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** 7](#_Toc58502460)

[**1. Bảng phân công công việc** 7](#_Toc58502461)

[**2. Sprint Backlog** 8](#_Toc58502462)

[**3. Kết thúc một sprint** 9](#_Toc58502463)

[**4. Tiến độ công việc** 9](#_Toc58502464)

[**II. DIAGRAMS** 13](#_Toc58502465)

[**1. Activity Diagram - Bản vẽ hoạt động** 13](#_Toc58502466)

[**1.1 Mở đầu** 13](#_Toc58502467)

[**1.2 Các thành phần của Dynamic Model** 13](#_Toc58502468)

[**1.3 Activity Diagram** 14](#_Toc58502469)

[**1.4 Cách xây dựng Activity Diagram** 17](#_Toc58502470)

[**1.5 Mục đích sử dụng bản vẽ Activity Diagram** 17](#_Toc58502471)

[**1.6 Lợi ích của bản vẽ Activity Diagram** 17](#_Toc58502472)

[**2. Deployment Diagram – Bản Vẽ Triển Khai** 18](#_Toc58502473)

[**2.1 Những ký hiệu trong bản vẽ triển khai** 18](#_Toc58502474)

[**2.2 Cách vẽ một sơ đồ triển khai** 19](#_Toc58502475)

[**2.3 Mục đích của sơ đồ triển khai** 21](#_Toc58502476)

[**2.4 Áp dụng Deployment Diagram vào source code php-training** 21](#_Toc58502477)

[**3. Sequence Diagram - Bản vẽ tuần tự** 24](#_Toc58502478)

[**3.1 Khái niệm** 24](#_Toc58502479)

[**3.2 Các thành phần của Sequence Diagram** 25](#_Toc58502480)

[**3.3 Xây dựng Sequence Diagram** 25](#_Toc58502481)

[**3.4 Ứng dụng Sequence Diagram** 26](#_Toc58502482)

[**4. Use Case Diagram - Bản vẽ Use Case** 27](#_Toc58502483)

[**4.1 Khái niệm** 27](#_Toc58502484)

[**4.2 Các thành phần trong bản vẽ Use Case** 27](#_Toc58502485)

[**4.2.1 Actor** 28](#_Toc58502486)

[**4.2.2 Use Case** 28](#_Toc58502487)

[**4.2.3 Relationship (Quan hệ)** 28](#_Toc58502488)

[**4.2.4 System Boundary** 31](#_Toc58502489)

[**4.3 Đặc tả Use Case** 32](#_Toc58502490)

[**4.4 Mục đích sử dụng Use Case Diagram** 32](#_Toc58502491)

[**4.5 Các bước xây dựng Use Case Diagram** 33](#_Toc58502492)

[**4.6 Áp dụng Use Case Diagram vào extension AutoCheck và CheckAuto** 33](#_Toc58502493)

[**5. Class Diagram - Bản vẽ về Class (lớp)** 36](#_Toc58502494)

[**5.1 Khái niệm** 36](#_Toc58502495)

[**5.2 Các thành phần trong bản vẽ Class** 36](#_Toc58502496)

[**5.3 Relationship (Quan hệ)** 37](#_Toc58502497)

[**5.4 Đặc tả Class** 40](#_Toc58502498)

[**5.5 Mục đích sử dụng bản vẽ Class** 40](#_Toc58502499)

[**5.6 Cách xây dựng bản vẽ Class** 40](#_Toc58502500)

[**5.7 Áp dụng vào extension Auto Login** 41](#_Toc58502501)

[**5.8 Áp dụng Class Diagram vào Extension AutoCheck và CheckAuto** 44](#_Toc58502502)

[**6. Package Diagram - Sơ đồ gói** 46](#_Toc58502503)

[**6.1 Khái niệm** 46](#_Toc58502504)

[**6.2 Mục đích của Sơ đồ gói** 46](#_Toc58502505)

[**6.3 Các thành phần trong bản vẽ Package Diagram** 46](#_Toc58502506)

[**6.4 Các khái niệm cơ bản về Package Diagram** 47](#_Toc58502507)

[**6.5 Ví dụ áp dụng** 49](#_Toc58502508)

[**7. Object Diagram - Sơ đồ đối tượng** 51](#_Toc58502509)

[**7.1 Khái niệm** 51](#_Toc58502510)

[**7.2 Mục đích sơ đồ đối tượng** 51](#_Toc58502511)

[**7.3 So sánh giữa Class Diagram và Object Diagram** 52](#_Toc58502512)

[**7.4 Áp dụng object diagram vào source code php-training** 52](#_Toc58502513)

[**8. State Diagram - Sơ đồ trạng thái** 55](#_Toc58502514)

[**8.1 Khái niệm** 55](#_Toc58502515)

[**8.2 Các thành phần trong State Diagram** 56](#_Toc58502516)

[**8.3 Mục đích sơ đồ trạng thái** 56](#_Toc58502517)

[**8.4 Áp dụng State Diagram vào source code php-training** 57](#_Toc58502518)

[**9. Collaboration Diagram - Sơ đồ cộng sinh** 59](#_Toc58502519)

[**9.1 Khái niệm** 59](#_Toc58502520)

[**9.2 Mục đích sơ đồ cộng sinh** 59](#_Toc58502521)

[**9.3 Áp dụng Collaboration Diagram vào source code php-training** 60](#_Toc58502522)

[**10. Component Diagram - Sơ đồ thành phần** 62](#_Toc58502523)

[**10.1 khái niệm** 62](#_Toc58502524)

[**10.2 Mục đích sơ đồ Component Diagram** 62](#_Toc58502525)

[**10.3 Áp dụng Component Diagram vào extension AutoCheck và CheckAuto** 63](#_Toc58502526)

[**10.4 Áp dụng Component Diagram vào source code php-training** 64](#_Toc58502527)

[**11. Interaction Diagram - Biểu đồ tương tác** 66](#_Toc58502528)

[**11.1 Khái niệm** 66](#_Toc58502529)

[**11.2 Mục đích sơ đồ Interaction Diagram** 67](#_Toc58502530)

[**12. Communication Diagram - Biểu đồ giao tiếp** 68](#_Toc58502531)

[**12.1 Khái niệm** 68](#_Toc58502532)

[**12.2 Mục đích sơ đồ Communication Diagram** 68](#_Toc58502533)

[**12.3 Các thành phần trong Communication Diagram** 68](#_Toc58502534)

[**12.4 Áp dụng Communication Diagram vào extension AutoCheck và CheckAuto** 69](#_Toc58502535)

[**III. TÀI LIỆU THAM KHẢO** 72](#_Toc58502536)

**MỤC LỤC HÌNH**

[Hình 1: Các thành phần của mô hình Dynamic Model 13](#_Toc58240242)

[Hình 2: Ví dụ về Activity Diagram của hoạt động rút tiền từ ATM 14](#_Toc58240243)

[Hình 3: Ký hiệu về Swimlane 15](#_Toc58240244)

[Hình 4: Ký hiệu nút Start 15](#_Toc58240245)

[Hình 5: Ký hiệu nút End 15](#_Toc58240246)

[Hình 6: Ký hiệu Activity 15](#_Toc58240247)

[Hình 7: Ký hiệu Branch 16](#_Toc58240248)

[Hình 8: Ký hiệu Fork 16](#_Toc58240249)

[Hình 9: Ký hiệu Join 16](#_Toc58240250)

[Hình 10: Ký hiệu của node 18](#_Toc58240251)

[Hình 11: Ký hiệu của Artifacts 18](#_Toc58240252)

[Hình 12: Ký hiệu quan hệ Association 19](#_Toc58240253)

[Hình 13: Ký hiệu quan hệ Dependence 19](#_Toc58240254)

[Hình 14: Ký hiệu về một device 19](#_Toc58240255)

[Hình 15: Ví dụ về Deployment Diagram với một người dùng Internet thông thường 20](#_Toc58240256)

[Hình 16: Deployment Diagram source code php-training 23](#_Toc58240257)

[Hình 17: Ví dụ Sequence Diagram cho hoạt động rút tiền ở ATM 24](#_Toc58240258)

[Hình 18: Ký hiệu về đối tượng trong bản vẽ sequence Diagram 25](#_Toc58240259)

[Hình 19: Ký hiệu về Stimulus trong bản vẽ Sequence Diagram 25](#_Toc58240260)

[Hình 20: Bản vẽ Use Case về ứng dụng ATM 26](#_Toc58240261)

[Hình 21: Ký hiệu của Actor 27](#_Toc58240262)

[Hình 22: Ký hiệu của Actor 27](#_Toc58240263)

[Hình 23: Ký hiệu về Use Case 27](#_Toc58240264)

[Hình 24: Ký hiệu quan hệ Association 28](#_Toc58240265)

[Hình 25: Ví dụ thể hiện Actor User sử dụng Use Case Login 28](#_Toc58240266)

[Hình 26: Ký hiệu quan hệ Generalization 28](#_Toc58240267)

[Hình 27: Ví dụ Actor User thừa kế toàn bộ quyền của Actor Guest 29](#_Toc58240268)

[Hình 28: Ký hiệu quan hệ Include 29](#_Toc58240269)

[Hình 29: Ví dụ về quan hệ Include giữa các Use Case 29](#_Toc58240270)

[Hình 30: Ký hiệu quan hệ Extend 29](#_Toc58240271)

[Hình 31: Ví dụ về quan hệ Extend giữa các Use Case 30](#_Toc58240272)

[Hình 32: Vùng System Boundary 30](#_Toc58240273)

[Hình 33: Use Case diagram extension 34](#_Toc58240274)

[Hình 34: Ví dụ về Class Diagram của ATM 35](#_Toc58240275)

[Hình 35: Ký hiệu về Class 36](#_Toc58240276)

[Hình 36: Ví dụ về một Class 36](#_Toc58240277)

[Hình 37: Ví dụ về Association 37](#_Toc58240278)

[Hình 38: Ví dụ về Aggregation 37](#_Toc58240279)

[Hình 39: Ví dụ về Composition 38](#_Toc58240280)

[Hình 40: Ví dụ về Generalization 38](#_Toc58240281)

[Hình 41: Các nguồn thông tin có thể tìm Class dự kiến 40](#_Toc58240282)

[Hình 42: Bản vẽ Class dự kiến 41](#_Toc58240283)

[Hình 43: Xác định thuộc tính và quan hệ cho các lớp 42](#_Toc58240284)

[Hình 44: Bổ sung phương thức cho các lớp 42](#_Toc58240285)

[Hình 45: Class diagram Extension AutoCheck và CheckAuto 44](#_Toc58240286)

[Hình 46: Ví dụ về Package Diagram 46](#_Toc58240287)

[Hình 47: Các ký hiệu hiển thị các gói 47](#_Toc58240288)

[Hình 48: Ví dụ về import package 47](#_Toc58240289)

[Hình 49: Ví dụ về Access Package 47](#_Toc58240290)

[Hình 50: Các package được xác định 48](#_Toc58240291)

[Hình 51: Hai mô-đun đang truy cập lẫn nhau 48](#_Toc58240292)

[Hình 52: Điều hướng thông tin vận chuyển 49](#_Toc58240293)

[Hình 53: Package xử lý đơn hàng 49](#_Toc58240294)

[Hình 54: Object diagram 50](#_Toc58240295)

[Hình 55: Class diagram 51](#_Toc58240296)

[Hình 56: Object diagram 51](#_Toc58240297)

[Hình 57: Object diagram áp dụng vào source code php-training 53](#_Toc58240298)

[Hình 58: State diagram 54](#_Toc58240299)

[Hình 59: Sơ đồ state diagram hoàn chỉnh 55](#_Toc58240300)

[Hình 60: Ký hiệu chi tiết UML 55](#_Toc58240301)

[Hình 61: State diagram áp dụng vào source code php-training 57](#_Toc58240302)

[Hình 62: Collaboration diagram 58](#_Toc58240303)

[Hình 63: State diagram áp dụng vào source code php-training 60](#_Toc58240304)

[Hình 64: Ký hiệu của component 61](#_Toc58240305)

[Hình 65: Ký hiệu của component dependency 61](#_Toc58240306)

[Hình 66: Component diagram 61](#_Toc58240307)

[Hình 67: Component diagram for extension 63](#_Toc58240308)

[Hình 68: Component diagram for source code php-training 64](#_Toc58240309)

[Hình 69: Biểu đồ tuần tự 65](#_Toc58240310)

[Hình 70: Biểu đồ giao tiếp 65](#_Toc58240311)

[Hình 71: Biểu đồ giao tiếp 67](#_Toc58240312)

[Hình 72: Liên kết giữa actor và object 68](#_Toc58240313)

[Hình 73: Liên kết giữa 2 object 68](#_Toc58240314)

[Hình 74: Mô tả tương tác giữa các đối tượng 68](#_Toc58240315)

[Hình 75: Communication Diagram for extensions 70](#_Toc58240316)

# **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**

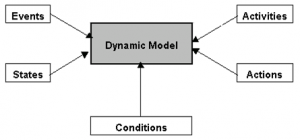
Theo UML thì hệ thống có thể được mô tả theo 2 mô hình tĩnh (Static Model) và mô hình động (Dynamic Model).

**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 1: 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.

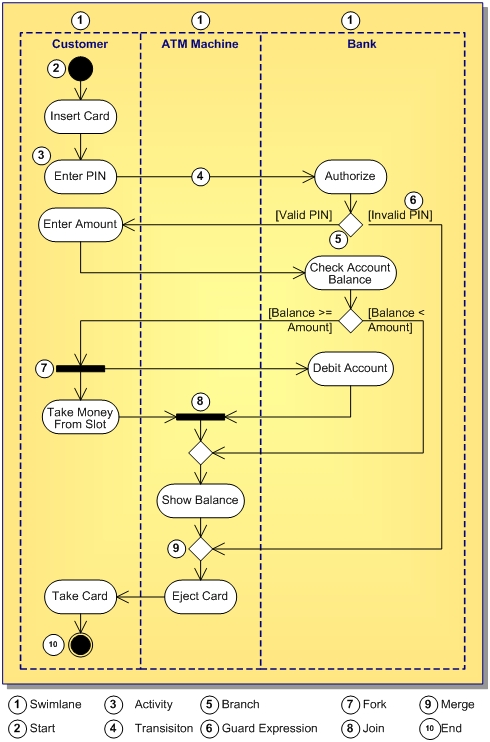
**States**: là các trạng thái của một đối tượng trong hệ thống, được mô tả bằng giá trị của một hoặc nhiều thuộc tính.

**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.



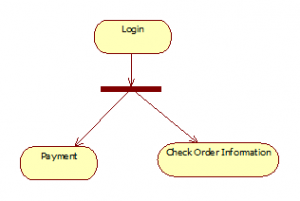
Hình 6: Ký hiệu Activity

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

Branch

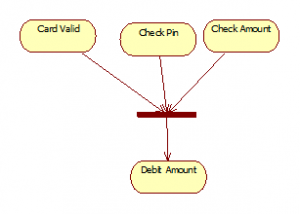
Hình 7: Ký hiệu Branch

**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 Mục đích 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.

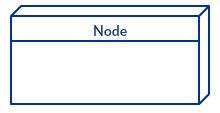
## **2. Deployment Diagram – Bản Vẽ Triển Khai**

**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 10: 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 11: 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 12: Ký hiệu quan hệ Association

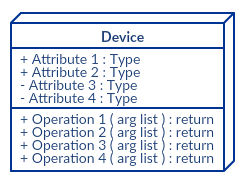
Component 2

Hình 13: 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 14: 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:**

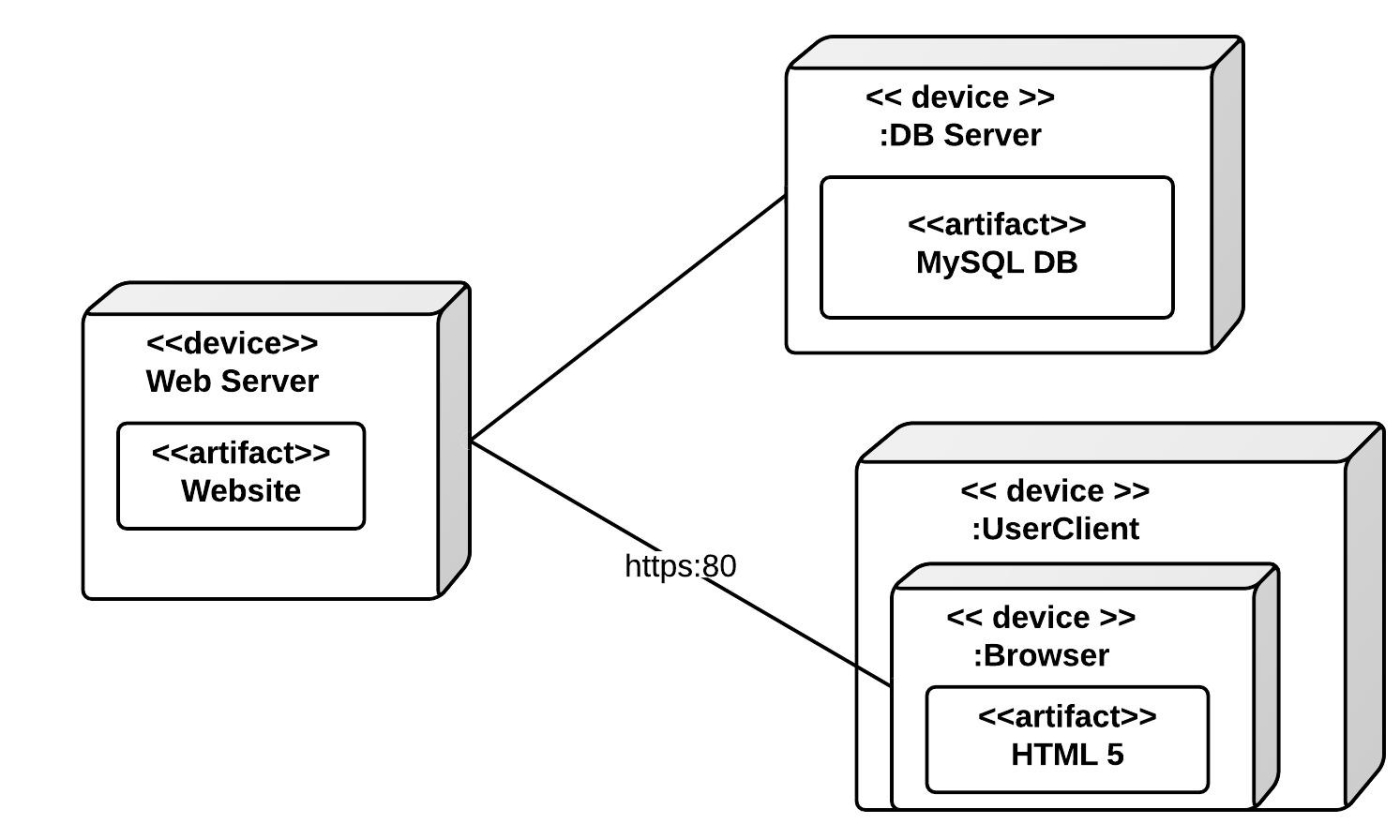
Xác định những yếu tố khác như các thành phần component, các đối tượng, các hoạt động mà bạn cần thêm để hoàn thành sơ đồ.

**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 15: 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.

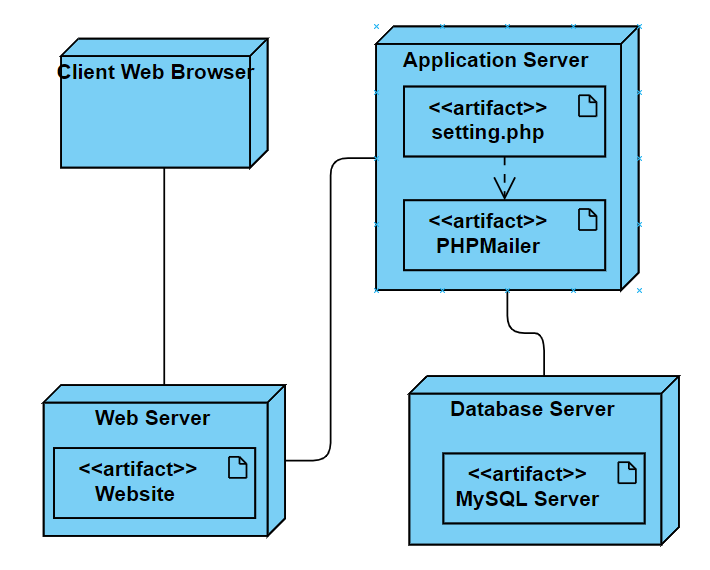
**Bước 2: Xác định mối quan hệ giữa các node**

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:



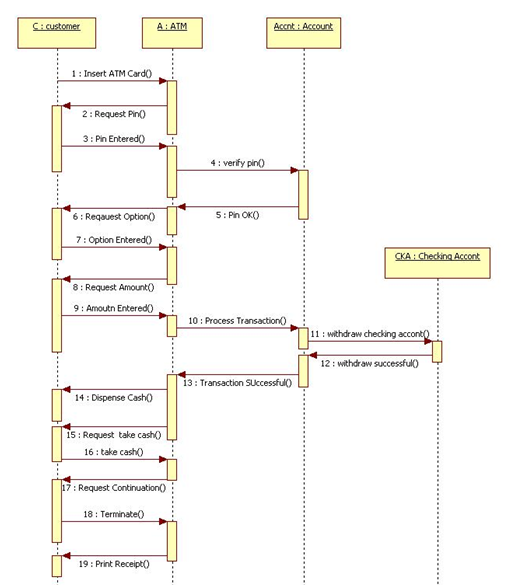
Hình 16: Deployment Diagram source code php-training

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.

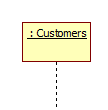


Hình 17: Ví dụ Sequence Diagram cho hoạt động rút tiền ở ATM

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

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

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



Hình 18: Ký hiệu về đối tượng trong bản vẽ sequence Diagram

**Stimulus (message):** thể hiện thông điệp từ một đối tượng này tương tác với một đối tượng khác.

seq-Notation2

Hình 19: Ký hiệu về Stimulus trong bản vẽ Sequence Diagram

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

### **3.3 Xây dựng Sequence Diagram**

**Bước 1**: Xác định chức năng cần thiết kế. Dựa vào Use Case Diagram để xác định xem chức năng nào cần thiết kế.

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

**Bước 3**: Đối chiếu với Class Diagram để xác định lớp trong hệ thống tham gia vào chức năng.

**Bước 4**: Vẽ Sequence Diagram.

**Bước 5**: Cập nhật lại bản vẽ Class Diagram.

### **3.4 Ứng dụng Sequence Diagram**

- Thiết kế các chức năng.

- Kiểm chứng và bổ sung method cho các Class.

- Sử dụng trong việc coding các chức năng.

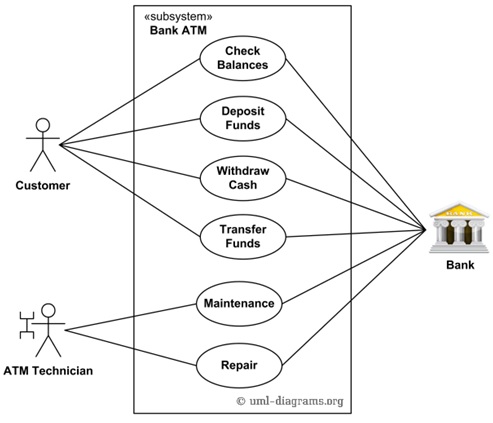
## **4. Use Case Diagram - Bản vẽ Use Case**

### **4.1 Khái niệm**

Use Case được mô tả trong ngôn ngữ UML qua biểu đồ Use Case (Use Case Diagram). Một biểu 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 20: Bản vẽ Use Case về ứng dụng ATM

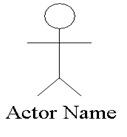
Nhìn bản vẽ này chúng ta thấy có hai người dùng là Customer và ATM Technician và một đối tượng sử dụng hệ thống là Bank. Bên cạnh đó nó mô tả các chức năng của hệ thống và người dùng nào dùng chức năng gì.

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**

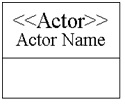
Actor được dùng để chỉ người sử dụng hoặc một đối tượng nào đó bên ngoài tương tác với hệ thống.

Actor được biểu diễn như sau:



Hình 21: Ký hiệu của Actor

Hoặc



Hình 22: 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 23: 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

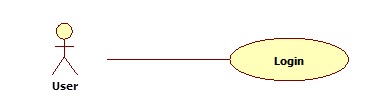
- Extend

**+ Quan hệ Association:**

Association

Hình 24: 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.



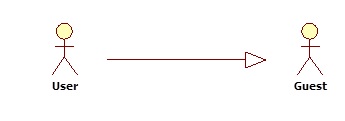
Hình 25: Ví dụ thể hiện Actor User sử dụng Use Case Login

**+ Quan hệ Generalization:**

Generalization

Hình 26: 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.



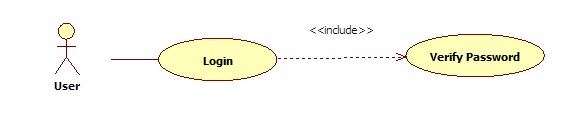
Hình 27: Ví dụ Actor User thừa kế toàn bộ quyền của Actor Guest

**+ Quan hệ Include:**



Hình 28: 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 29: Ví dụ về quan hệ Include giữa các Use Case

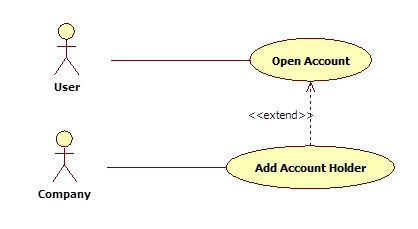
Use Case “Verify Password” có thể gộp chung vào Use Case Login nhưng ở đây chúng ta tách ra để cho các Use Case khác sử dụng hoặc để module hóa cho dễ hiểu, dễ cài đặt.

**+ Quan hệ Extend**



Hình 30: 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.

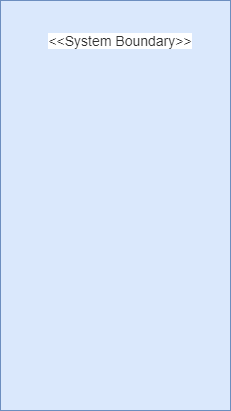


Hình 31: Ví dụ về quan hệ Extend giữa các Use Case

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 32: 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.

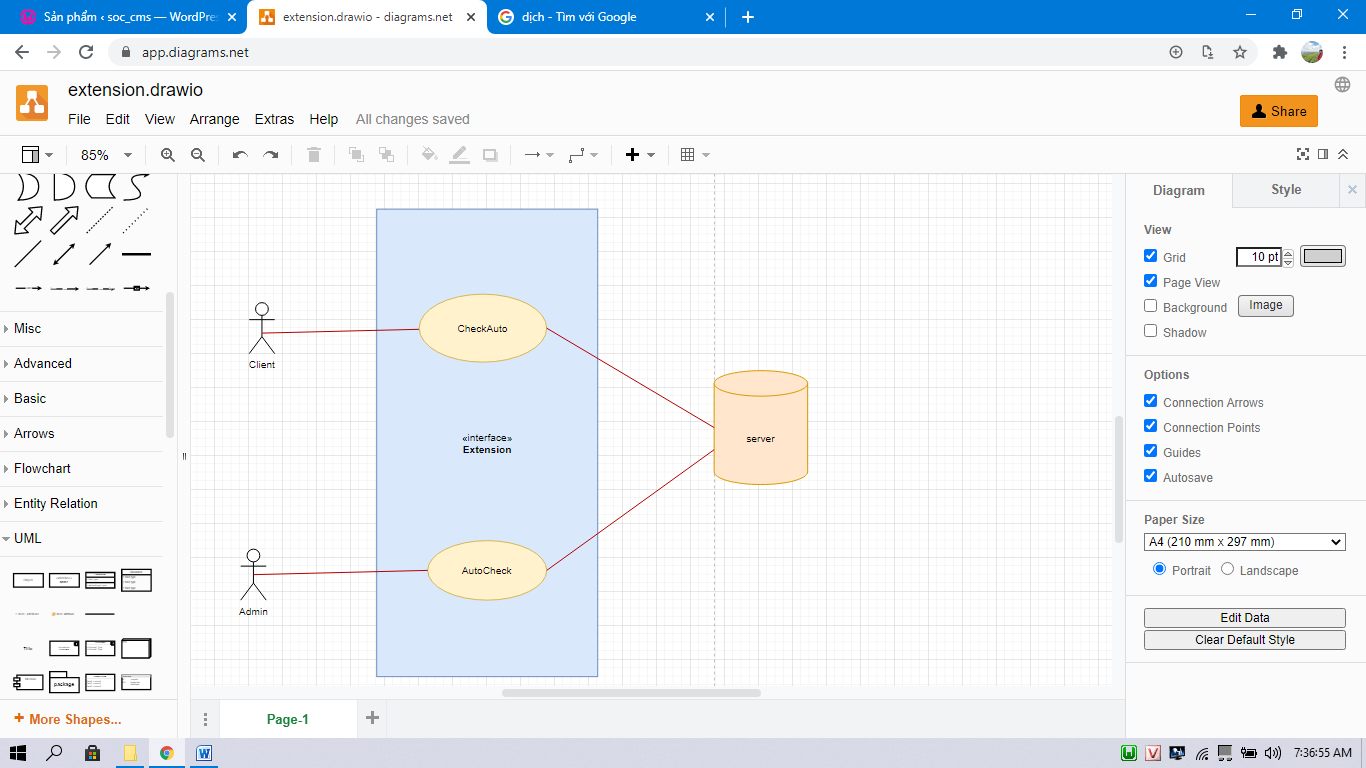
**Bước 3:** Xác định các quan hệ

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.



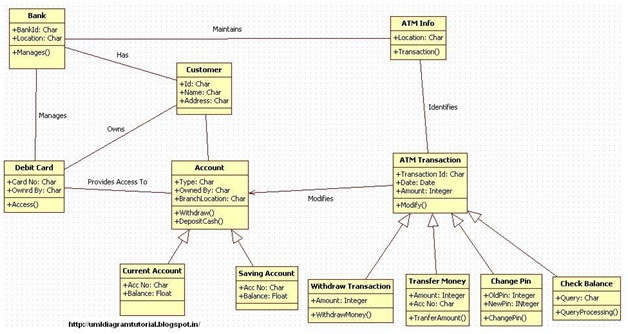
Hình 33: Use Case diagram extension

## **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 34: Ví dụ về Class Diagram của ATM

**Classes (Các lớp)**

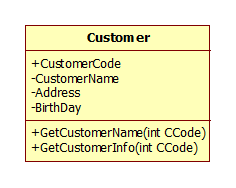
Class là thành phần chính của bản vẽ Class Diagram. Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống.



Hình 35: 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 36: 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 37: 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 38: 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 39: 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 40: 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

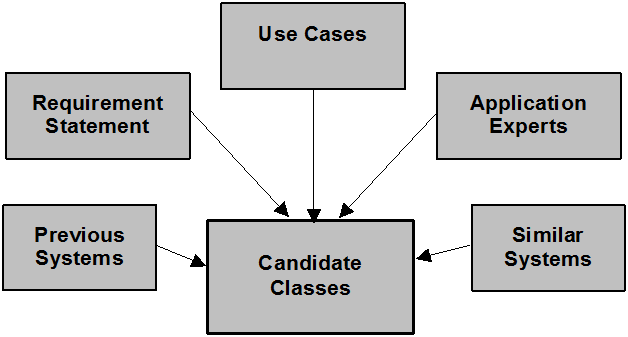
- Sử dụng để phân tích chi tiết các chức năng (Sequence Diagram, State Diagram v.v…)

- 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 41: 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.

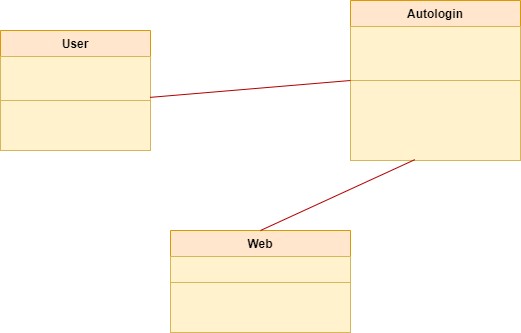
### **5.7 Áp dụng vào extension Auto Login**

**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:



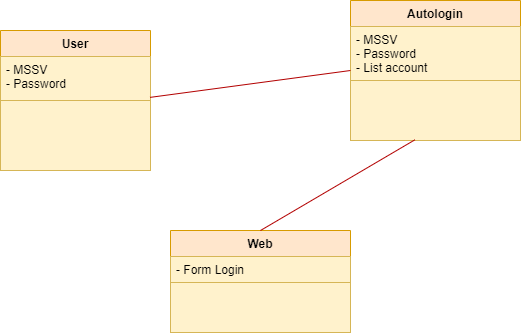
Hình 42: Bản vẽ Class dự kiến

**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

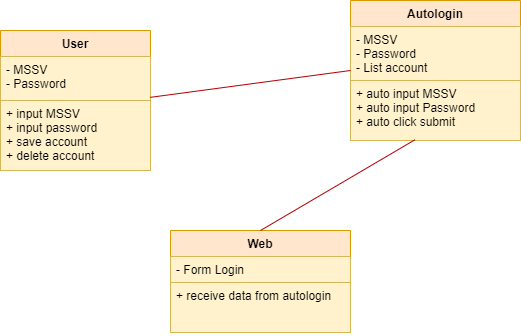
- Web: có thuộc tính Form Login



Hình 43: 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 44: 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.

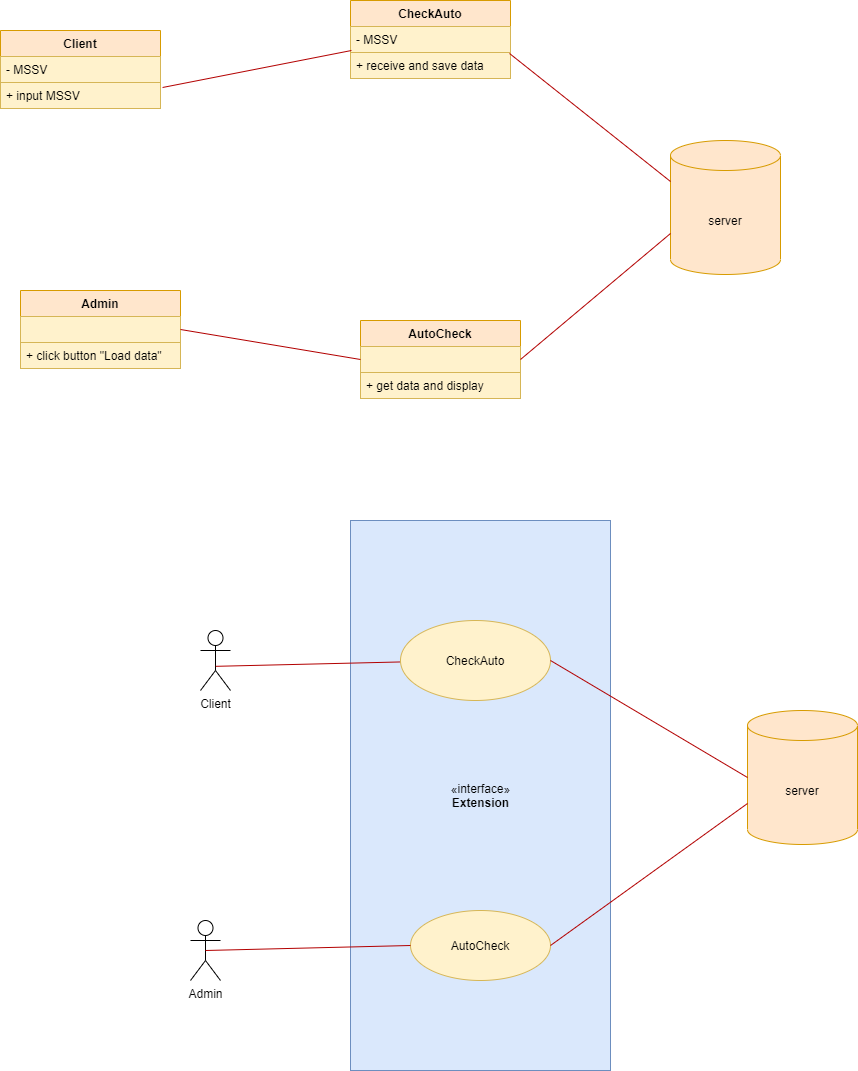
**Bước 2: Xác định mối quan hệ giữa các class**

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 45: 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.

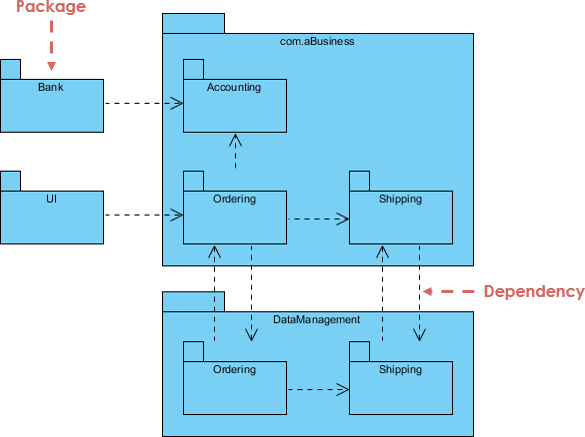
Sơ đồ gói có thể được sử dụng để đơn giản hóa các sơ đồ lớp phức tạp, nó có thể nhóm các lớp thành các gói.

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.

### **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 46: 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 chấm là phụ thuộc.

- Một gói phụ thuộc vào gói khác nếu những thay đổi trong gói kia có thể buộc những thay đổi trong gói đầu tiên.

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

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

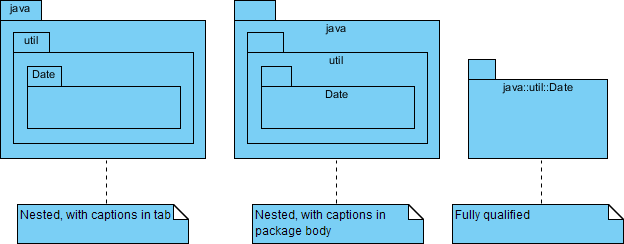
Có một số ràng buộc trong khi sử dụng sơ đồ gói, chúng như sau:

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

- Packages có thể bao gồm toàn bộ sơ đồ, tên của các thành phần một mình hoặc không có thành phần nào cả.

- Tên đầy 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:



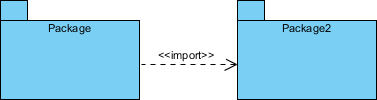
Hình 47: Các ký hiệu hiển thị các gói

**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

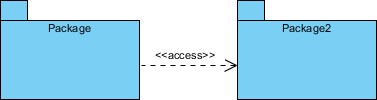
<<import>> - một gói nhập chức năng của gói khác.



Hình 48: 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 49: Ví dụ về Access Package

### **6.5 Ví dụ áp dụng**

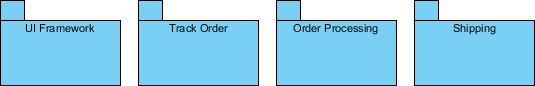
Ví dụ về sơ đồ gói - Hệ thống xử lý đơn hàng:

Dựa trên mô tả dự án, trước tiên chúng ta nên xác định các gói trong hệ thống và sau đó liên kết chúng với nhau theo mối quan hệ:

Xác định các gói của hệ thống:

- Có một mô-đun đặt hàng theo dõi, nó phải nói chuyện với mô-đun khác để biết về chi tiết đơn hàng, chúng tôi gọi nó là "Chi tiết đơn hàng".

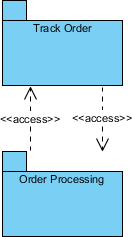
- Tiếp theo sau khi tìm nạp Chi tiết đơn hàng, nó phải biết về chi tiết vận chuyển, hãy để chúng tôi gọi đó là "Vận chuyển".



Hình 50: Các package được xác định

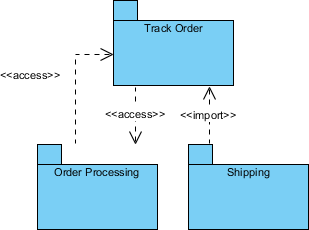
Xác định các phụ thuộc trong Hệ thống:

- Theo dõi đơn đặt hàng phải lấy chi tiết đơn hàng từ "Chi tiết đơn hàng" và "Chi tiết đơn hàng" phải biết thông tin theo dõi do khách hàng cung cấp. Hai mô-đun đang truy cập lẫn nhau đủ <<access>> phụ thuộc kép.



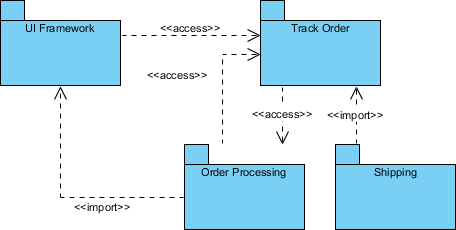
Hình 51: Hai mô-đun đang truy cập lẫn nhau

Để biết thông tin vận chuyển, "Vận chuyển" có thể nhập "Theo dõi Đơn hàng" để điều hướng dễ dàng hơn.



Hình 52: Điều hướng thông tin vận chuyển

Cuối cùng, sự phụ thuộc của Đơn hàng theo dõi vào Khung giao diện người dùng cũng được ánh xạ để hoàn thành Sơ đồ gói cho hệ thống con Xử lý đơn hàng.

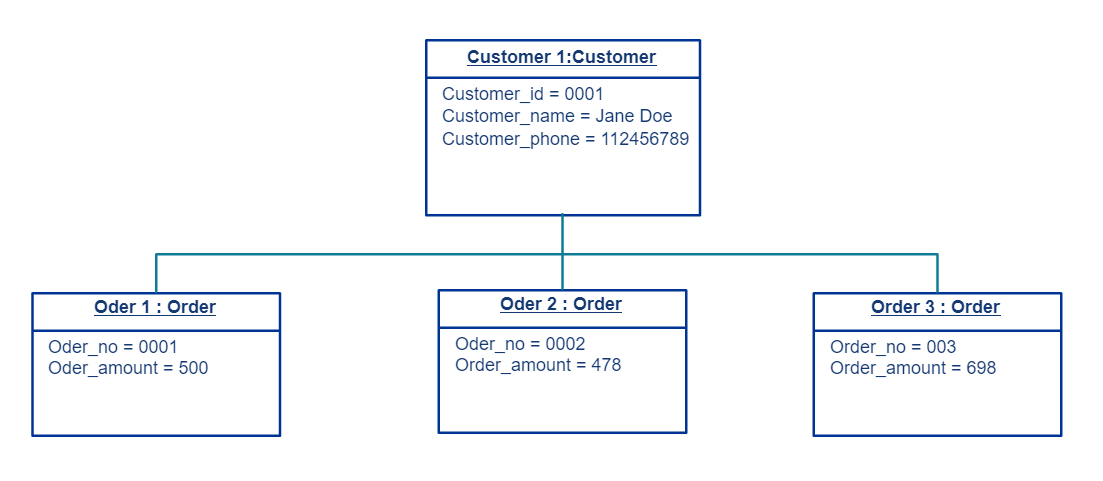


Hình 53: Package xử lý đơn hàng

## **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.



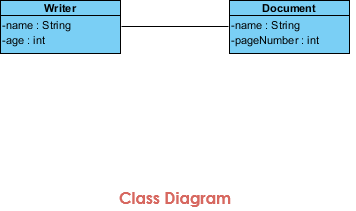
Hình 54: Object diagram

### **7.2 Mục đích sơ đồ đối tượng**

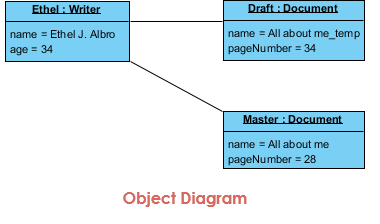
Sự khác biệt giữa sơ đồ lớp với sơ đồ đối tượng là: Sơ đồ lớp thì thể hiện một mô hình trừu tượng bao gồm các lớp và mối quan hệ giữa chúng. Tuy nhiên, sơ đồ đối tượng lại biểu diễn cho một cá thể tại một thời điểm cụ thể, bản chất cụ thể. Điều này có nghĩa sơ đồ đối tượng sẽ gần với hoạt động hệ thống 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 biểu 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 55: Class diagram



Hình 56: 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.**

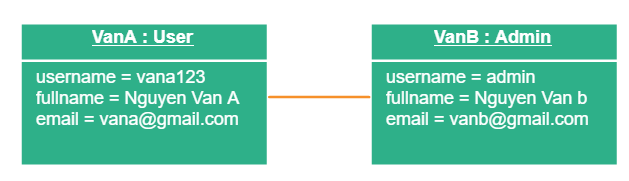
Với VanA thuộc đối tượng User với thuộc tính:

* 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:

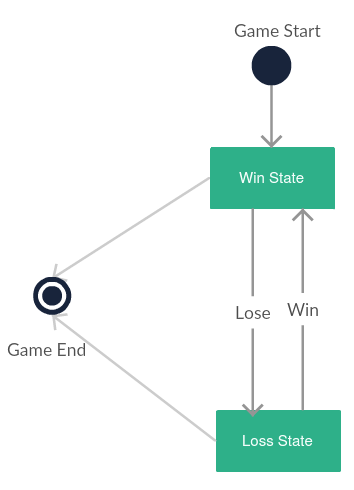


Hình 57: Object diagram áp dụng vào source code php-training

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

### **8.1 Khái niệm**

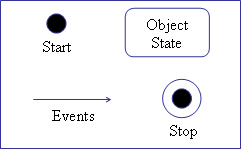
Đây là một biểu đồ trạng thái được dùng bổ sung cho **Class Diagram**. State Diagram chỉ ra tất cả các trạng thái mà đối tượng của Class có thể có và các event gây ra sự thay đổi của trạng thái. Event làm thay đổi của trạng thái xảy ra khi một đối tượng tự gửi thông tin dữ liệu đến cho State.



Hình 58: State diagram

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

State Diagram gồm:



Hình 60: 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**.

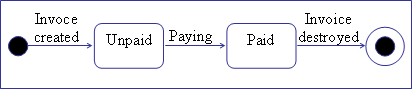
Những sự kiện gây nên sự biến đổi từ một trạng thái này sang một trạng thái khác.

### **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 59: 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:

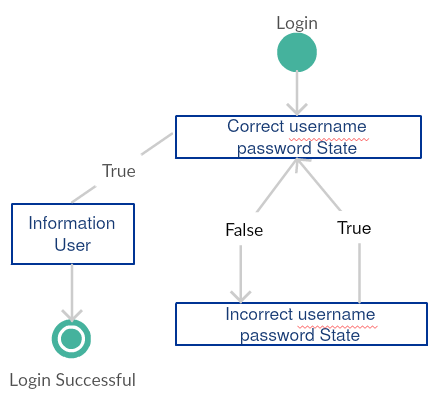
Trạng thái 1:

* 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 61: 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:**

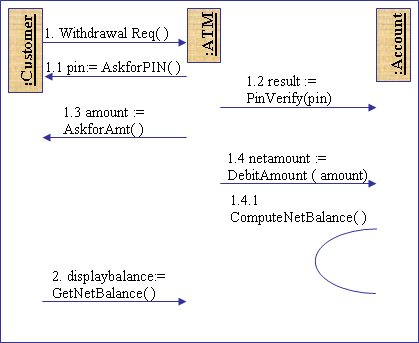
Tiến hành thực hiện chức năng đăng nhập khi người dùng nhập sai username, password sẽ yêu cầu người dùng nhập lại **username**, **password** đúng với **username**, **password** đăng ký, sau khi người dùng nhập đúng quay lại trường hợp 1 và trả về người dùng trang thông tin cá nhân kèm thông báo đăng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

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

### **9.1 Khái niệm**

Collaboration Diagram hay còn được gọi là biểu 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 biểu đồ cộng tác một hoặc nhiều đối tượng được biểu diễn bằng ký hiệu lớp. Thứ tự các event trong biểu 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 biểu đồ tuần tự (State Diagram).



Hình 62: Collaboration diagram

### **9.2 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

### **9.3 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.

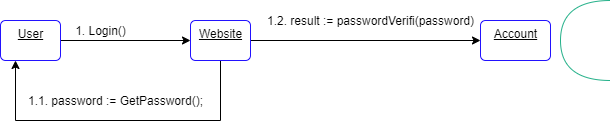
Khi người dùng muốn thay đổi mật khẩu tại trang quản lý thông tin cá nhân, người dùng cần phải nhập lại đúng mật khẩu hiện tại và xác nhận lại mật khẩu mới lần nữa thì mới thành công.

**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**

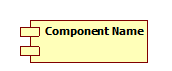


Hình 63: State diagram áp dụng vào source code php-training

## **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 64: Ký hiệu của component

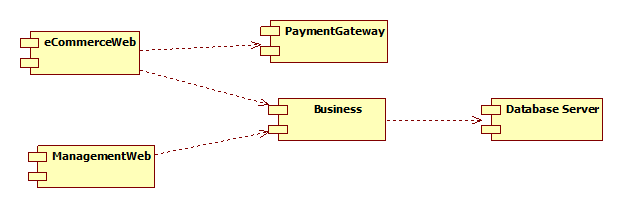
Ngoài ra còn có Component Dependency thể hiện quan hệ giữa các thành phần lại với nhau, các thành phần của phần mềm luôn luôn cần sử dụng một hoặc nhiều chức năng ở các thành phần khác trong hệ thống nên việc quan hệ Component Dependency được sử dụng thường xuyên.



Hình 65: 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 66: 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 Áp dụng Component Diagram vào extension AutoCheck và CheckAuto**

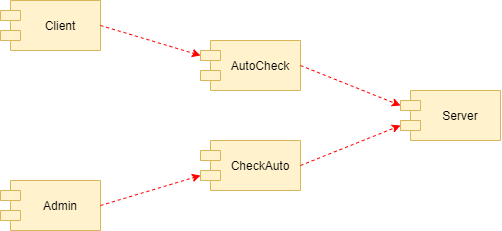
**Phân tích các hệ thống cho một Component**:

- 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 67: Component diagram for extension

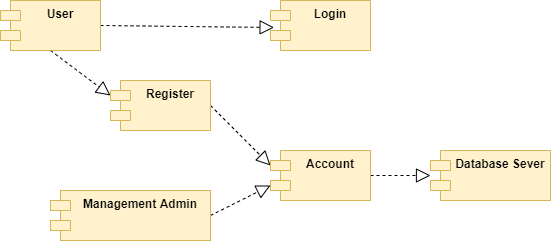
### **10.4 Áp dụng Component Diagram vào source code php-training**

**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.



Hình 68: Component diagram for source code php-training

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

### **11.1 Khái niệm**

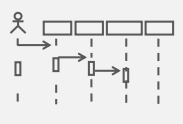
Interaction Diagram là sơ đồ được sử dụng để mô tả một số loại tương tác giữa các yếu tố khác nhau trong mô hình. Tương tác này là một phần của hành vi năng động của hệ thống.

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 biểu đồ tương tác:

− Biểu đồ tuần tự (Sequence diagram):

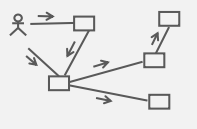
Một cách nhìn hướng về trình tự thời gian tương tác giữa các đối tượng.



Hình 69: Biểu đồ tuần tự

− Biểu 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 70: Biểu đồ giao tiếp

### **11.2 Mục đích sơ đồ Interaction Diagram**

Mục đích của biểu 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 sơ đồ 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 sơ đồ 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 - Biểu đồ giao tiếp**

### **12.1 Khái niệm**

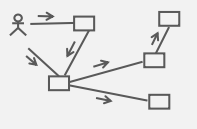
Là biểu đồ tương tác tập trung vào tổ chức các đối tượng tham gia tương tác.

Biểu 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 71: Biểu 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**

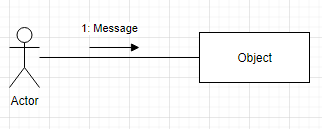
Lược đồ cộng tác có 4 thành phần chính:

- Object: Đối tượng tham gia quá trình tương tác giữa người dùng và hệ thống.

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

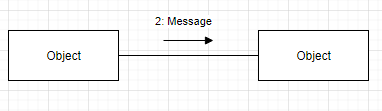
- Instance Link: Biểu diễn liên kết giữa 2 đối tượng.

+ Liên kết giữa actor và object:



Hình 72: Liên kết giữa actor và object

+ Liên kết giữa 2 object:

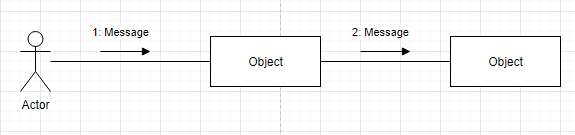


Hình 73: Liên kết giữa 2 object

- Message:

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

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



Hình 74: Mô tả tương tác giữa các đối tượng

### **12.4 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

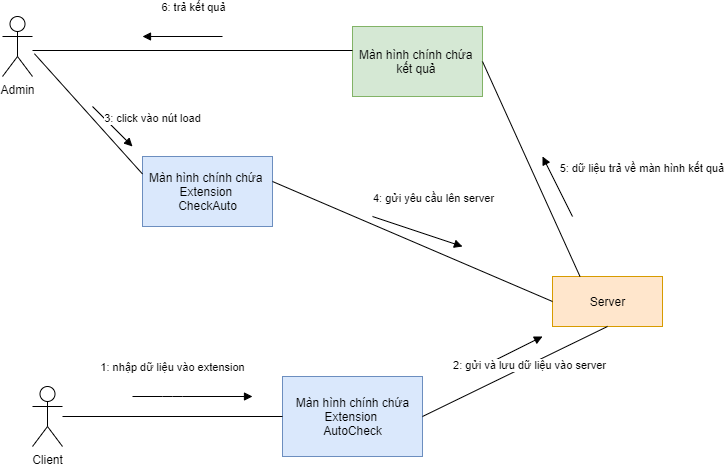
**Xác định các thành phần:**

Actor: gồm client và admin

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

Các liên kết: liên kết giữa actor và object, admin và object.

Message: gồm 6 message.



Hình 75: Communication Diagram for extensions

# **III. TÀI LIỆU THAM KHẢO**

* <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>