

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



ĐỒ ÁN CHUYÊN NGÀNH  
HƯỚNG CÔNG NGHỆ PHẦN MỀM (CO4029)

---

Báo cáo Đồ án – version 0.2

GRADING SYSTEM

---

Giảng viên hướng dẫn: Lê Đình Thuận

Sinh viên thực hiện: Lê Minh Châu 2010947

Tp. Hồ Chí Minh, Tháng 3/2024

# Mục lục

<b>1 Tổng quan dự án</b>	<b>3</b>
1.1 Bối cảnh . . . . .	3
1.2 Ý tưởng . . . . .	3
1.3 Đánh giá . . . . .	4
1.4 Phương hướng phát triển . . . . .	4
<b>2 Công cụ liên quan</b>	<b>6</b>
2.1 Git, Github & Github Action . . . . .	6
2.2 Công cụ chống gian lận . . . . .	7
2.3 Công cụ trong hệ thống Grading System . . . . .	8
<b>3 User story</b>	<b>9</b>
3.1 Admin . . . . .	9
3.2 Người giảng dạy . . . . .	9
3.3 Học viên . . . . .	9
<b>4 Functional &amp; Non-Functional Requirements</b>	<b>10</b>
4.1 Functional Requirements . . . . .	10
4.2 Non-Functional Requirements . . . . .	11
<b>5 Activity</b>	<b>12</b>
5.1 Student host activity . . . . .	12
5.1.1 Studen host . . . . .	12
5.1.2 Student host activity diagram . . . . .	13
5.2 Teacher host activity . . . . .	14
5.2.1 Teacher host . . . . .	14
5.2.2 Teacher host activity diagram . . . . .	15
5.3 Grading and checking activity . . . . .	16
5.3.1 Grading and Checking . . . . .	16
5.3.2 Grading and Checking activity diagram . . . . .	16

<b>6</b>	<b>Usecases</b>	<b>17</b>
6.1	System diagram . . . . .	17
6.2	Usecase senario . . . . .	19
6.2.1	Manage classes . . . . .	19
6.2.2	. . . . .	19
6.2.3	. . . . .	19
6.2.4	. . . . .	19
6.2.5	. . . . .	19
6.2.6	. . . . .	19
6.2.7	. . . . .	19
6.2.8	. . . . .	19
6.2.9	. . . . .	19
6.2.10	. . . . .	19
6.2.11	. . . . .	19
6.2.12	. . . . .	19
6.2.13	. . . . .	19
6.2.14	. . . . .	19
6.2.15	. . . . .	19

# 1 Tổng quan dự án

## 1.1 Bối cảnh

Gian lận trong thi cử luôn là vấn đề được chú ý. Trong hội thảo "Đánh giá hoạt động học tập trong dạy học trực tuyến: Từ thiết kế đến triển khai" do Ban Đại Học, ĐHQG-HCM tổ chức ngày 26/08/2021, Theo TS Nguyễn Tấn Đại, trước khi thi trực tuyến, ProctorU - công ty cung cấp dịch vụ theo dõi người dự thi và kiểm tra ID, đã bắt quả tang sinh viên gian lận chỉ dưới 1% trong số 340.000 bài thi từ tháng 1-3/2020. Khi áp dụng thi trực tuyến có giám thị theo dõi từ xa, tỷ lệ sinh viên gian lận đã tăng trên 8% trong 1,3 triệu bài thi từ tháng 4-6/2020.

Đối với ngành khoa học máy tính, gian lận có thể là copy code của một ai đó, hoặc sử dụng công cụ (tiêu biểu có thể kể đến AI) trong các bài kiểm tra. Điều này sẽ làm giảm chất lượng giáo dục cũng như chất lượng học viên, và nhiều hệ lụy khác, không chỉ ảnh hưởng cá nhân đó mà cả tổ chức giảng dạy.

Mục tiêu của dự án:

- Giảm thiểu tình trạng gian lận.
- Tăng chất lượng đầu ra của học viên.
- Có thể sử dụng bởi nhiều cơ sở giảng dạy khác nhau
- Có thể sử dụng cho một server cỡ nhỏ

## 1.2 Ý tưởng

Grading System là hệ thống chấm điểm bài kiểm tra trong ngành khoa học máy tính bằng cách tận dụng Github (Gitlab, ...), đồng thời sử dụng những hệ thống đánh giá gian lận được phát triển và sử dụng bởi các tổ chức tin cậy để đánh giá bài làm của học viên, sau đó sẽ được đánh giá lại bởi giảng viên nhằm xác định chính xác bài làm gian lận và có biện pháp xử lý phù hợp.

Ý tưởng của hệ thống như sau:

1. Giảng viên sử dụng Github tạo repo đề, testcase mẫu, các file workflow tương ứng với các ngôn ngữ được cho phép (có thể tự tạo, hoặc chọn template có sẵn của hệ thống).
2. Học viên fork private repo mà giảng viên đã tạo, chọn branch ngôn ngữ thích hợp, có thể config lại tùy theo nhu cầu cá nhân, giải quyết bài toán và commit lại vào repo của mình.
3. Sau khi học viên commit, Github Action sẽ thực hiện quá trình CI dựa trên file workflow và gửi kết quả về hệ thống.
4. Hệ thống sử dụng công cụ đánh giá gian lận kiểm tra bài làm hợp lệ.
5. Trả về kết quả và thông tin chi tiết (điểm, testcase passed, cheating status) cho giảng viên qua email dưới dạng file csv.

### 1.3 Đánh giá

Thông thường, quy trình hoạt động của các online judge là người dùng sẽ viết code trên text editor, sau khi hoàn thành sẽ gửi bài làm đến server. Server sẽ sử dụng IDE của mình để run bài làm với các testcase được tạo sẵn, sau đó trả về kết quả cho người dùng. Tuy nhiên, với một hệ thống nhỏ, việc run code trên IDE có thể khiến server bị quá tải (tiêu biểu như lỗi Runtime Error sẽ tiêu tốn rất nhiều tài nguyên server nếu như không có/cơ chế chịu lỗi chưa hoàn thiện, hoặc có quá nhiều request cùng lúc).

Đặc điểm của hệ thống là tận dụng khả năng của các hệ thống khác sẵn có, được phát triển qua nhiều năm và được tin tưởng bởi cộng đồng phát triển nhằm nâng cao sự ổn định, chính xác và giảm tải cho server cục bộ, cũng như công sức phát triển hệ thống nhưng vẫn đạt được chất lượng nhất định. Đồng thời tiết kiệm thời gian của giảng viên bằng việc tự động hóa một số công việc trong quá trình chấm điểm bài kiểm tra của học viên.

Một số ưu điểm dựa trên chiến lược phát triển của hệ thống:

- Ổn định, đáng tin cậy  
Hệ thống sử dụng các hệ thống sẵn có như Github, ... để chạy và đánh giá bài làm của học viên. Các hệ thống này thường có quy mô lớn và được cộng đồng tin tưởng. Do đó, có thể đảm bảo được việc hạn chế downtime, đồng thời giảm lỗi từ phía server khi compile bài làm.
- Giảm tải cho server  
Bài làm của học viên được compile và lưu trữ trên Github. Hệ thống không cần tốn chi phí cho những hành động này. Vì vậy, những tài nguyên đó có thể sử dụng để thực hiện các process khác.
- Hỗ trợ đa ngôn ngữ  
Các bài làm có thể sử dụng nhiều loại ngôn ngữ khác nhau, tùy từng học viên. Đối với một số môn học (DSA, ...), ngôn ngữ thường không chiếm vai trò quá quan trọng, do đó, học viên có thể sử dụng ngôn ngữ phù hợp để làm bài sẽ giúp hỗ trợ việc học tốt hơn.
- Tự động hóa  
Hầu hết các quá trình chấm điểm và đánh giá gian lận bài kiểm tra đều sử dụng công cụ tự động hóa, giảm thiểu công sức cần phải bỏ ra của giảng viên.
- Ngoài ra, học viên cũng có thể trau dồi kiến thức về các nền tảng chia sẻ code như Github / Gitlab.

### 1.4 Phương hướng phát triển

Lúc trước, việc gian lận có thể là copy bài làm, hoặc cùng hợp tác để vượt qua kì thi. Các công cụ hiện có thường có thể đánh giá được do đã được phát triển qua nhiều năm để đối phó với tình trạng này.

Tuy nhiên, hiện nay, rất nhiều công cụ khác rất có tiềm năng để vượt qua được những công cụ đánh giá gian lận hiện tại, tiêu biểu là AI. Dù vậy, hệ thống vẫn có thể khả thi do hiện tại, các đoạn code mà AI sinh ra thường không quá khác biệt. Nếu có một vài học viên sử dụng, hệ thống vẫn có thể nhận biết được.

Nhưng tiềm năng của AI đối với việc này hoàn toàn không thể dự đoán được. Một vấn đề phát sinh, nếu AI có thể sinh mã hoàn toàn khác giữa các học viên, mọi chuyện sẽ phức tạp hơn nhiều. Hệ thống lúc này cần phải có công cụ hiệu quả hơn để đối phó với tình huống này. Cũng chính vì vậy, hệ thống được xây dựng theo hướng có thể sử dụng nhiều công cụ đánh giá khác nhau thay vì thiết kế riêng biệt dành riêng cho hệ thống. Đây chính là thách thức lâu dài mà hệ thống cần phải phát triển thêm.

Ngoài ra, hệ thống được thiết kế để sử dụng công cụ đánh giá gian lận của bên thứ ba. Do đó, trong tương lai hệ thống hoàn toàn có thể áp dụng một công cụ khác được thiết kế không phải dành riêng cho ngành khoa học máy tính. Việc mở rộng khả năng xử lý gian lận sang nhiều lĩnh vực khác cũng là hướng phát triển quan trọng của hệ thống này.

## 2 Công cụ liên quan

### 2.1 Git, Github & Github Action

#### *Git*

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System – DVCS), nó là một trong những hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay. Git cung cấp cho mỗi lập trình viên kho lưu trữ (repository), lưu lại tất cả các file trong toàn bộ dự án và ghi lại toàn bộ lịch sử thay đổi của file. Mỗi sự thay đổi được lưu lại sẽ được và thành một version (phiên bản).

#### *Github*

Github là nền tảng lưu trữ cloud-based được thiết kế đặc biệt cho các dự án phần mềm. Github có đầy đủ những tính năng của Git, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

#### *CI/CD*

Continuous Integration / Continuous Deployment (CI/CD) là quá trình tự động quá việc kiểm tra mã nguồn (test), xây dựng (build) và triển khai (deploy) của dự án.

*Continuous Intergration* (CI): là quá trình tự động hóa trong phát triển phần mềm, cho phép các thành viên trong đội kiểm tra và hợp nhất các mã nguồn vào Repository chung. Quá trình CI được diễn ra như sau:

1. Tích hợp mã nguồn: Các lập trình viên commit và push code của mình lên repository.
2. Kích hoạt CI: CI server giám sát repository và kiểm tra liên tục sự xem có sự thay đổi nào hay không. Khi phát ra sự thay đổi, CI server bắt đầu quá trình CI.
3. Build code: Với code mới nhất trên repository, CI server sẽ build mã nguồn này thành một phần mềm hoàn chỉnh.
4. Chạy các test case: Sau khi quá trình build hoàn tất, công cụ CI server bắt đầu chạy kiểm tra đơn vị (unit test), kiểm tra tích hợp (integration test), kiểm tra giao diện (UI test) hoặc các kiểm tra khác tùy thuộc vào yêu cầu của dự án.
5. Đưa ra thông báo: Nếu có lỗi trong quá trình kiểm tra, CI server sẽ đưa ra thông báo lỗi cho các lập trình viên biết. Các thông báo này thường được gửi qua email, chat...
6. Hợp nhất mã nguồn: Nếu không có lỗi, CI server sẽ tự động hợp nhất mã nguồn của các lập trình viên vào một phiên bản mới nhất của phần mềm. Phiên bản mới này được lưu trữ trong kho chung của dự án và có thể được triển khai vào các môi trường khác nhau của dự án.

*Continuous Deployment* (CD): Là quá trình triển khai tự động sau khi quá trình CI được diễn ra thành công. Quá trình CD sẽ được kích hoạt để triển khai ứng dụng của chúng ta vào các môi trường của dự án. Các gói phần mềm đã build thành công được triển khai bằng cách sử dụng các công cụ tự động hoặc được triển khai thủ công.

## ***Github Action***

*Github Action* là nền tảng miễn phí do Github cung cấp để tự động hóa quá trình CI/CD, cho phép người dùng tự định nghĩa các file workflows, tự động hóa các quy trình, hoạt động trong phát triển phần mềm. Một số đặc điểm nổi bật:

- Tích hợp sẵn với Github: Vì Github Actions được phát triển bởi Github nên nó được tích hợp sẵn với nền tảng Github. Do đó, người dùng có thể sử dụng các tính năng của Github Actions trực tiếp từ trang web của Github.
- Hỗ trợ chạy trên nhiều hệ điều hành: Github Actions có thể chạy trên nhiều hệ điều hành khác nhau: Windows, macOS, Ubuntu... Người dùng có thể kiểm tra ứng dụng của mình trên nhiều nền tảng khác nhau và đảm bảo rằng nó hoạt động đúng cách trên mọi nền tảng.
- Hỗ trợ chạy trên các môi trường ảo: Cho phép người dùng thực hiện trong các môi trường ảo khác nhau, người dùng có thể kiểm tra ứng dụng của mình trên nhiều môi trường đầy một cách dễ dàng. Người dùng có thể định nghĩa ngôn ngữ, các version của nó để run code.
- Tích hợp các công cụ bên thứ ba: Các công cụ bên thứ ba có thể được tích hợp vào quá trình CI/CD. Ví dụ, ta có thể sử dụng Github Actions để triển khai ứng dụng của mình lên AWS hoặc Azure.

Người dùng có thể sử dụng Github Hosted để chạy chương trình trên Github hoặc sử dụng Self Hosted Runner nếu có thêm các dependency đặc biệt.

Trong ***Grading System***, người dùng có thể sử dụng Github như một công cụ lưu trữ, tạo môi trường và run code bằng các file workflows. Số điểm của học viên được tính dựa vào kết quả của quá trình CI trên Github.

## **2.2 Công cụ chống gian lận**

Quá trình CI trên Github chỉ có thể phát hiện lỗi, kiểm tra các testcase, ... và hoàn toàn không thể phát hiện gian lận. Do đó, hệ thống cần sử dụng một công cụ khác phát hiện gian lận riêng dựa trên những đoạn mã đủ tiêu chuẩn trên Github.

Một trong những công cụ nổi bật chống gian lận là MOSS (Measure Of Software Similarity), được phát triển bởi đại học Stanford. Ứng dụng chính của MOSS là phát hiện đạo văn trong các lớp học lập trình. Một trong những ưu điểm nổi bật của MOSS là khả năng phát hiện những đoạn mã cơ bản, phổ biến trong nội bộ lớp học, hoặc cho phép những đoạn mã mẫu có thể copy để sử dụng. Ngoài ra, MOSS cũng có thể phát hiện được những trường hợp có thay đổi tên biến, tên hàm, thêm comment, ... nhằm đánh lạc hướng người kiểm tra.

Tuy nhiên, MOSS không phải là công cụ tự động hóa hoàn toàn quá trình kiểm tra gian lận. MOSS chỉ có thể chỉ ra và đánh dấu những điểm bất thường. Việc quyết định gian lận hay không vẫn cần người giảng dạy thực hiện. Dù vậy, MOSS tiết kiệm rất nhiều thời gian cho giảng viên trong quá trình điều tra gian lận giữa các học viên.



## 2.3 Công cụ trong hệ thống Grading System

Các công cụ trên chỉ là một vài trong số những công cụ có thể sử dụng cho hệ thống. Như đã đề cập ở trên, hệ thống được phát triển theo hướng có thể lựa chọn công cụ khác (Gitlab thay cho Github, sử dụng công cụ đánh giá gian lận khác, ...), có thể tích hợp vào cùng hệ thống (có thể sử dụng cả Gitlab và Github cho Repo và testing, integrate nhiều công cụ chống gian lận khác nhau vào cùng một hệ thống, ...) hoặc sử dụng một hệ thống testing và cheating được thiết kế phù hợp với nghiệp vụ riêng. Đây cũng là một trong những hướng phát triển lâu dài của hệ thống Grading System nhằm hướng đến thích nghi với quá trình phát triển cũng như có thể sử dụng trong nhiều lĩnh vực khác nhau.

Về cơ bản, quá trình hoạt động của Grading System khi tích hợp các công cụ như sau:

1. Người dùng sẽ tạo Repo để kiểm tra trên nền tảng lưu trữ của họ (ở đây là Github).
2. Học viên sẽ hoàn thành đề thi và nộp lại hệ thống.
3. Hệ thống sẽ tự động sử dụng quá trình CI của Github để chạy mã.
4. Sau khi có kết quả, người dùng sẽ chọn lọc và gửi những đoạn mã để kiểm tra gian lận, có thể là loại những bài làm compile lỗi, ...
5. Người dùng dựa trên kết quả trả về, review lại các bài làm và đánh giá.

Như vậy, hệ thống Grading System không phải là một hệ thống tự động hoàn toàn. Dù vậy, nếu so với quá trình chấm bài truyền thống, sử dụng Grading System sẽ giảm thiểu công sức, thời gian phải bỏ ra khi mà hầu hết các quá trình đều đã được tự động hóa. Hơn nữa, những quá trình cần sự can thiệp của con người trong hệ thống đều là những quá trình cần thiết nhằm đảm bảo việc chấm gian lận thực sự là đúng đắn, do công cụ chỉ hỗ trợ con người, nó không thể hoàn toàn thay thế được con người, đặc biệt là trong những trường hợp phức tạp như việc này, ít nhất là cho đến thời điểm hiện tại.

## 3 User story

### 3.1 Admin

- Là admin hệ thống, tôi muốn mình có thể quản lý các lớp học.

### 3.2 Người giảng dạy

- Là người giảng dạy, tôi muốn mình có thể xem các lớp mình đang giảng dạy.
- Là người giảng dạy, tôi muốn mình có thể upload tài liệu cho học viên.
- Là người giảng dạy, tôi muốn mình có thể gia hạn thời gian làm bài kiểm tra.
- Là người giảng dạy, tôi muốn mình có thể upload, sửa đổi đề kiểm tra.
- Là người giảng dạy, tôi muốn mình có thể upload, sửa đổi testcase.
- Là người giảng dạy, tôi muốn mình có thể xem, trả lời thắc mắc của học viên.
- Là người giảng dạy, tôi muốn mình có thể xem danh sách các bài làm.
- Là người giảng dạy, tôi muốn mình có thể review các bài làm.
- Là người giảng dạy, tôi muốn mình có thể upload điểm lên hệ thống.
- Là người giảng dạy, tôi muốn mình có thể xem thống kê điểm số của học viên.

### 3.3 Học viên

- Là học viên, tôi muốn mình có thể xem các lớp mà mình đang theo học.
- Là học viên, tôi muốn mình có thể quản lý các deadline của mình.
- Là học viên, tôi muốn mình có thể xem đề kiểm tra.
- Là học viên, tôi muốn mình có thể xem testcase mẫu.
- Là học viên, tôi muốn mình có thể hỏi, thảo luận về bài kiểm tra.
- Là học viên, tôi muốn mình có thể upload bài làm của mình.
- Là học viên, tôi muốn mình có thể xem điểm của mình.
- Là học viên, tôi muốn mình có thể xem thứ hạng của mình.

## 4 Functional & Non-Functional Requirements

### 4.1 Functional Requirements

#### *Admin*

- Quản lý lớp học: admin có quyền xem, thêm, xóa lớp học.

#### *Giảng viên*

- Xem danh sách các lớp học: mỗi giảng viên có thể có nhiều lớp học, và các bài kiểm tra, tài liệu, forum riêng cho mỗi lớp.
- Tạo bài kiểm tra: giảng viên có thể tạo bài kiểm tra trên hệ thống, các thông tin được cung cấp được dùng để tạo một Repo trong tài khoản của người dùng.
- Upload tài liệu, bài kiểm tra: giảng viên có thể upload bài kiểm tra, testcase mẫu cùng những tài liệu có liên quan cho học viên, các file này sẽ được thêm vào trong Repo đã tạo trước đó.
- Xem danh sách bài làm: danh sách các bài làm có thể là các branch / fork đã được tạo ra khi học viên bắt đầu làm bài.
- Review bài làm: các bài làm của học viên có thể được review bằng cách truy cập các nhánh cụ thể.
- Tạo diễn đàn trao đổi: với mỗi bài kiểm tra, giảng viên có thể tạo diễn đàn cho học viên có thể hỏi đáp, cập nhật thông tin về bài kiểm tra.
- Chỉnh sửa điểm số: mỗi bài làm có thể do bị phát hiện gian lận, cần scale điểm, giảng viên có thể sử dụng chức năng này để chỉnh sửa lại điểm số học viên.
- Thống kê bài làm: giảng viên có thể xem thông tin thống kê về số học viên tham gia, phân bố điểm số, ... trong bài kiểm tra.
- Xem hướng dẫn hệ thống: hệ thống được xây dựng dựa trên Github, do đó, đối với một số người dùng mới, sẽ cần hướng dẫn để sử dụng hệ thống hiệu quả, tránh mất nhiều thời gian.

#### *Học viên*

- Xem danh sách các lớp đang theo học: một học viên có thể đang tham gia nhiều lớp học, và có thể có nhiều bài kiểm tra cùng lúc.
- Tham gia bài kiểm tra: học viên có thể chọn tham gia bài kiểm tra hoặc không. Tuy nhiên, việc không tham gia bài kiểm tra vẫn gây ảnh hưởng đến điểm số nếu như bài kiểm tra là bắt buộc bởi giảng viên.
- Xem thông tin chi tiết bài kiểm tra: học viên có thể xem thông tin về các bài kiểm tra, tài liệu, testcase mẫu mà giảng viên đã upload.

- Tham gia diễn đàn trao đổi: học viên có thể tham gia diễn đàn trao đổi nhằm giải quyết các thắc mắc về bài kiểm tra.
- Nộp bài làm: học viên sau khi hoàn thành sẽ nộp (commit) bài làm lên branch được tạo khi tham gia bài kiểm tra.
- Xem điểm số: học viên có thể xem điểm số bài kiểm tra sau khi giảng viên đã chỉnh sửa điểm số.
- Xem hướng dẫn hệ thống: hệ thống được xây dựng dựa trên Github, do đó, đối với một số người dùng mới, sẽ cần hướng dẫn để sử dụng hệ thống hiệu quả, tránh mất nhiều thời gian.

## 4.2 Non-Functional Requirements

### *Tương thích*

- Tương thích với trình duyệt Chrome, Edge.

### *Đơn giản*

- Mỗi chức năng không quá 4 thao tác.
- Người dùng có thể sử dụng hệ thống sau khi xem hướng dẫn sử dụng.

### *Chịu tải*

- Đáp ứng được với lớp học tối đa 100 người.

### *Localization*

- Hệ thống hỗ trợ ngôn ngữ Tiếng Việt.
- Hệ thống hiển thị giờ dựa trên đồng hồ 24 giờ.
- Hệ thống hiển thị ngày theo định dạng dd/mm/yyyy.

## 5 Activity

Hệ thống sẽ cung cấp hai lựa chọn là Student host và Teacher host.

### 5.1 Student host activity

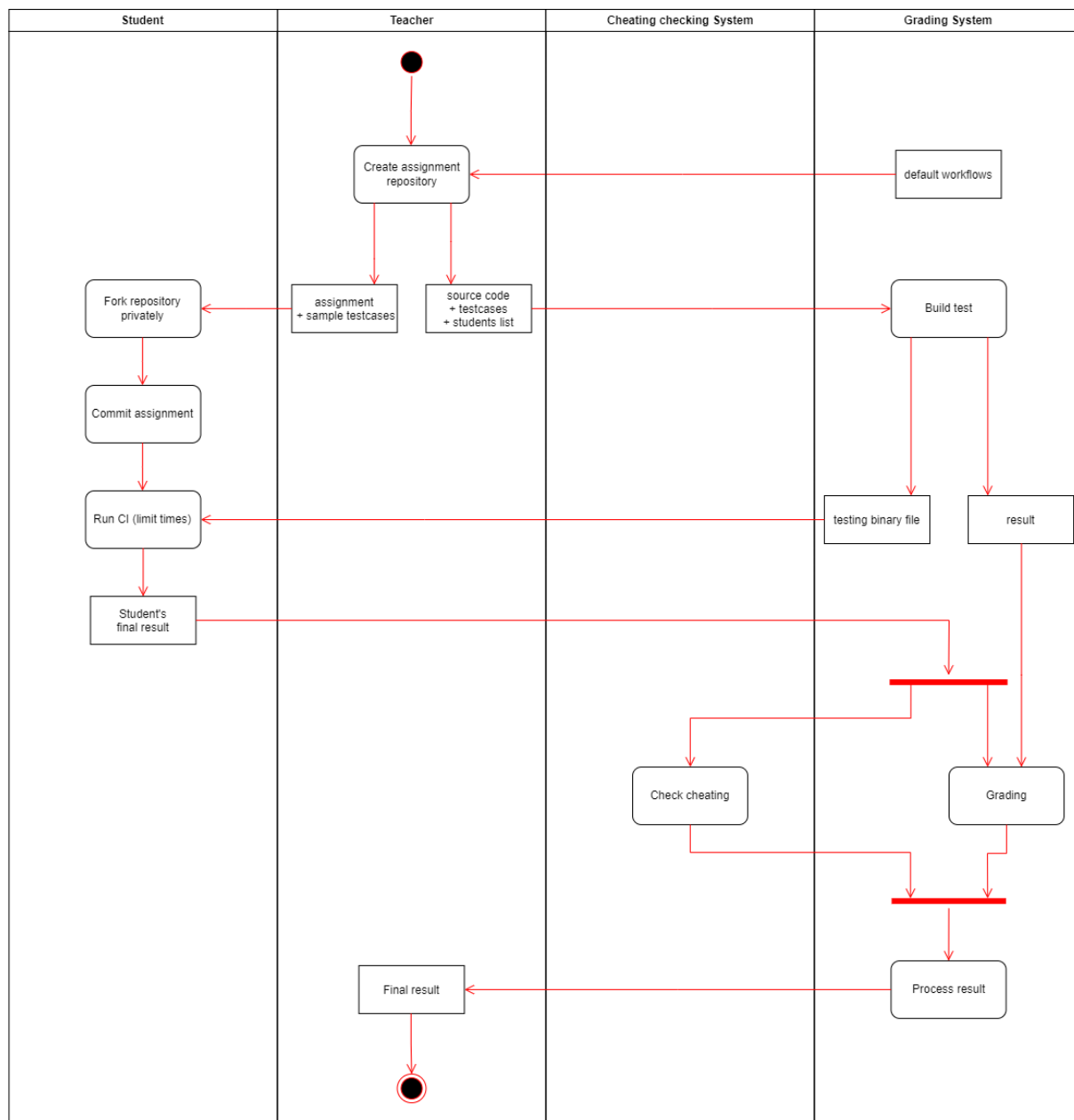
#### 5.1.1 Student host

Đối với Student host, hệ thống sẽ sử dụng quá kết quả của quá trình CI bên phía Github của học viên và đánh giá thông qua kết quả của source code mà người dùng cung cấp. Luồng thực thi của hệ thống như sau:

1. Người dùng sẽ tạo một repo đề dành cho học viên, danh sách học viên, source code cho bài kiểm tra và testcases để cung cấp cho hệ thống.
2. Hệ thống sẽ run file test tạo ra hai file: một file chứa testcase được build thành một binary file, file còn lại là kết quả của source code được người dùng cung cấp.
3. Học viên sẽ fork privately repo đề, hoàn thành bài kiểm tra và commit kết quả lên repo cá nhân. Người dùng có thể giới hạn số lần commit của học viên. Kết quả của quá trình CI có thể ẩn đối với học viên, và sẽ được gửi đến server cùng source code của lần commit đó.
4. Sau khi đến hạn nộp bài, hệ thống sẽ gửi source code của học viên đến công cụ kiểm tra gian lận, kết quả quá trình CI của học viên sẽ được so sánh với kết quả run bởi source code do người dùng cấp và đánh giá kết quả.
5. Kết quả của quá trình chấm gian lận và so sánh với kết quả của người dùng sẽ được xử lý, tổng hợp và gửi lại cho người dùng.

### 5.1.2 Student host activity diagram

Quá trình hoạt động của Student host được mô tả bằng sơ đồ sau:



**Hình 1:** System Activity diagram - Student host

## 5.2 Teacher host activity

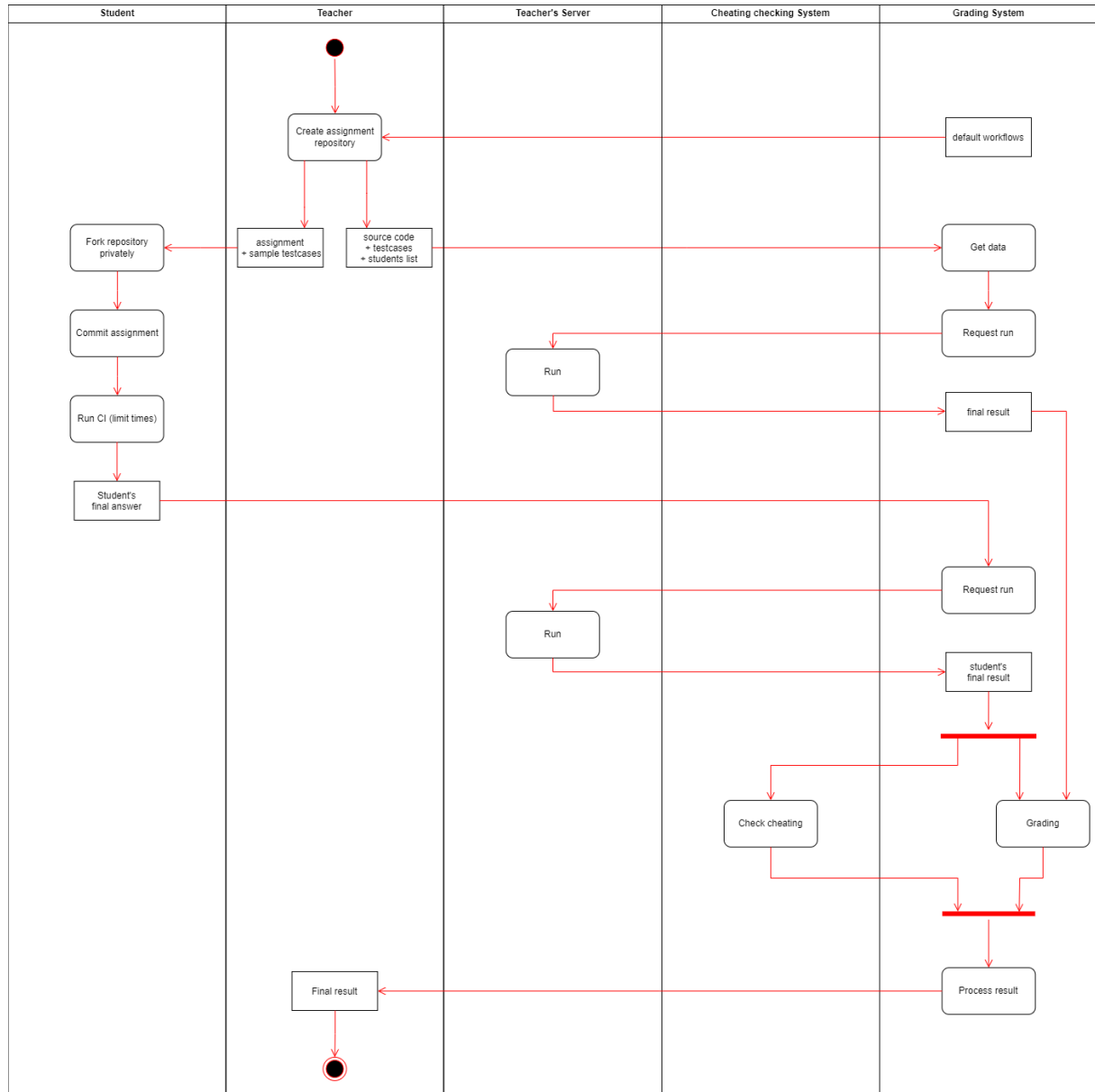
### 5.2.1 Teacher host

Đối với Teacher host, hệ thống sẽ sử dụng API IDE của người dùng cung cấp để run các source code của người dùng và của học viên. Kết quả vẫn sẽ được lấy và đánh giá tương tự như Student host. Luồng thực thi của hệ thống như sau:

1. Người dùng sẽ tạo một repo đề dành cho học viên, danh sách học viên, source code cho bài kiểm tra và testcases để cung cấp cho hệ thống.
2. Học viên sẽ fork privately repo đề, hoàn thành bài kiểm tra và commit kết quả lên repo cá nhân. Người dùng có thể giới hạn số lần commit của học viên.
3. Hệ thống sẽ chuyển tiếp request đến server IDE của người dùng, run source code của học viên và lấy lại kết quả. Source code của lần commit đó sẽ được gửi đến hệ thống. Kết quả này sẽ không được gửi lại cho học viên.
4. Sau khi đến hạn nộp bài, hệ thống sẽ gửi source code của học viên đến công cụ kiểm tra gian lận, kết quả run source code của học viên sẽ được so sánh với kết quả run bởi source code do người dùng cấp và đánh giá kết quả.
5. Kết quả của quá trình chấm gian lận và so sánh với kết quả của người dùng sẽ được xử lý, tổng hợp và gửi lại cho người dùng.

### 5.2.2 Teacher host activity diagram

Quá trình hoạt động của Teacher host được mô tả bằng sơ đồ sau:



**Hình 2:** System Activity diagram - Teacher host



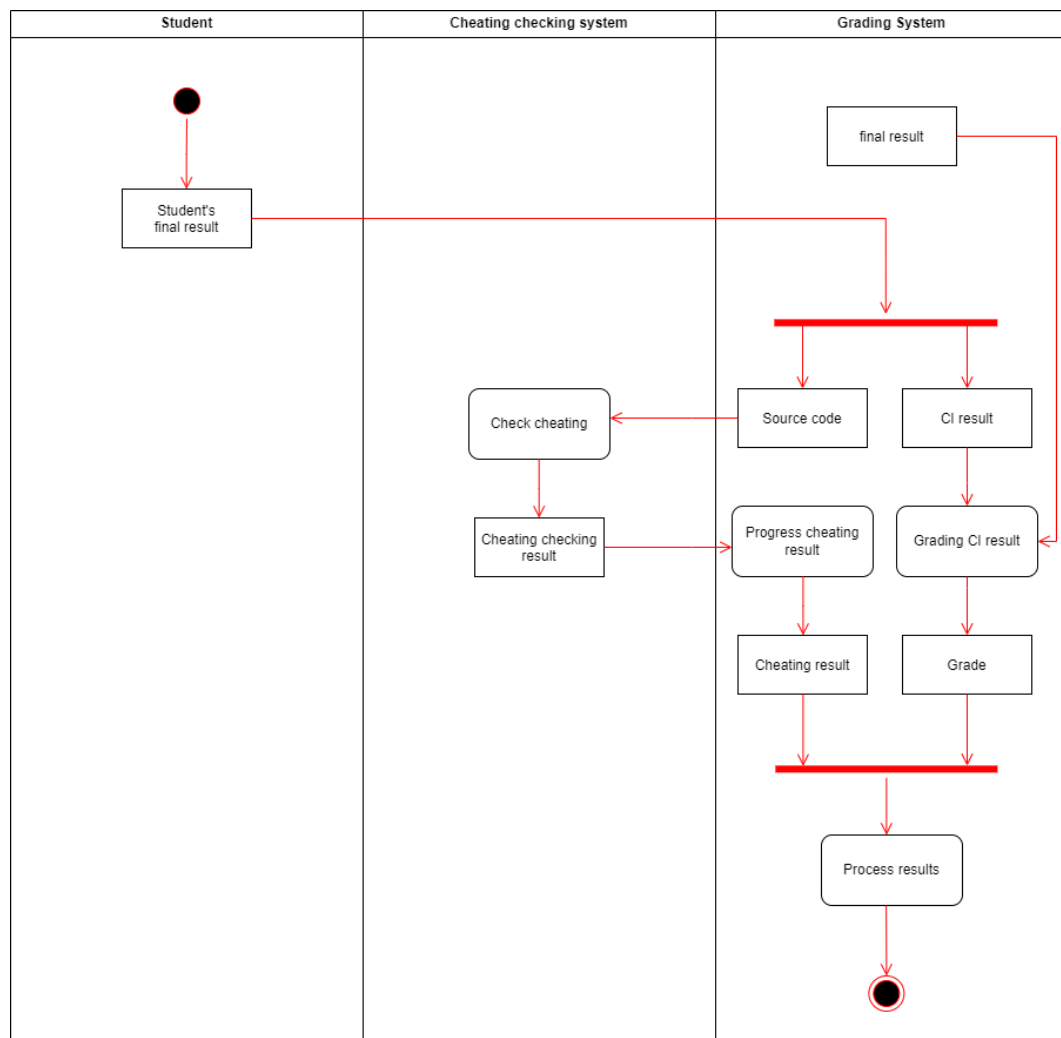
## 5.3 Grading and checking activity

### 5.3.1 Grading and Checking

Grading and Checking là module chính của hệ thống. Mặc định, hệ thống sẽ chấm theo phần trăm testcase passed. Tuy nhiên, nếu người dùng có phương pháp chấm điểm khác, hệ thống có thể sử dụng cách chấm đó. Ngoài ra, người dùng cũng có thể chọn công cụ đánh giá gian lận mà mình tin tưởng. Luồng thực thi của module như sau:

1. Module sẽ nhận kết quả của người dùng cùng với kết quả và source code của học viên.
2. Source code của học viên sẽ được gửi đến công cụ đánh giá gian lận, đồng thời, kết quả từ source code của học viên sẽ được so sánh với kết quả từ source code của người dùng.
3. Kết quả kiểm tra gian lận và điểm số sẽ được xử lý và gửi lại cho người dùng.

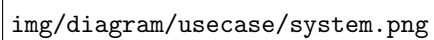
### 5.3.2 Grading and Checking activity diagram



*Hình 3: Grading and Checking result activity diagram*

## 6 Usecases

### 6.1 System diagram



img/diagram/usecase/system.png

*Hình 4: System diagram*

Use-case name	
Create by	
Date created	
Actor	Admin
Description	
Trigger	
Pre-condition	
Post-condition	
Normal flow	1. 2. 3. 4.
Alternative flow	4.1
Exception	
Note and issue	
None-functional	

Bảng 1



## 6.2 Usecase senario

### 6.2.1 Manage classes

### 6.2.2

### 6.2.3

### 6.2.4

### 6.2.5

### 6.2.6

### 6.2.7

### 6.2.8

### 6.2.9

### 6.2.10

### 6.2.11

### 6.2.12

### 6.2.13

### 6.2.14

### 6.2.15



## Tài liệu

- [1] Cheating rate does not decrease
- [2] How to use Github Action
- [3] Github Action for multiple language
- [4] CI/CD and Github Action
- [5] Measure of Software Similarity - MOSS
- [6] Set up automatic testing with Github Action