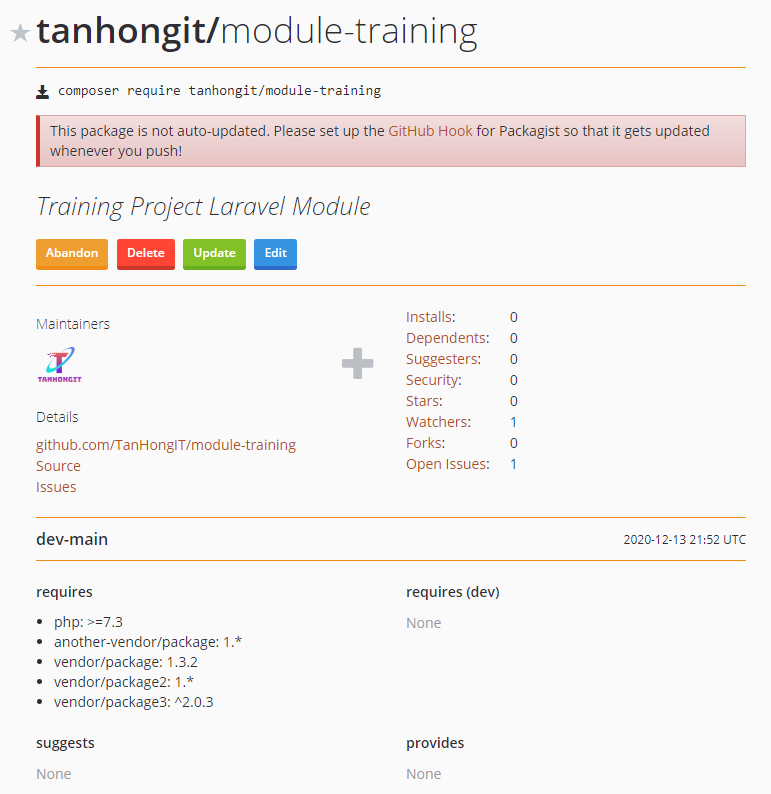
Nếu bạn thấy có thông báo như hình dưới thì bạn đã kiểm tra thành công.



**Hình 158: Check thành công**

Bây giờ bạn hãy nhấn vào nút **Submit** để kiến hành xác nhận.

Sau khi đã nhấn **Submit** bạn sẽ có một trang hiển thị thông tin package giống như hình bên dưới.



**Hình 159: Result submit**

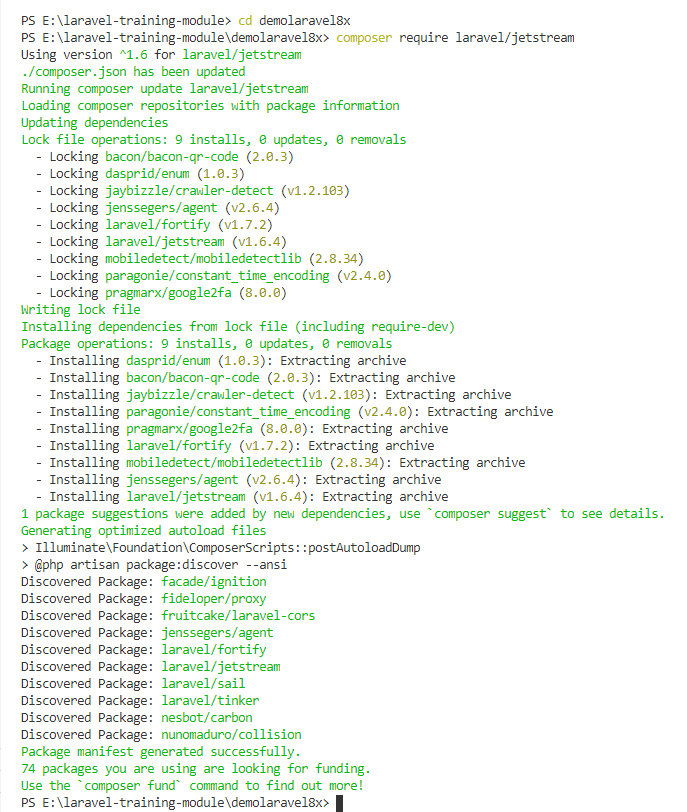
**3. Tạo Chức năng login thông qua thư viện jetstream**

**3.1 Tạo chức năng login thông qua email**

Sau khi đã tạo thành công project cơ bản với laravel và bây giờ sẽ tiếp tục thêm tính năng login, register thông qua thư viện hỗ trợ laravel mang tên **jetstream** với câu lệnh như sau:

**cd demolaravel8x**

**composer require laravel/jetstream**

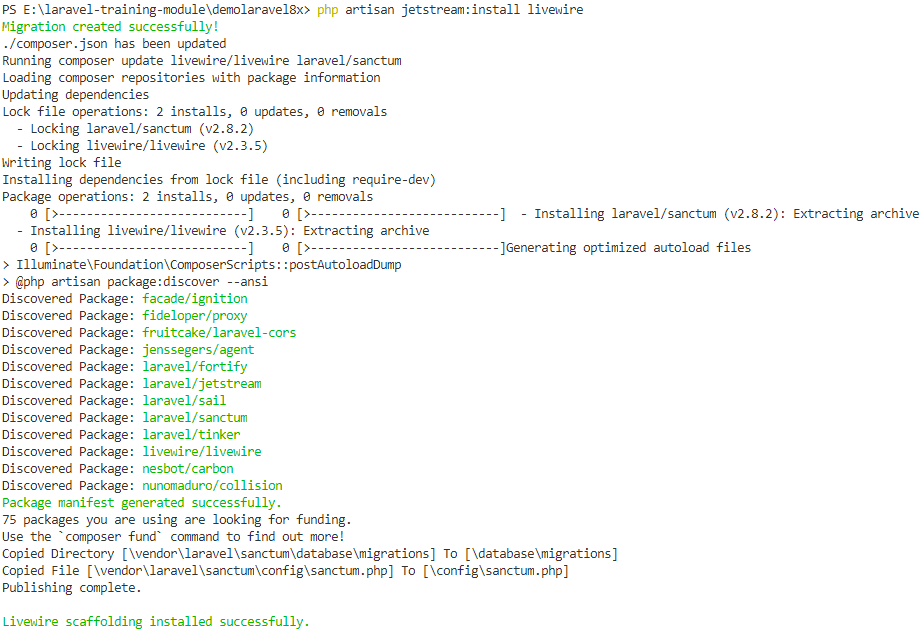


**Hình 160: Cài đặt thư viện jetstream**

Trong đó, **demolaravel8x** là tên đường dẫn đến tên project laravel mà bạn đã tạo.

Sau khi đã cài đặt thư viện **jeatstream** và được kết quả như hình trên tiếp tục cài đặt gói hỗ trợ của thư viện jetstream thông qua câu lệnh:

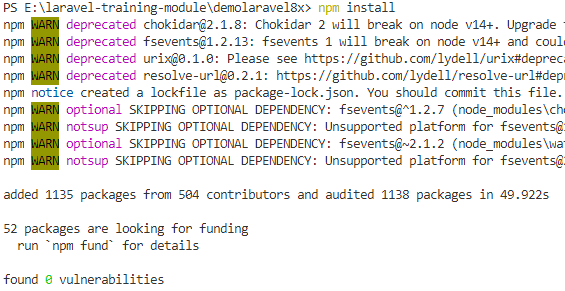
**php artisan jetstream:install livewire**



**Hình 161: Cài đặt gói hỗ trợ thư viện jetsteam**

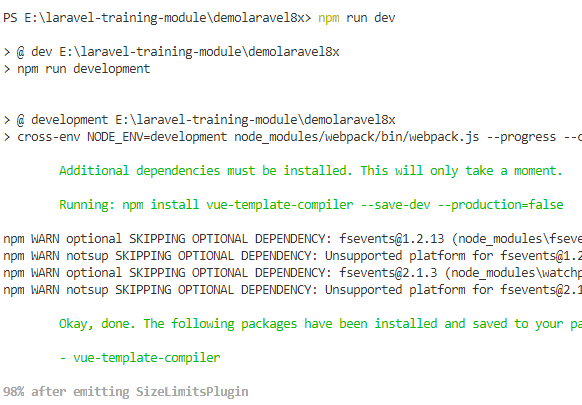
Sau khi cài đặt thành công lúc này bạn sẽ cần phải cài đặt thêm **npm** với câu lệnh như sau:

**npm install** và sẽ được kết quả như hình dưới đây:



**Hình 162: Cài đặt npm**

Sau khi hoàn thành cài đặt **npm** và đạt được kết quả như hình (số lượng packages có thể thay đổi tùy vào số lượng package cần được cài đặt và cập nhất trong máy) tiến hành chạy câu lệnh: **npm run dev**

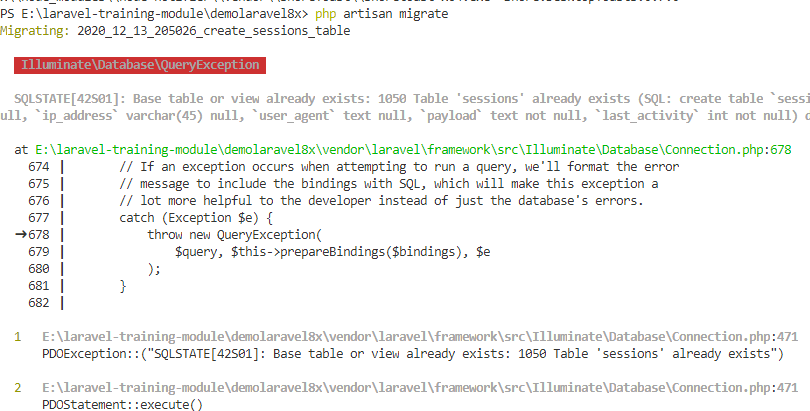


**Hình 163: Quá trình khởi chạy npm dev**



**Hình 164: Hoàn tất khởi chạy npm dev**

Sau khi cài đặt và thực thi câu lệnh **npm run dev** sẽ tiếp tục chạy lệnh: **php artisan migrate** hoàn tất việc cập nhật các table lên database. Tuy nhiên khi chạy bạn sẽ phát hiện lỗi như hình dưới đây:

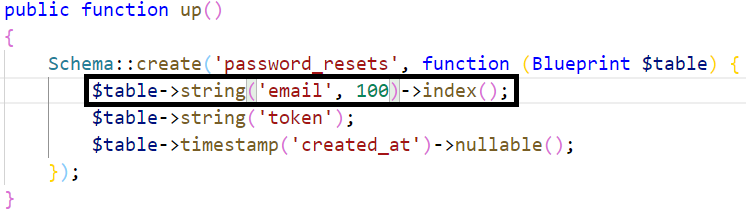
****

**Hình 165: Lỗi update table lên database**

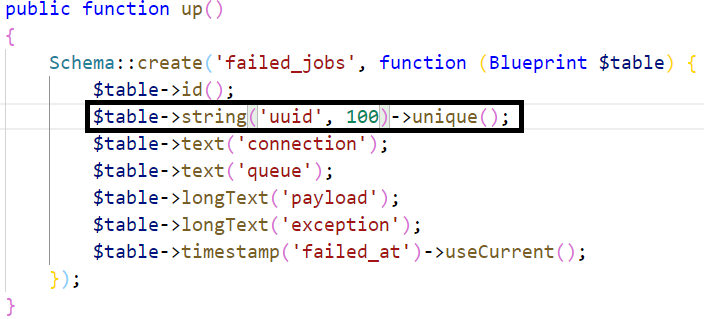
Vào file migration theo đường dẫn: **demolaravel8x/database/migrations/** tìm đến file **create\_user\_table.php** tiến hành cập nhật số lượng ký tự cho email trong các table theo các hình dưới đây:



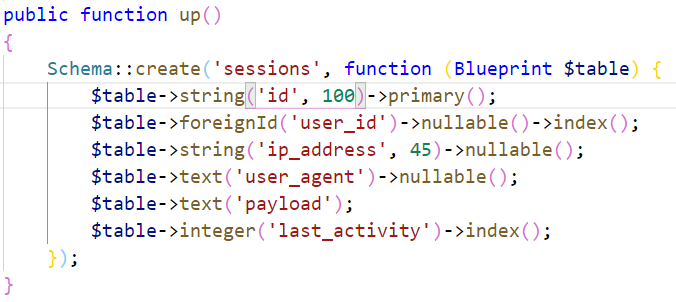
**Hình 166: Cập nhật số lượng ký tự cho email trong table create\_user**



**Hình 167: Cập nhật số lượng ký tự email trong table create\_password\_resets**

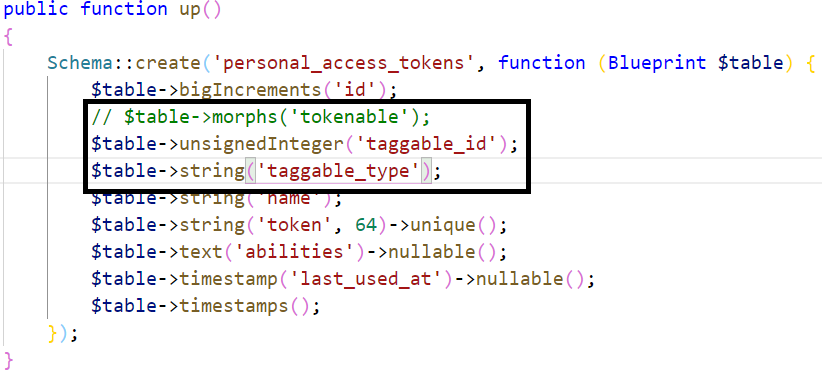


**Hình 168: Cập nhật số lượng ký tự cho table failed\_jobs**



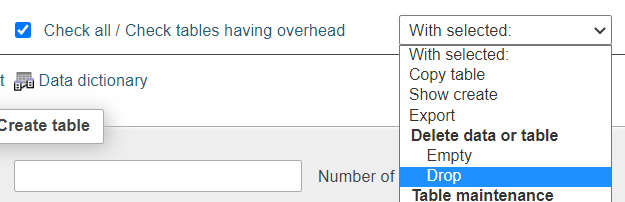
**Hình 169: Cập nhật số lượng ký tự cho table create\_session**

Sau khi đã chỉnh sửa các file liên quan đến ký tự ta sẽ tiếp tục chỉnh sửa trong file **create\_personal\_access\_tokens\_table.php** để tránh tình trạng xung đột giữa các table với nhau nên sẽ thêm một cột mới với tên là **taggable**\_**id** vó kiểu dữ liệu là số nguyên không dấu (**unsignedInteger**) theo hình dưới đây:

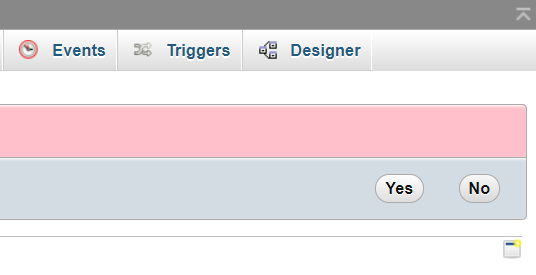


**Hình 170: Cập nhật table create\_personal tránh xung đột**

Sau khi cập nhật các ký tự cho các table cần thiết, ta tiến hành truy cập phpmyadmin và chọn drop các table cũ trong database và sau đó chọn **yes**:

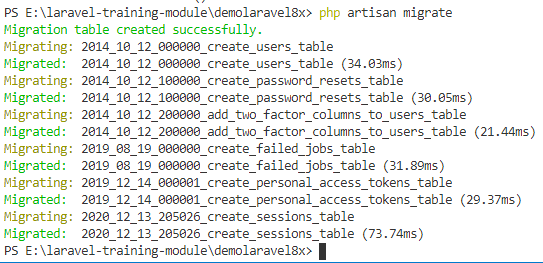


**Hình 171: Chọn tất cả các table và drop**



**Hình 172: Chấp nhận drop tất cả các table**

Tiến hành cập nhật lại file database thông qua câu lệnh: **php artisan migrate** sẽ được kết quả như hình dưới đây:



**Hình 173: Cập nhật tất cả table lên database**

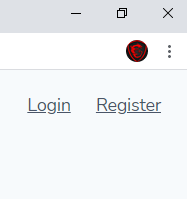
Sau khi đã cập nhật thành công file database cho project và giờ ta sẽ tiến hành chạy câu lệnh dùng để kích hoạt máy ảo của laravel như sau:

**php artisan serve**



**Hình 174: Kích hoạt máy ảo laravel**

Sau khi truy cập vào máy ảo của laravel bạn sẽ thấy **login** và **register** vừa được cập nhật tại góc trên cùng bên phải và tiến hành đăng ký, đăng nhập bình thường:

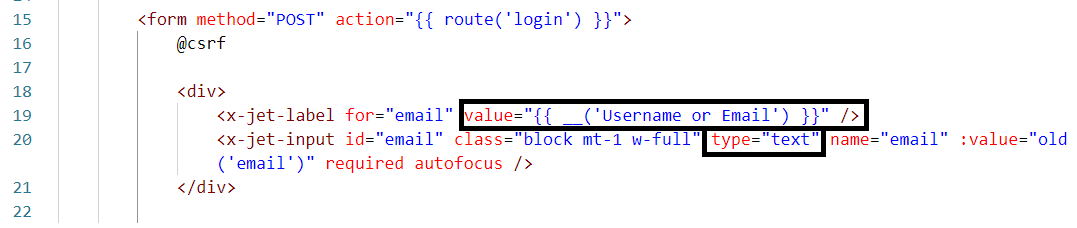


**Hình 175: Chức năng login và register**

**3.2 Tạo chức năng login thông qua email hoặc username**

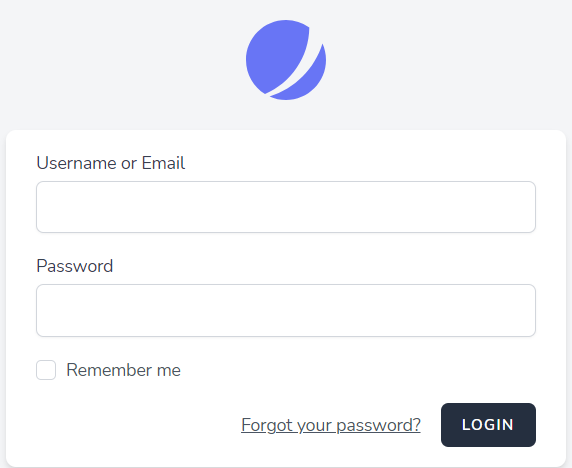
Để có thể đăng nhập thông qua email hoặc username ta sẽ phải đưa **type** của field nhập liệu trong form từ dạng **email** về dạng **text** (nằm trong file **login.blade.php**) theo đường dẫn như sau:

**\resources\views\auth\login.blade.php**

****

**Hình 176: Thay đổi giá trị type và value**

Tại đoạn value chỉ đơn thuần là xuất ra câu thông báo Username or Email cho người dùng nhận biết như hình dưới đây:



**Hình 177: Thông báo người dùng login thông qua username hoặc email**

Sau khi đã chỉnh sửa **type input** thì ta sẽ tiếp tục chỉnh sửa trong **CORE(vendor)** nhận giá trị username hoặc là email.

**LƯU Ý:**

Vì đây là **core** của cả framework laravel nên sẽ cực kỳ nguy hiểm nếu bạn không biết bạn đang làm gì vì điều này có thể làm framework **hoạt động không bình thường** hoặc **không thể hoạt động.**

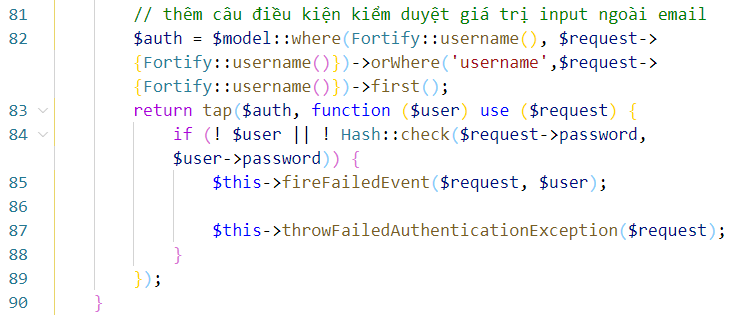
Để có thể gọi là tạm thời biết được bạn đang chỉnh sửa gì trong core thì hãy tìm đến file của git có tên là **.gitignore** và xóa dòng **/vendor** để git nhận và cho biết đã và đang chỉnh sửa gì, file nào trong **core.**

Sau khi đã xóa dòng **/vendor** ta sẽ tiến hành chỉnh sửa như sau:

Tìm đến file theo đường dẫn: **\vendor\laravel\fortify\src\Actions\RedirectIfTwoFactorAuthenticatable.php**

Khóa câu lệnh cũ bằng cách comment code thông qua tổ hợp phím **Ctrl + Shift.**

Và thực hiện cập nhật chức năng như hình dưới đây:



**Hình 178: Cập nhật giá trị đầu vào username hoặc email**

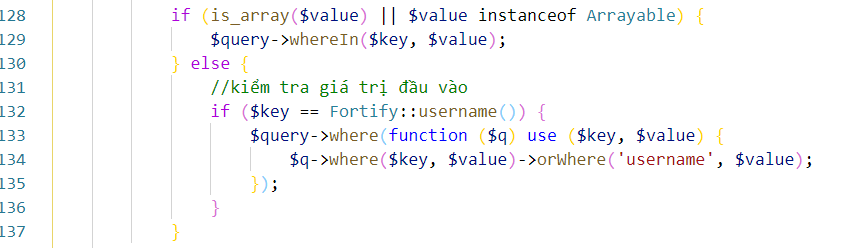
Ta sẽ tạo một biến mới có tên là **$auth** và cho nhận giá trị nhập vào là username hoặc email thông qua: **orWhere**. Và truyền vào hàm **tap(),** hiện tại chỉ dừng ở mức tinh chỉnh cho việc login bằng username hoặc email nên chúng ta sẽ không đi sâu vào cách vận hành của thư viện **jetstream**.

Tiếp đến ta sẽ truy cập vào file theo đường dẫn:

**\vendor\laravel\framework\src\Illuminate\Auth\EloquentUserProvider.php**

Đừng quên comment lại code mặc định của core trước khi chỉnh sửa code. (Dễ dàng quay lại code cũ nếu có trở ngại)

Tìm đến hàm **retrieveByCredentials()** Tại hàm **else** ta sẽ tiến hành cập nhật lại giá trị vào là username hoặc email như hình dưới đây:



**Hình 179: Kiểm điều kiện đầu vào**

Nếu trong **else** chỉ có **$q->where($key, $value)** thì giá trị nhận vào mặc định sẽ là **email** do **thư viện định nghĩa từ trước**, vậy nên lúc này ta sẽ cần điều kiện **if** nhận giá trị đầu vào và tiến hành xây dựng hàm lấy ra giá trị **username** hoặc **email** do người dùng nhập vào.

## **4. List User**

Sau khi hoàn thành chức năng login thông qua username hoặc email như đã hướng dẫn trên ta sẽ tiếp tục làm đến chức năng tạo list user để hiển thị tất cả users ra một danh sách riêng biệt.

Điều đầu tiên khi tạo một route mới cho nơi để hiển thị danh sách user.

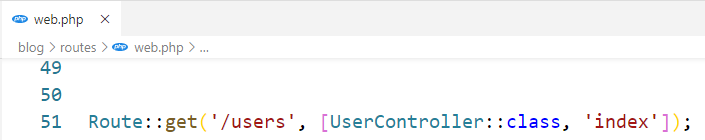
**Route::get('/(tên root), [(tên controller)::class, '(hàm hiển thị mặc định)']);**

Sau khi đã tuân thủ theo quy tắc tạo route cơ bản ta sẽ tiến hành cập nhật vào file **web.ph**p theo đường dẫn sau: **project\_name/routes/web.php.**



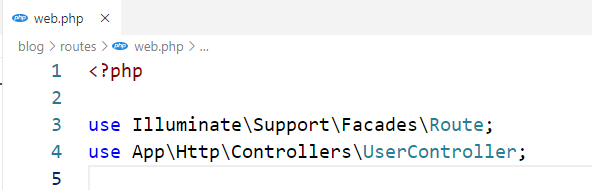
**Hình 180: Đường dẫn tới file web.php**

Sau khi tìm đến file **web.php** là nơi chứa toàn bộ route, ta sẽ tiến hành cập cật file như sau:



**Hình 181: Route users**

Để cho Route nhận được controller của user và trả về view ta sẽ cần phải khai báo controller trong file web.php như sau:



**Hình 182: Khai báo UserController**

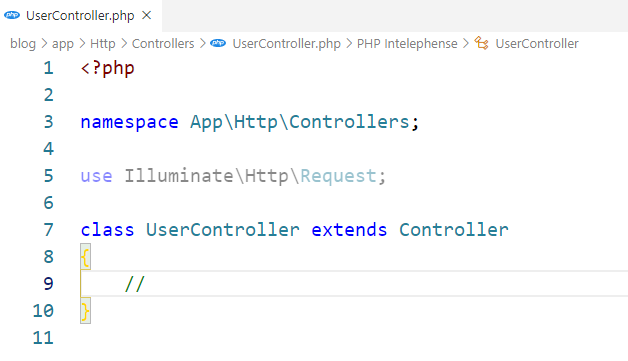
Vì mục đích làm một trang hiển thị toàn bộ các user nên sẽ dùng **UserController** để thực hiện điều đó, vậy nên ta sẽ tiến hành dùng câu lệnh như sau:

**php artisan make:controller (tên controller)**

****

**Hình 183: Tạo UserController**

Hoàn thành tạo controller thông qua câu lệnh ta sẽ được file **UserController.php** với nội dung như sau:



**Hình 184: UserController**

Để có thể cho route biết view cần được hiển thị ta sẽ tiến hành tạo hàm index bên trong controller như hình sau:



**Hình 185: Tạo hàm index trả về view**

Sau khi đã tạo thành công controller cho user ta tiến hành tạo trang giao diện cho list user theo đường dẫn dưới đây (thư mục users là tự tạo):



**Hình 186: đường dẫn tạo trang cho list user**

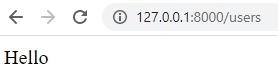
Tiến hành tạo giao diện cơ bản và chạy thử nghiệm cho trang list users như sau:



**Hình 187: Giao diện cơ bản trang list users**

Sau khi đã tạo xong **UserController, list.blade.php** và định nghĩa route trong **web.php** thì thực hiện tiếp bước sau.

Chạy câu lệnh **php artisan serve** và khởi động thông qua server của Laravel để được kết quả:



Tiến hành đưa ra toàn bộ danh sách users có trong database ta cần đến model Flight để có thể làm một cách đơn giản nhất, tiến hành cài đặt thông qua câu lệnh:

**php artisan make:model Flight**

****

**Hình 188: Cài đặt model Flight cho project**

Sau khi cài đặt xong thư viện ta tiến hành chỉnh sửa file UserController cho việc xuất lấy tất cả thông tin user như hình dưới đây:



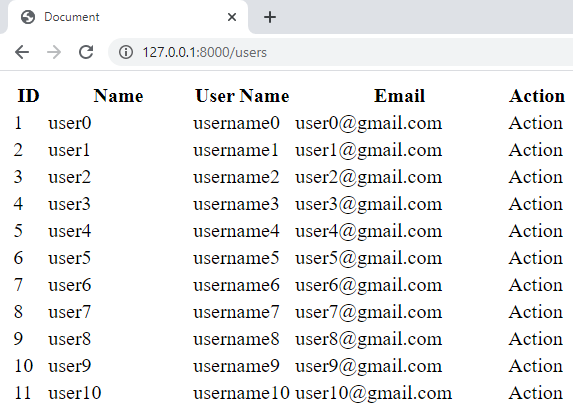
**Hình 189: Lấy tất cả thông tin uers**

Sau khi đã lấy được thông tin tất cả users ta tiến hành đẩy dữ liệu lên trang giao diện list user như hình dưới đây:



**Hình 190: Xuất thông tin của tất cả user ra trang list users**

Vì đây là trang với giao diện cơ bản nên sẽ không có CSS nên sẽ được kết quả như hình dưới đây:



Hình : Kết quả trang list user

# **TÀI LIỆU THAM KHẢO**

* <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>