

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



ĐỒ ÁN TỔNG HỢP - CÔNG NGHỆ PHẦN MỀM

GRADE PORTAL SYSTEM  
MODULE TELEGRAM-BOT  
TRA CỨU ĐIỂM

GVHD: ThS. Lê Đình Thuận

SVTH: Nhóm Tây tiến đoàn binh không mọc tóc

Trần Thanh Phong (2212571)

Đặng Quốc Phong (2212548)

Lê Vĩnh Nghiệp (2212213)

Trần Minh Quân (2212823)

Trương Anh Tuấn (2213810)

Nguyễn Lê Hoàng Phúc (2212629)

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 12 2024



# Mục lục

<b>Mục lục</b>	i
<b>Danh sách thành viên</b>	ii
<b>Danh sách bảng</b>	iii
<b>Danh sách hình vẽ</b>	iv
<b>1 Tổng quan về dự án</b>	1
1.1 Mục tiêu của dự án . . . . .	1
1.2 Đối tượng sử dụng . . . . .	2
1.3 Công nghệ sử dụng . . . . .	2
<b>2 Quá trình triển khai telegram bot</b>	3
2.1 Đặc tả yêu cầu . . . . .	3
2.2 Thiết kế và triển khai . . . . .	7
2.3 Kiểm thử . . . . .	18
2.4 Kế hoạch phát triển trong tương lai . . . . .	27
<b>3 Tổng kết và bài học</b>	28
3.1 Tổng kết . . . . .	28
3.2 Bài học kinh nghiệm . . . . .	29

# Danh sách thành viên

STT	Họ tên	MSSV	Nhiệm vụ	Đóng góp	Ký tên
1	Trần Thanh Phong	2212571	1.Hiện thực toàn bộ câu lệnh + kết nối API, 2.Deploy docker	100%	
2	Đặng Quốc Phong	2212548	1.Hiện thực toàn bộ câu lệnh + kết nối API, 2.Handle lỗi OTP, 3.Usecase diagram, 4.Deploy docker	100%	
3	Lê Vĩnh Nghiệp	2212213	1.Handle lỗi hiển thị điểm, 2.Báo cáo, 3.Test bot	100%	
4	Trần Minh Quân	2212823	1.Handle lỗi hiển thị thông tin cá nhân, 2.Deploy docker	100%	
5	Trương Anh Tuấn	2213810	1.Handle lỗi register, 2.Activity diagram	100%	
6	Nguyễn Lê Hoàng Phúc	2212629	1.Xây dựng cấu trúc thư mục, 2.Handle lỗi login, 3.Báo cáo	100%	
<b>Thông tin PO:</b>				<b>Ký tên</b>	
Họ và tên: Nguyễn Lê Hoàng Phúc Sđt: 0766909533 Email: phuc.nguyenlehoang707@hcmut.edu.vn					
<b>Nhận xét của giảng viên:</b> <i>(Đây là khu vực nhận xét của giảng viên)</i>				<b>Ký tên</b>	

# **Danh sách bảng**

2.1.1	Yêu cầu chức năng của bot Telegram . . . . .	3
2.1.2	Yêu cầu phi chức năng của bot Telegram . . . . .	4
2.1.3	Bảng tổng hợp các câu lệnh trong bot Telegram. . . . .	5

# Danh sách hình vẽ

2.2.1	Mô tả kiến trúc Client-server kinh điển của hệ thống Telegram-bot. (Nguồn: <a href="#">How To Create a Telegram Bot? [5 Easy Steps]</a> ) . . . . .	8
2.2.2	Sơ đồ Usecase mô tả chức năng của toàn bộ hệ thống. . . . .	9
2.2.3	Sơ đồ Activity mô tả luồng hoạt động chính của hệ thống. . . . .	10
2.2.4	Khởi tạo, cấu hình bot . . . . .	11
2.2.5	Hàm xử lý lệnh lấy OTP và phản hồi cho người dùng . . . . .	12
2.2.6	Hàm xử lý lệnh đăng ký . . . . .	12
2.2.7	Hàm xử lý yêu cầu đăng nhập . . . . .	12
2.2.8	Hàm xử lý yêu cầu tra cứu điểm . . . . .	13
2.2.9	Hàm xử lý yêu cầu xem tất cả điểm . . . . .	13
2.2.10	Dockerfile . . . . .	14
2.2.11	Lệnh tạo docker image . . . . .	15
2.2.12	Lệnh đặt tên cho image . . . . .	15
2.2.13	Lệnh đẩy image lên Docker Hub . . . . .	15
2.2.14	docker-compose.yml . . . . .	15
2.2.15	Lệnh kiểm tra logs . . . . .	16
2.2.16	Lệnh cập nhật ứng dụng . . . . .	16
2.3.1	Giao diện khi nhập lệnh /start trên bot Telegram . . . . .	23
2.3.2	Thông báo của bot khi người dùng nhập sai mật khẩu . . . . .	23
2.3.3	Bot hiển thị điểm thành phần của môn học có mã học phần CO3103-HK233 . . . . .	24
2.3.4	Bot hiển thị thông báo mã học phần không hợp lệ . . . . .	24
2.3.5	Bot hiển thị tất cả điểm môn học của sinh viên . . . . .	25
2.3.6	Bot hiển thị lịch sử tra cứu điểm của sinh viên . . . . .	26

# Chương 1

## Tổng quan về dự án



### 1.1 Mục tiêu của dự án

Dự án Telegram bot là một trong ba thành phần trong hệ sinh thái Grade\_Portal thuộc phạm vi Đồ án tổng hợp (hướng công nghệ phần mềm) diễn ra trong 15 tuần học của kỳ 241 năm học 2024-2025, được thiết kế riêng để phục vụ cộng đồng sinh viên Đại học Bách khoa - Đại học Quốc gia Thành phố Hồ Chí Minh. Với mục tiêu tối ưu hóa trải nghiệm người dùng, bot mang đến một kênh thông tin nhanh chóng, trực quan và tiện lợi để sinh viên tra cứu điểm số, theo dõi tiến độ học tập ngay trên nền tảng nhắn tin phổ biến Telegram. Module Telegram-bot tra cứu điểm được tạo ra với các mục tiêu chính sau:

- *Thứ nhất*, giúp sinh viên chủ động theo dõi điểm số, từ đó đưa ra những quyết định học tập hiệu quả và kịp thời. Đây cũng chính là mục tiêu cốt lõi, chức năng chính của bot.

## **1.2. Đối tượng sử dụng**

---

- *Thứ hai*, trở thành một công cụ hữu dụng trong công tác tra cứu điểm số cá nhân, hỗ trợ nhu cầu xem điểm nhanh chóng của sinh viên khi website chính thức của nhà trường gấp sự cố. Sinh viên chỉ cần sử dụng một giao diện duy nhất trên Telegram để nắm bắt mọi thông tin liên quan đến kết quả học tập.
- *Thứ ba*, biến Telegram-bot này không chỉ đơn thuần là một công cụ tra cứu điểm số mà còn là một người bạn đồng hành, giúp sinh viên lập kế hoạch học tập, quản lý thời gian hiệu quả và đạt được những mục tiêu đã đề ra.

## **1.2 Đối tượng sử dụng**

Đối tượng phục vụ chính và chủ yếu của bot telegram được nhóm tác giả phát triển là *Sinh viên* đang học tập và nghiên cứu tại trường Đại học Bách khoa - Đại học Quốc gia thành phố Hồ Chí Minh. Khi có nhu cầu tra cứu điểm số, hệ thống bot luôn sẵn sàng hỗ trợ và phục vụ. Cụ thể:

- Sinh viên trình độ năm nhất: Cần làm quen với môi trường đại học và theo dõi sát sao kết quả học tập.
- Sinh viên trình độ năm hai, năm ba: Cần quản lý điểm để đăng ký các môn học chuyên ngành và chuẩn bị cho việc bảo vệ đồ án.
- Sinh viên trình độ năm bốn: Cần theo dõi điểm tổng kết để đảm bảo tốt nghiệp đúng hạn.

## **1.3 Công nghệ sử dụng**

- **Source control:** Github
- **Meeting:** Google Meet
- **Ngôn ngữ lập trình:** Golang
- **Nền tảng triển khai:** Telegram Bot API
- **Database:** MongoDB
- **Testing:** Postman
- **Deploy:** Github, Docker

## Chương 2

# Quá trình triển khai telegram bot

### 2.1 Đặc tả yêu cầu

#### 2.1.1 Yêu cầu chức năng

Xem bảng 2.1.1.

#### 2.1.2 Yêu cầu phi chức năng

Xem hình 2.1.2.

#### 2.1.3 Cấu trúc câu lệnh sơ bộ

Xem bảng 2.1.3.

1. STUDENT	1.1	Sinh viên có thể xem điểm số (điểm thành phần, điểm thi cuối kỳ, và điểm trung bình môn) theo từng môn, từng học kỳ của cá nhân thông qua việc gửi tin nhắn đúng cú pháp cho Telegram bot
	1.2	Sinh viên có thể kiểm tra thông tin cá nhân của phiên đăng nhập hiện tại (tên + MSSV).
	1.3	Sinh viên phải thông qua hệ thống xác thực tài khoản - Authentication (Mỗi phiên đăng nhập chỉ tương ứng với một tài khoản của một sinh viên, đăng nhập với mật khẩu và OTP).

Bảng 2.1.1: Yêu cầu chức năng của bot Telegram

## **2.1. Đặc tả yêu cầu**

---

<b>2. PRODUCT</b>	2.1	<b>Hiệu suất:</b> Thời gian phản hồi của bot không vượt quá 2s/ một yêu cầu từ phía người dùng cho mọi câu lệnh được thực hiện.
	2.2	<b>Khả năng đáp ứng:</b> Bot phải có khả năng xử lý ít nhất 1000 yêu cầu đồng thời mà không gây gián đoạn dịch vụ.
	2.3	<b>Tính khả dụng:</b> Bot phải có tính khả dụng 24/7.
	2.4	<b>Tính tương thích:</b> Cần tương thích với phiên bản Telegram mới nhất (phương cách sử dụng API).
	2.5	<b>Tính hiệu dụng:</b> Sinh viên có thể sử dụng bot chỉ với hướng dẫn nhanh, ngắn gọn từ tin nhắn bot gửi khi bắt đầu cuộc trò chuyện.
<b>3. ORGANIZATIONAL</b>	3.1	Tất cả các phiên bản mã nguồn của bot phải được quản lý qua GitHub, và có mô tả rõ ràng.

Bảng 2.1.2: Yêu cầu phi chức năng của bot Telegram

Bảng 2.1.3: Bảng tổng hợp các câu lệnh trong bot Telegram.

STT	Lệnh	Chức năng	Mô tả	Bot hiển thị
1	/login + [MSSV] + [password]	Đăng nhập vào hệ thống.	Cho phép sinh viên đăng nhập bằng mã số sinh viên (MSSV) và mật khẩu đã đăng ký trước đó.	Nếu đăng nhập thành công: Hiển thị chào mừng. Nếu thất bại: báo lỗi.
2	/grade + [Mã học phần]	Tra cứu điểm theo mã học phần.	Người dùng nhập mã học phần (courseID) để nhận được điểm chi tiết cho môn học tương ứng.	Nếu thành công: hiển thị điểm chi tiết môn học. Nếu lỗi: báo lỗi.
3	/allgrade	Xem tất cả điểm của các môn đã học.	Hiển thị bảng điểm chi tiết của tất cả học phần sinh viên đã hoàn thành.	Nếu thành công: hiển thị danh sách điểm chi tiết (nhiều môn). Nếu lỗi: báo lỗi.
4	/history	Xem lịch sử điểm đã tra cứu.	Hiển thị các môn học mà người dùng đã tra cứu gần đây nhất.	Nếu có lịch sử: hiển thị lịch sử đã tra cứu. Nếu không có lịch sử: báo không có lịch sử.
5.	/clear	Xóa lịch sử tra cứu	Xóa toàn bộ lịch sử tra cứu của người dùng.	Nếu xóa thành công: “Lịch sử tra cứu đã được xóa”. Nếu xảy ra lỗi: “Error”.
6	/info	Xem thông tin phiên đăng nhập	Hiển thị thông tin cá nhân của sinh viên (họ tên, lớp, MSSV, email, ...).	Nếu thành công: hiển thị thông tin cá nhân của sinh viên. Nếu lỗi: báo lỗi.
7	/register [MSSV Password OTP]	Đăng ký tạo phiên đăng nhập	Đăng ký tài khoản sinh viên trong hệ thống.	Nếu thành công: thông báo thành công. Nếu lỗi: báo lỗi.
Tiếp theo trang sau				

## 2.1. Đặc tả yêu cầu

STT	Lệnh	Chức năng	Mô tả	Bot hiển thị
8	/getotp [MSSV]	Lấy OTP để đăng ký hoặc đổi mật khẩu	Gửi mã OTP đến email sinh viên để hỗ trợ đăng ký hoặc lấy lại mật khẩu.	Nếu thành công: báo kết quả. Nếu lỗi: báo lỗi.
9.	/login [MSSV Password]	Đăng nhập	Đăng nhập vào hệ thống để sử dụng các tính năng liên quan đến tài khoản sinh viên.	Nếu thành công: thông báo đăng nhập thành công. Nếu thất bại: thông báo lỗi.

## 2.2 Thiết kế và triển khai

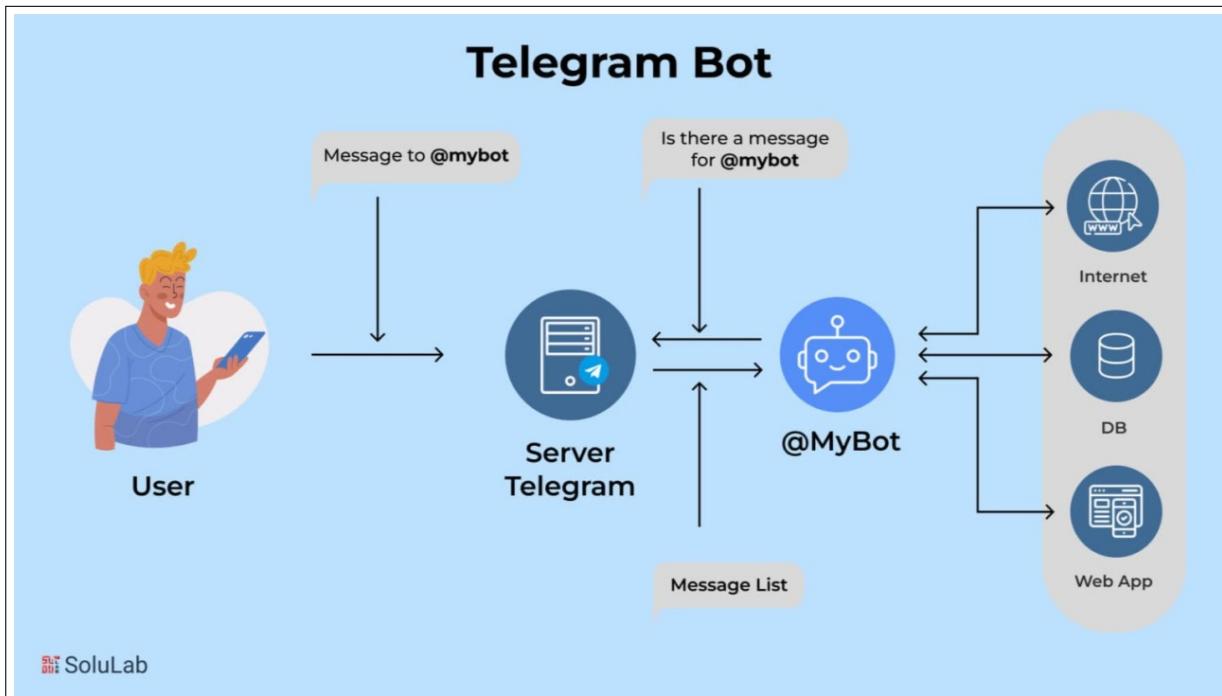
### 2.2.1 Thiết kế kiến trúc

Kiến trúc của hệ thống telegram bot này là mô hình Client - Server kinh điển (Tham khảo hình 2.2.1).

#### Thành phần chính trong kiến trúc

- Client (Telegram Bot)
  - Chức năng:
    - \* Gửi và nhận thông tin từ người dùng qua giao diện Telegram.
    - \* Xử lý các lệnh từ sinh viên (ví dụ: tra cứu điểm, xóa lịch sử, lấy OTP, đăng nhập).
    - \* Hiển thị phản hồi từ hệ thống dưới dạng văn bản hoặc JSON (nếu cần chi tiết).
  - Giao tiếp:
    - \* Sử dụng Telegram Bot API để kết nối với máy chủ Telegram.
- Application Server (Bot Backend)
  - Vai trò:
    - \* Xử lý logic chính của bot, bao gồm kiểm tra lệnh, lấy thông tin từ các dịch vụ phụ trợ, và gửi phản hồi lại cho người dùng.
  - Chức năng chính:
    - \* Xử lý lệnh: Kiểm tra các cú pháp lệnh từ người dùng và ánh xạ chúng đến các hàm xử lý tương ứng (ví dụ: HandleHelp, HandleRegister).
    - \* Quản lý người dùng: Lưu trữ trạng thái và lịch sử tra cứu của người dùng trong cơ sở dữ liệu.
    - \* Tích hợp API: Kết nối với các dịch vụ khác (như dịch vụ quản lý điểm số của Grade\_Portal).
  - Ngôn ngữ:
    - \* Triển khai bằng Go (Golang) để đảm bảo hiệu năng và khả năng xử lý đồng thời cao.
- Database Server

## 2.2. Thiết kế và triển khai



Hình 2.2.1: Mô tả kiến trúc Client-server kinh điển của hệ thống Telegram-bot. (Nguồn: [How To Create a Telegram Bot? \[5 Easy Steps\]](#))

- Mục đích: Lưu trữ thông tin liên quan đến người dùng, lịch sử tra cứu, và dữ liệu cần thiết để thực hiện các lệnh.
- Dữ liệu được lưu trữ gồm:
  - \* ID người dùng Telegram.
  - \* Lịch sử lệnh.
  - \* Thông tin đăng nhập và token.
- Hệ thống API của Grade\_Portal
  - Chức năng: Cung cấp dữ liệu điểm số, thông tin khóa học, hoặc bất kỳ thông tin liên quan khác mà Telegram bot yêu cầu.
  - Đặc điểm: Được triển khai như một dịch vụ RESTful API.

### Quy trình hoạt động

1. Người dùng gửi yêu cầu qua Telegram bot. Ví dụ: `/help` hoặc `/login [MSSV] [password]`.
2. Bot chuyển yêu cầu tới Application Server: Bot sử dụng Telegram Bot API để gửi dữ liệu lệnh đến backend.

## 2.2. Thiết kế và triển khai

3. Application Server xử lý lệnh:

- Kiểm tra cú pháp và tính hợp lệ của lệnh.
- Nếu lệnh cần dữ liệu từ hệ thống Grade\_Portal, server gửi yêu cầu đến API tương ứng.
- Lưu trữ lịch sử tra cứu vào cơ sở dữ liệu.

4. Kết quả từ API của Grade\_Portal:

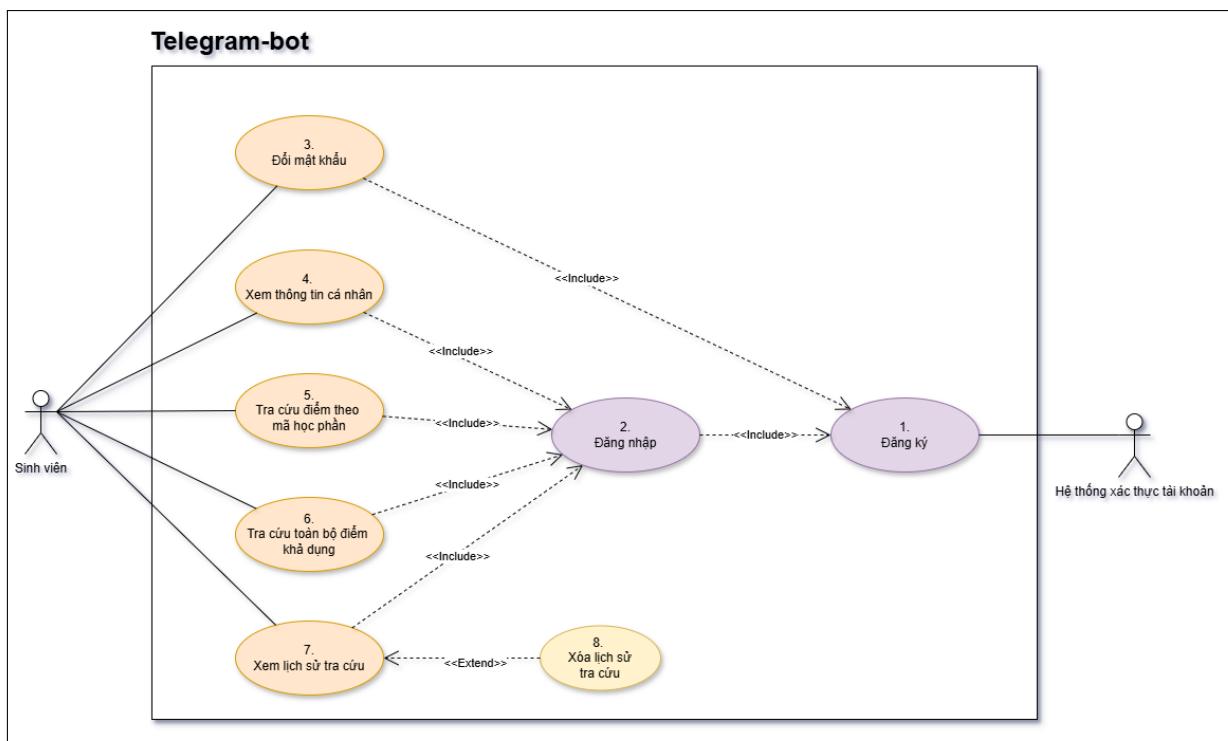
- Application Server nhận kết quả từ API (ví dụ: điểm số của một môn học).
- Chuẩn bị phản hồi để gửi lại người dùng.

5. Bot gửi phản hồi đến người dùng:

- Bot trả về thông tin được định dạng (dạng text cho các phản hồi thông thường hoặc JSON cho các phản hồi điểm số) cho người dùng qua Telegram.

### 2.2.2 Thiết kế thành phần

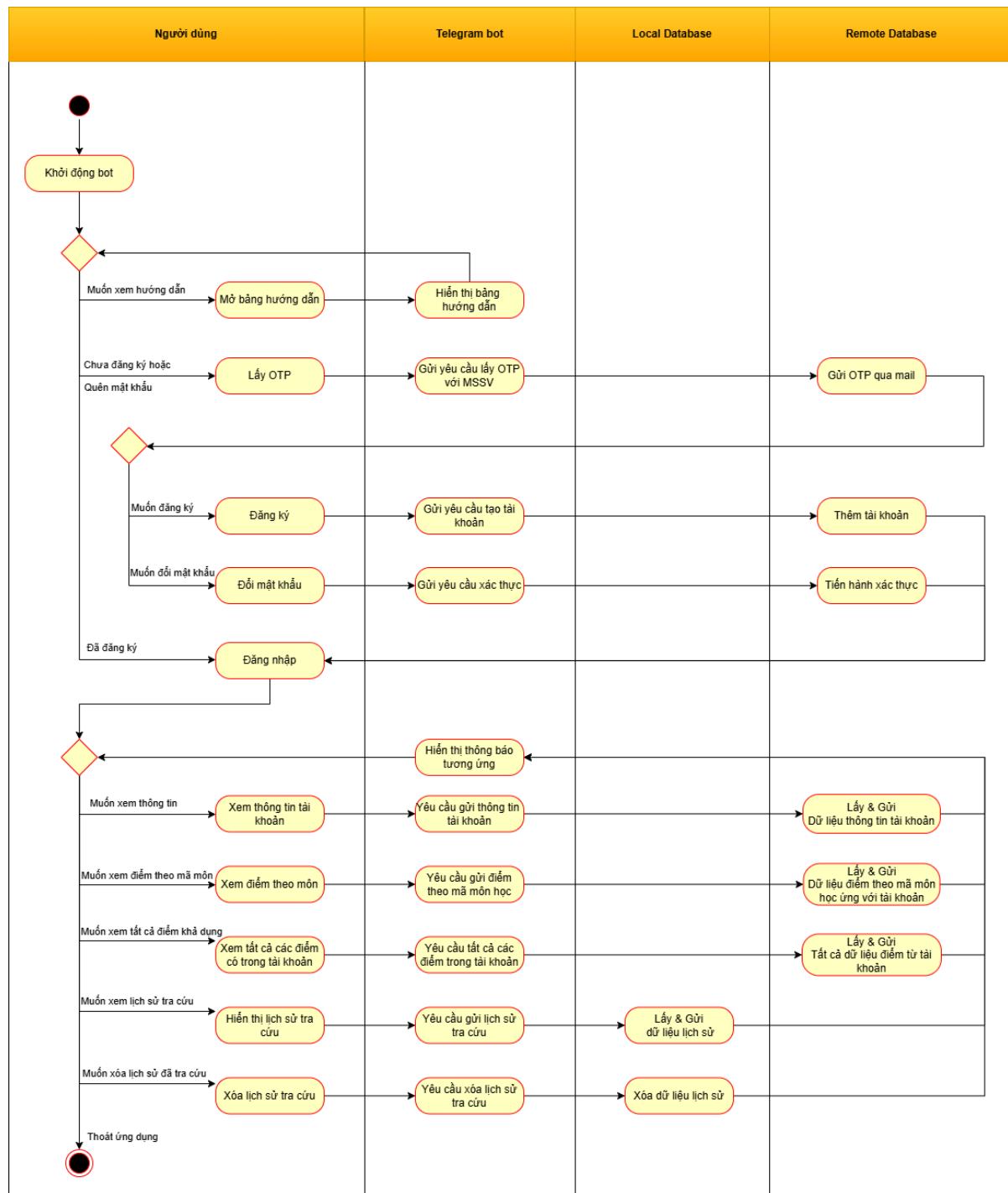
#### Sơ đồ Usecase (hình 2.2.2)



Hình 2.2.2: Sơ đồ Usecase mô tả chức năng của toàn bộ hệ thống.

## 2.2. Thiết kế và triển khai

## Sơ đồ Activity (hình 2.2.3)



Hình 2.2.3: Sơ đồ Activity mô tả luồng hoạt động chính của hệ thống.

### 2.2.3 Xây dựng chức năng

#### Mô tả chức năng

Bot hỗ trợ sinh viên đăng ký, đăng nhập bằng mã số sinh viên và tra cứu điểm môn học qua mã học phần hoặc tra cứu điểm của tất cả môn học.

#### Các bước chức năng chính

1. Lấy OTP: Sinh viên nhập lệnh /getotp + [MSSV] để lấy OTP phục vụ cho việc đăng ký.
2. Đăng ký: Sinh viên nhập lệnh /register + [MSSV] + [Password] + [OTP] để đăng ký tài khoản.
3. Đăng nhập: Sinh viên nhập lệnh /login + [MSSV] + [Password] để đăng nhập vào hệ thống.
4. Tra cứu điểm: Sinh viên nhập lệnh /grade + [Mã học phần] để tra cứu điểm của môn học bằng mã học phần.
5. Xem tất cả điểm: Sinh viên nhập lệnh /allGrade để xem điểm của tất cả môn học.

#### Chi tiết quá trình phát triển

- Khởi tạo bot

```
func Start() {
    // Dùng hàm LoadConfig() định nghĩa trong file config.go - package config.
    cfg := config.LoadConfig()
    // Khởi tạo bot
    bot, err := tgbotapi.NewBotAPI(cfg.BOT_TOKEN)
    if err != nil {
        log.Fatalf("Failed to create bot: %v", err) // In lỗi chi tiết
    }
    // connect DBMongo
    config.ConnectMongoDB(cfg.DBURL)
    fmt.Printf("Authorized on account %s\n", bot.Self.UserName)
```

Hình 2.2.4: Khởi tạo, cấu hình bot

- Xây dựng hàm để bot trả về OTP cho người dùng

## 2.2. Thiết kế và triển khai

```
func HandleOTP(bot *tgbotapi.BotAPI, update tgbotapi.Update, mssv string, cfg *config.Config) {
    _, err := services.GetOTP(mssv, cfg)
    var response string
    if err == nil {
        response = "OTP đã được gửi về email của bạn, vui lòng kiểm tra email."
    } else {
        response = err.Error()
    }
    msg := tgbotapi.NewMessage(update.Message.Chat.ID, response)
    bot.Send(msg)
}
```

Hình 2.2.5: Hàm xử lý lệnh lấy OTP và phản hồi cho người dùng

- Xây dựng hàm để bot xử lý việc đăng ký của người dùng

```
func HandleRegister(bot *tgbotapi.BotAPI, update tgbotapi.Update, input string, cfg *config.Config) {
    parts := strings.Split(input, " ")
    var mssv, pw, otp string
    mssv, pw, otp = parts[0], parts[1], parts[2]
    resp, err := services.RegisterStudent(mssv, pw, otp, cfg)
    var response string
    if err == nil {
        response = resp.Msg + ", vui lòng login bằng cú pháp /login_mssv_password để sử dụng dịch vụ."
    } else {
        response = "Error fetching student info:" + err.Error()
    }
    msg := tgbotapi.NewMessage(update.Message.Chat.ID, response)
    bot.Send(msg)
}
```

Hình 2.2.6: Hàm xử lý lệnh đăng ký

- Xây dựng hàm để bot xử lý việc đăng nhập vào hệ thống

```
// Gửi yêu cầu đăng nhập qua API
resp, err := services.Login(update.Message.Chat.ID, mssv, pw, cfg)
var response string
if err != nil {
    // Log lỗi để hỗ trợ debug
    log.Printf("Login error for MSSV %s: %v", mssv, err)

    // Xử lý các loại lỗi
    if strings.Contains(err.Error(), "Timeout") {
        response = "API không phản hồi, vui lòng thử lại sau."
    } else if strings.Contains(err.Error(), "HTTP 400") {
        response = "Sai MSSV hoặc mật khẩu. Vui lòng kiểm tra và thử lại."
    } else if strings.Contains(err.Error(), "unexpected status code") {
        response = "Hệ thống đang gặp sự cố, vui lòng thử lại sau."
    } else {
        response = fmt.Sprintf("Đăng nhập thất bại. Chi tiết lỗi: %s", err.Error())
    }
} else {
    // Đăng nhập thành công
    response = "Đăng nhập thành công! Các khóa học bạn đang có là: " + strings.Join(resp.ListCourse, ", ")
}

// Gửi phản hồi đến người dùng
msg := tgbotapi.NewMessage(update.Message.Chat.ID, response)
bot.Send(msg)
}
```

Hình 2.2.7: Hàm xử lý yêu cầu đăng nhập

- Xây dựng hàm để bot xử lý và trả về điểm chi tiết của môn học cho người dùng

## 2.2. Thiết kế và triển khai

```
func HandleGrade(bot *tgbotapi.BotAPI, update tgbotapi.Update, semesterOrCourseID string, cfg *config.Config) {
    resp, err := services.GetGrades(update.Message.Chat.ID, semesterOrCourseID, cfg)
    var response string
    if err != nil {
        response = "Không thể lấy dữ liệu điểm: " + err.Error()
        msg := tgbotapi.NewMessage(update.Message.Chat.ID, response)
        bot.Send(msg)
        return
    }
    result := map[string]interface{}{
        "course_id": semesterOrCourseID,
        "course_name": resp.Name,
        "scores": map[string]interface{}[{
            "a_BT": resp.Score.BT,
            "b_TN": resp.Score.TN,
            "c_BTL": resp.Score.BTL,
            "d_GK": resp.Score.GK,
            "e_CK": resp.Score.CK,
        }],
    }
    jsonData, err := json.MarshalIndent(result, "", " ")
    if err != nil {
        response = "Lỗi khi tạo điểm vui lòng thử lại sau: " + err.Error()
        msg := tgbotapi.NewMessage(update.Message.Chat.ID, response)
        bot.Send(msg)
        return
    }
    msgText := fmt.Sprintf(````json\n%s\n````, string(jsonData))
    msg := tgbotapi.NewMessage(update.Message.Chat.ID, msgText)
    msg.ParseMode = "MarkdownV2"
    bot.Send(msg)
}
```

Hình 2.2.8: Hàm xử lý yêu cầu tra cứu điểm

- Xây dựng hàm để bot xử lý và trả về tất cả điểm môn học cho người dùng

```
func HandleAllGrade(bot *tgbotapi.BotAPI, update tgbotapi.Update, cfg *config.Config) {
    resp, err := services.GetAllGrades(update.Message.Chat.ID, cfg)
    if err != nil {
        response := "Không thể lấy dữ liệu điểm, vui lòng thử lại sau"
        msg := tgbotapi.NewMessage(update.Message.Chat.ID, string(response))
        bot.Send(msg)
    } else {
        for _, grade := range resp.AllGrades {
            gradeData := map[string]interface{}{
                "course_id": grade.Ms,
                "course_name": grade.Name,
                "scores": map[string]interface{}{
                    "a_BT": grade.Score.BT,
                    "b_TN": grade.Score.TN,
                    "c_BTL": grade.Score.BTL,
                    "d_GK": grade.Score.GK,
                    "e_CK": grade.Score.CK,
                },
            }
            responseJSON, err := json.MarshalIndent(gradeData, "", " ")
            if err != nil {
                fmt.Println("Lỗi khi mã hóa JSON:", err)
                msg := tgbotapi.NewMessage(update.Message.Chat.ID, "Không thể xử lý dữ liệu JSON.")
                bot.Send(msg)
                return
            }
            msgText := fmt.Sprintf(````json\n%s\n````, string(responseJSON))
            msg := tgbotapi.NewMessage(update.Message.Chat.ID, string(msgText))
            msg.ParseMode = "MarkdownV2"
            bot.Send(msg)
        }
    }
}
```

Hình 2.2.9: Hàm xử lý yêu cầu xem tất cả điểm

## 2.2. Thiết kế và triển khai

---

### 2.2.4 Triển khai - Deployment

#### Quy trình triển khai

##### 1. Thiết lập Docker Image

- Mục tiêu: Đóng gói ứng dụng bot Telegram vào một Docker Image trên máy local.
- Dockerfile: là một tệp văn bản chứa tập hợp các lệnh để tự động xây dựng một Docker Image.

```
FROM golang:1.23 AS builder

WORKDIR /app

# Copy go.mod and go.sum and download dependencies
COPY go.mod go.sum .
RUN go mod download

# Copy the source code and build the binary
COPY . .
RUN CGO_ENABLED=0 GOOS=linux go build -o main main.go

# Stage 2: Run
FROM alpine:latest

# Thiết lập thư mục làm việc
WORKDIR /root/

# Sao chép binary từ giai đoạn build
COPY --from=builder /app/main .

# Expose the port
EXPOSE 8080

# Start the application
CMD ["./main"]
```

Hình 2.2.10: Dockerfile

- Chạy lệnh build để tạo một docker image trên máy cá nhân.

```
docker build -t grade-portal-tele .
```

Hình 2.2.11: Lệnh tạo docker image

### 2. Đẩy Image vừa tạo lên Docker Hub

- Đặt tên cho image theo định dạng chuẩn.

```
docker tag grade-portal-tele phongthanphong/grade-portal-tele:latest
```

Hình 2.2.12: Lệnh đặt tên cho image

- Đẩy image lên Docker Hub.

```
docker push phongthanphong/grade-portal-tele:latest
```

Hình 2.2.13: Lệnh đẩy image lên Docker Hub

### 3. Xây dựng docker-compose giúp kéo image dự án về và chạy trên máy cá nhân

```
services:  
  app:  
    image: phongthanphong/grade-portal-tele:latest  
    env_file:  
      - .env  
    ports:  
      - "8080:8080"  
    container_name: grade-portal-tele  
    restart: unless-stopped
```

Hình 2.2.14: docker-compose.yml

File docker-compose giúp kéo image grade-portal-tele phiên bản latest trên Docker Hub của user phongthanphong.

Load các biến môi trường từ file .env vào container.

Map cổng 8080 của máy host tới cổng 8080 của container.

Đặt tên cho container là grade-portal-tele.

Container sẽ tự động khởi động lại khi gặp lỗi trừ khi bạn dừng nó thủ công.

## 2.2. Thiết kế và triển khai

---

### 4. Xây dựng CI/CD với Github Actions

GitHub Actions được sử dụng để tự động hóa quy trình triển khai.

Workflow bao gồm hai giai đoạn:

- Xây dựng và đẩy Docker Image lên Docker Hub.
- Triển khai ứng dụng trên máy chủ.

## Kết quả triển khai

### 1. Tự động hóa

- Workflow tự động kích hoạt khi có thay đổi trên nhánh main.
- Docker Image được tự động build và đẩy lên Docker Hub.

### 2. Triển khai nhanh chóng

- Image từ Docker Hub được kéo xuống và triển khai trên máy chủ.
- Quá trình deploy không gây downtime nhờ vào việc xóa container cũ và chạy container mới.

### 3. Tối ưu hóa bảo mật

- Các thông tin nhạy cảm như BOT\_TOKEN, DATABASE\_URL, và API\_URL được bảo mật qua GitHub Secrets.

## Giám sát và bảo trì

### 1. Kiểm tra logs của container

```
docker logs -f grade-portal
```

Hình 2.2.15: Lệnh kiểm tra logs

### 2. Cập nhật ứng dụng

```
docker compose up -d
```

Hình 2.2.16: Lệnh cập nhật ứng dụng

## **2.2. Thiết kế và triển khai**

---

### **Kết luận**

Việc triển khai dự án Bot Telegram hỗ trợ tra cứu điểm bằng Docker và CI/CD giúp giảm thiểu lỗi thủ công, cải thiện hiệu suất triển khai và đảm bảo tính đồng nhất của môi trường. Hệ thống này hỗ trợ quá trình phát triển linh hoạt và dễ dàng mở rộng.

## 2.3. Kiểm thử

---

# 2.3 Kiểm thử

## 2.3.1 Kiểm thử chức năng

### Mục tiêu kiểm thử

- Đảm bảo các chức năng của bot Telegram hoạt động đúng với yêu cầu đề ra.
- Phát hiện lỗi tiềm ẩn và kiểm tra hiệu suất của hệ thống trong các tình huống sử dụng thực tế.

### Phương pháp kiểm thử

- Kiểm thử thủ công:** Gửi các lệnh trực tiếp vào bot Telegram, ghi nhận phản hồi của bot và kiểm tra với kết quả mong đợi, từ đó ghi nhận lại để tiến hành nâng cấp, khắc phục nếu có vấn đề.

### Kịch bản kiểm thử

STT	Chức năng	Mô tả kiểm thử	Kết quả mong đợi
1	Lấy OTP	Gửi lệnh /getotp [MSSV] để lấy về OTP cho việc đăng ký hoặc đổi mật khẩu. <ul style="list-style-type: none"><li><b>Case 1:</b> Nhập đúng cú pháp và mã số sinh viên.</li><li><b>Case 2:</b> Nhập đúng cú pháp nhưng sai mã số sinh viên.</li></ul>	<ul style="list-style-type: none"><li><b>Case 1:</b> Bot trả về OTP.</li><li><b>Case 2:</b> Bot báo lỗi sai MSSV.</li></ul>

### 2.3. Kiểm thử

2	Đăng ký	<p>Gửi lệnh <code>/register [MSSV Password OTP]</code> để đăng ký tài khoản trong hệ thống.</p> <ul style="list-style-type: none"> <li><b>Case 1:</b> Nhập đúng cú pháp, mã số sinh viên và OTP.</li> <li><b>Case 2:</b> Nhập đúng cú pháp nhưng sai OTP.</li> </ul>	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiện thông báo đăng ký thành công.</li> <li><b>Case 2:</b> Bot báo lỗi.</li> </ul>
3	Đăng nhập	<p>Gửi lệnh <code>/login [MSSV Password]</code> để sinh viên có thể đăng nhập vào hệ thống.</p> <ul style="list-style-type: none"> <li><b>Case 1:</b> Nhập đúng cú pháp, mã số sinh viên và password đã đăng ký trước đó.</li> <li><b>Case 2:</b> Nhập đúng cú pháp nhưng sai MSSV hoặc mật khẩu.</li> </ul>	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiện thông báo chào mừng sinh viên đến hệ thống.</li> <li><b>Case 2:</b> Bot báo lỗi.</li> </ul>
4	Xem thông tin sinh viên	<p>Gửi lệnh <code>/info</code> để xem thông tin của sinh viên trong phiên đăng nhập.</p> <ul style="list-style-type: none"> <li><b>Case 1:</b> Nhập đúng cú pháp.</li> </ul>	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiển thị thông tin sinh viên.</li> </ul>
5	Tra cứu điểm	<p>Gửi lệnh <code>/grade [Mã học phần]</code> để xem điểm chi tiết của môn học có mã học phần đó.</p> <ul style="list-style-type: none"> <li><b>Case 1:</b> Nhập đúng cú pháp và đúng mã học phần.</li> <li><b>Case 2:</b> Nhập sai mã học phần.</li> </ul>	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiển thị điểm chi tiết của môn học.</li> <li><b>Case 2:</b> Bot báo lỗi mã học phần sai hoặc không tồn tại.</li> </ul>

### 2.3. Kiểm thử

---

6	Tra cứu lịch sử	<p>Gửi lệnh /history để xem lịch sử điểm của các môn học mà sinh viên đã tra cứu.</p> <ul style="list-style-type: none"> <li>• <b>Case 1:</b> Nhập đúng cú pháp.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Case 1:</b> Bot hiển thị lịch sử tra cứu điểm trong hệ thống.</li> </ul>
---	-----------------	---	--

#### Công cụ kiểm thử

- **Bot Telegram:** Kiểm tra giao tiếp với bot từ ứng dụng Telegram bằng việc nhập lệnh trực tiếp từ người dùng.

#### Kết quả kiểm thử

STT	Chức năng	Kết quả thực tế	Kết luận	Hướng khắc phục (nếu có lỗi)
1	Lấy OTP	<ul style="list-style-type: none"> <li>• <b>Case 1:</b> Bot trả về được OTP chính xác.</li> <li>• <b>Case 2:</b> Bot báo lỗi sai MSSV.</li> </ul>	Thành công	
2	Đăng ký	<ul style="list-style-type: none"> <li>• <b>Case 1:</b> Bot hiện thông báo đăng ký thành công.</li> <li>• <b>Case 2:</b> Bot báo lỗi.</li> </ul>	Thành công	
3	Đăng nhập	<ul style="list-style-type: none"> <li>• <b>Case 1:</b> Bot hiện thông báo chào mừng sinh viên đăng nhập vào được hệ thống.</li> <li>• <b>Case 2:</b> Bot báo lỗi.</li> </ul>	Thành công	

### 2.3. Kiểm thử

4	Xem thông tin sinh viên	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiển thị đúng thông tin của sinh viên.</li> </ul>	Thành công	
5	Tra cứu điểm	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiển thị được chính xác điểm thành phần của môn học có mã môn được sinh viên nhập vào.</li> <li><b>Case 2:</b> Bot báo lỗi.</li> </ul>	Thành công	
6	Tra cứu lịch sử	<ul style="list-style-type: none"> <li><b>Case 1:</b> Bot hiển thị đúng lịch sử tra cứu gồm các môn học được sinh viên tra cứu điểm trước đó.</li> </ul>	Thành công	

#### 2.3.2 Kiểm thử giao diện và trải nghiệm người dùng

##### Mục tiêu kiểm thử

- Đảm bảo giao diện bot hiển thị chính xác, không lỗi font hoặc tràn khung trên các nền tảng khác nhau.
- Đánh giá trải nghiệm người dùng để đảm bảo các câu lệnh của bot dễ sử dụng, thân thiện.
- Đảm bảo các thông báo, phản hồi của bot rõ ràng, phù hợp.

##### Kịch bản kiểm thử

STT	Kịch bản	Kết quả mong đợi
1	Người dùng nhập lệnh /start	Bot hiển thị thông báo chào mừng và các câu lệnh có sẵn

### 2.3. Kiểm thử

---

2	Người dùng nhập sai mật khẩu của lệnh /login	Bot hiển thị thông báo sai mật khẩu
3	Người dùng nhập lệnh /grade + [Mã học phần]	Bot hiển thị điểm của môn học dưới định dạng JSON, với đầy đủ các điểm thành phần
4	Người dùng nhập sai mã học phần của lệnh /grade	Bot hiển thị thông báo mã học phần không hợp lệ hoặc đã hết hạn.
5	Người dùng nhập lệnh /allGrade	Bot hiển thị điểm của tất cả môn học của sinh viên dưới định dạng JSON.
6	Người dùng nhập lệnh /history	Bot hiển thị lịch sử tra cứu điểm của sinh viên dưới định dạng JSON.

#### Quá trình kiểm thử

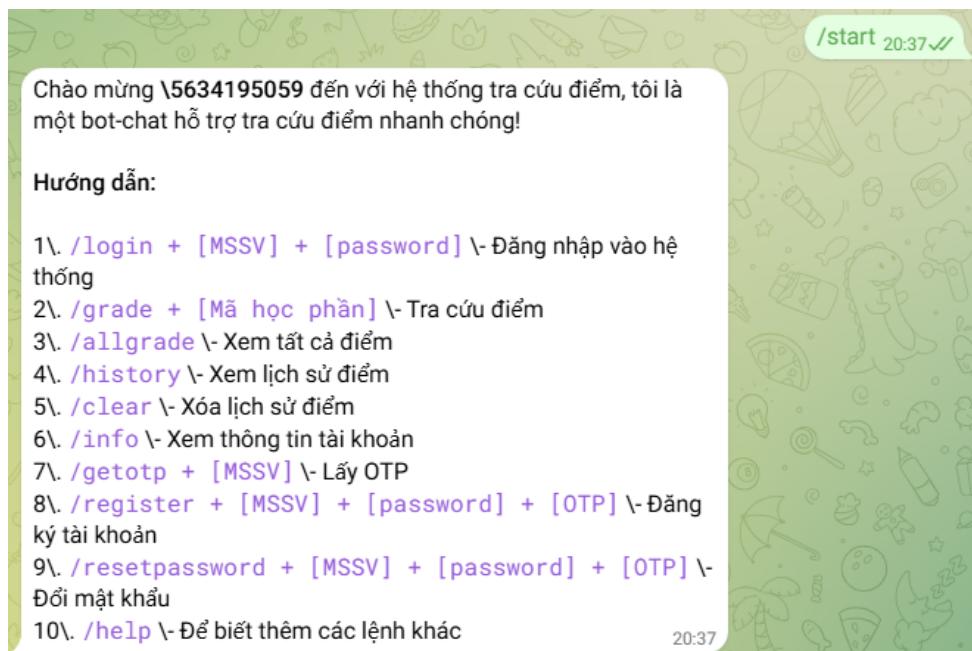
- **Nền tảng kiểm thử:** Telegram Web
- **Phương pháp kiểm thử:** Sử dụng trực tiếp giao diện Telegram để thao tác nhập lệnh và quan sát phản hồi.
- **Quá trình kiểm thử:**
  1. Thực hiện các kịch bản kiểm thử như đã liệt kê.
  2. Ghi nhận kết quả thực tế, bao gồm hình ảnh minh họa.
  3. Đánh giá các lỗi liên quan đến hiển thị hoặc phản hồi của bot (nếu có).

#### Kết quả kiểm thử

##### Kịch bản 1: Người dùng nhập lệnh /start

- **Kết quả thực tế:** Bot hiển thị đúng tin nhắn chào mừng và danh sách các lệnh như sau:

## 2.3. Kiểm thử



Hình 2.3.1: Giao diện khi nhập lệnh /start trên bot Telegram

- **Trạng thái:** Pass

**Kịch bản 2:** Người dùng nhập sai mật khẩu của lệnh /login

- **Kết quả thực tế:** Bot hiển thị thông báo sai mật khẩu:



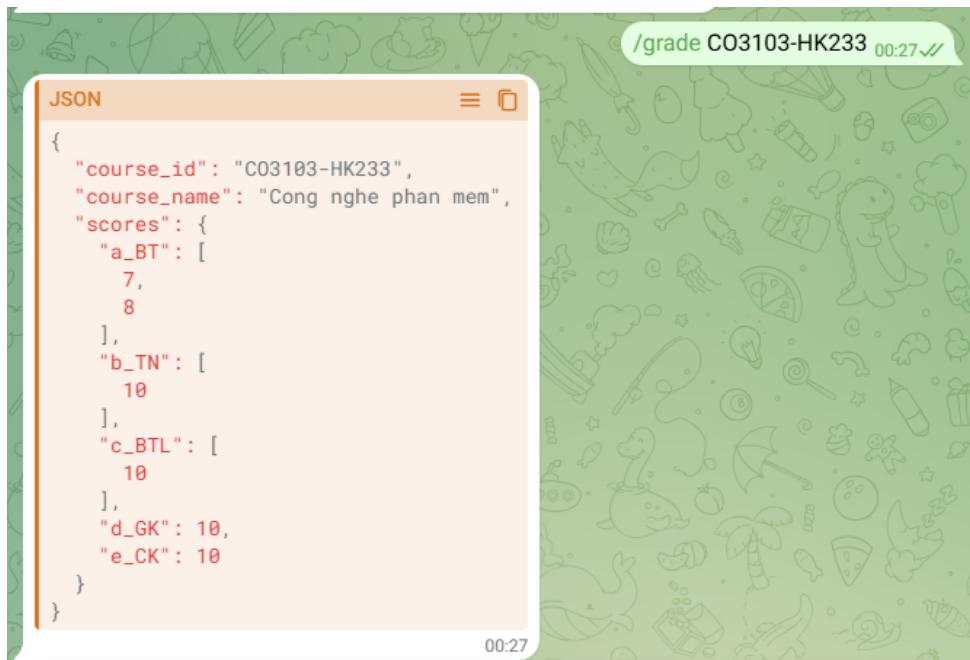
Hình 2.3.2: Thông báo của bot khi người dùng nhập sai mật khẩu

- **Trạng thái:** Pass

**Kịch bản 3:** Người dùng nhập lệnh /grade + [Mã học phần]

- **Kết quả thực tế:** Bot hiển thị được điểm thành phần của môn học:

## 2.3. Kiểm thử

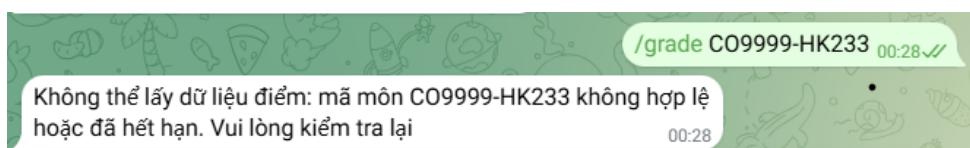


Hình 2.3.3: Bot hiển thị điểm thành phần của môn học có mã học phần CO3103-HK233

- **Trạng thái:** Pass

**Kịch bản 4:** Người dùng nhập sai mã học phần của lệnh /grade + [Mã học phần]

- **Kết quả thực tế:** Bot hiển thị thông báo mã học phần không hợp lệ hoặc hết hạn:



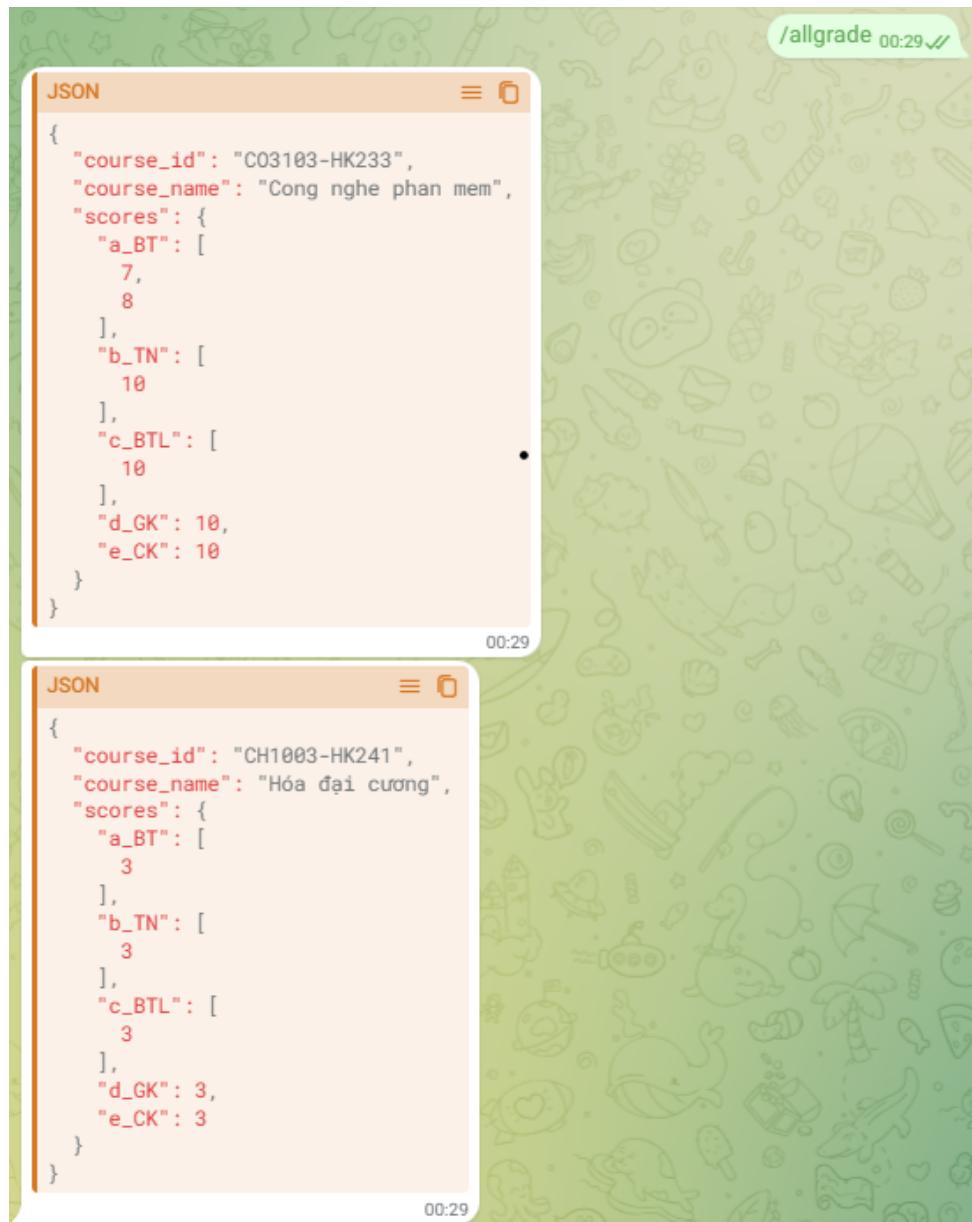
Hình 2.3.4: Bot hiển thị thông báo mã học phần không hợp lệ

- **Trạng thái:** Pass

**Kịch bản 5:** Người dùng nhập lệnh /allGrade

- **Kết quả thực tế:** Bot hiển thị được điểm của tất cả môn học của sinh viên:

### 2.3. Kiểm thử



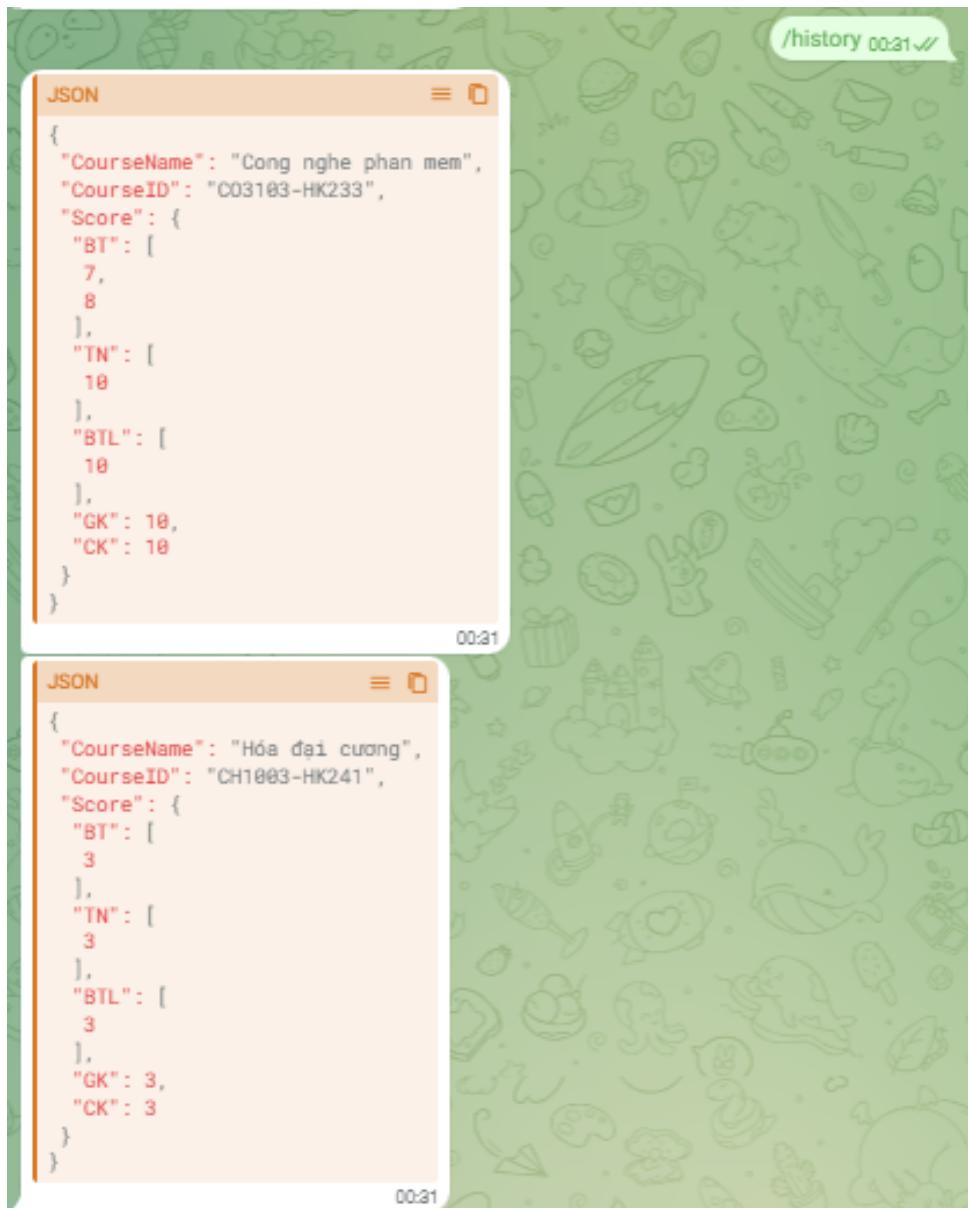
Hình 2.3.5: Bot hiển thị tất cả điểm môn học của sinh viên

- **Trạng thái:** Pass

**Kịch bản 6:** Người dùng nhập lệnh /history

- **Kết quả thực tế:** Bot hiển thị được lịch sử tra cứu điểm của sinh viên:

### 2.3. Kiểm thử



Hình 2.3.6: Bot hiển thị lịch sử tra cứu điểm của sinh viên

- **Trạng thái:** Pass

### Tổng kết kiểm thử

- Tất cả kịch bản kiểm thử đều **Pass**.
- Giao diện và phản hồi của bot hoạt động đúng yêu cầu.
- Câu lệnh dễ sử dụng, thân thiện với người dùng.
- Tin nhắn bot hiển thị rõ ràng, không có lỗi.

## **2.4. Kế hoạch phát triển trong tương lai**

---

### **2.4 Kế hoạch phát triển trong tương lai**

Sau khi hoàn thiện các chức năng cơ bản, bot Telegram hỗ trợ tra cứu điểm có thể được phát triển với các tính năng được mở rộng và cải tiến như sau:

1. Phát triển các thêm tính năng mới:

- Phát triển thêm tính năng cho giáo viên: thêm chức năng, các câu lệnh để giáo viên có thể nhập điểm, bổ sung hoặc thay đổi điểm bằng việc gửi file bảng điểm, có thể là .csv hoặc tương tự cho bot.
- Thông báo cho người dùng: hệ thống có thể được phát triển để cung cấp thông báo, có thể cho sinh viên hoặc giáo viên, để thông báo khi điểm có sự thay đổi như vừa có điểm môn học mới hoặc điểm cũ có sự thay đổi, ...
- Hỗ trợ đa ngôn ngữ: phát triển giao diện song ngữ để phục vụ cả sinh viên trong nước và du học sinh nước ngoài.

2. Cải tiến hiệu năng và bảo mật:

- Tối ưu hóa tốc độ phản hồi khi truy vấn cơ sở dữ liệu lớn.
- Phát triển để bot hoạt động tốt ngay cả khi số lượng truy cập trở nên lớn hơn.
- Nâng cao bảo mật thông tin cá nhân, có thể tích hợp với hệ thống xác thực SSO của trường.

3. Tích hợp công nghệ mới:

- Tích hợp AI: có thể ứng dụng AI để bot phản hồi tốt hơn về các yêu cầu bị lỗi của sinh viên hay để trả lời tự động các câu hỏi phức tạp từ sinh viên.
- Kết nối với hệ thống LMS: tích hợp với hệ thống LMS của trường để đồng bộ dữ liệu tự động.

Những kế hoạch này sẽ được triển khai theo từng giai đoạn cụ thể nhằm đảm bảo chất lượng và hiệu quả của hệ thống trong tương lai.

# Chương 3

## Tổng kết và bài học

### 3.1 Tổng kết

Báo cáo này trình bày toàn bộ quá trình xây dựng và phát triển bot Telegram hỗ trợ tra cứu điểm cho sinh viên. Bot được thiết kế nhằm tự động hóa quy trình tra cứu điểm, giảm thiểu thời gian công sức trong việc tra cứu điểm cho sinh viên. Qua quá trình kiểm thử, hệ thống đã chứng minh tính hoạt động ổn định, đáp ứng tốt phần lớn các yêu cầu ban đầu. Nhưng bên cạnh đó, vì một số lí do mà dự án chưa thể được hoàn thiện một cách tốt nhất.

Về những gì đã làm được, trong quá trình phát triển bot, nhóm đã tích hợp tốt các công nghệ như được yêu cầu ban đầu. Đầu tiên, môi trường phát triển và triển khai được chuẩn hóa thông qua Docker, giúp đảm bảo tính đồng nhất, dễ dàng tái tạo môi trường trên các hệ thống khác nhau. Tiếp theo, quy trình phát triển đã được tích hợp CI/CD, giúp rút ngắn thời gian triển khai, giảm thiểu lỗi và đảm bảo chất lượng trong những lần cập nhật hệ thống. Về phần chức năng, bot đã hoàn thành tốt các chức năng cơ bản cho sinh viên như đăng nhập, đăng ký, tra cứu điểm thành phần của từng môn cũng như tra cứu điểm tất cả môn học, ngoài ra còn có một chức năng như xem thông tin sinh viên, tra cứu lịch sử.

Mặc dù đã đạt được nhiều thành công, dự án vẫn còn một số hạn chế và tính năng chưa được triển khai đầy đủ. Một tính năng quan trọng chưa thể được hoàn thành là các câu lệnh dành cho giảng viên để thực hiện việc tra cứu, nhập cũng như chỉnh sửa điểm. Ngoài ra, hệ thống còn chưa có chức năng thông báo cho người dùng khi điểm có chỉnh sửa hoặc bổ sung. Hệ thống cũng chưa được tối ưu hóa hiệu năng, đôi khi tốc độ phản hồi giảm nếu số lượng truy cập tăng đột biến.

Tóm lại, bot Telegram tra cứu điểm sinh viên, tuy còn nhiều thiếu sót, cơ bản đã hoàn thành mục tiêu đề ra, mang lại lợi ích thiết thực và mở ra nhiều cơ hội cải tiến trong tương lai.

## **3.2 Bài học kinh nghiệm**

### **3.2.1 Khó khăn thách thức**

Trong quá trình phát triển bot, nhóm đã gặp phải không ít khó khăn thách thức:

- **Khó khăn về ngôn ngữ lập trình:** Đa số thành viên trong nhóm lần đầu tiếp cận với ngôn ngữ lập trình Golang cũng như bot Telegram, theo đó quá trình phát triển bot nhiều lần phải bị gián đoạn.
- **Tích hợp công nghệ mới:** Việc làm quen và áp dụng Docker để đóng gói hệ thống cũng như sử dụng CI/CD để tự động hóa quy trình đòi hỏi nhiều thời gian để nghiên cứu và thử nghiệm.
- **Vấn đề giao tiếp:** Lần đầu làm việc trong dự án có quy mô lớn, nhóm gặp một số vấn đề trong việc giao tiếp, dẫn đến việc vài mục tiêu đề ra không thể hoàn thành đúng tiến độ.

### **3.2.2 Bài học kinh nghiệm**

Từ những khó khăn thách thức ở trên, nhóm cũng rút ra được một số bài học kinh nghiệm:

- **Nắm bắt công nghệ mới:** Việc nghiên cứu và áp dụng Docker cũng như CI/CD đã mang lại nhiều kinh nghiệm thực tiễn trong việc đóng gói ứng dụng và triển khai tự động. Nhờ vậy, quá trình phát triển về sau trở nên dễ dàng hơn, đồng thời tối ưu thời gian triển khai và cập nhật bot.
- **Hợp tác và quản lý dự án:** Để dự án được triển khai thuận lợi thì cần có sự hợp tác của tất cả thành viên. Phân chia công việc rõ ràng, quản lý mã nguồn qua Git, chia nhỏ công việc, xác định mục tiêu rõ ràng, sử dụng các công nghệ mới giúp công việc trở nên nhanh chóng, hiệu quả và tránh được xung đột trong quá trình phát triển.
- **Bảo mật:** Vấn đề bảo mật cần được ưu tiên hàng đầu để tránh việc điểm của sinh viên bị phổ biến tràn lan ra bên ngoài. Phương pháp đăng ký bằng mã số sinh viên, OTP cũng như mật khẩu đã đảm bảo dữ liệu của sinh viên được bảo vệ an toàn.