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

Lớp TN01 - Nhóm TN409

Bài tập lớn Mở rộng (CO300A)

Công nghệ Phần mềm

"Mini Software Engineering Project: Smart Event Reminder System (SERS)"

Giảng viên: Nguyễn Thành Công

Sinh viên: Nguyễn Thái Học - 2311100 *(Lớp TN01)*
Lê Nguyễn Kim Khôi - 2311671 *(Lớp TN01)*
Nguyễn Thiện Minh - 2312097 *(Lớp TN01)*
Huỳnh Đức Nhân - 2312420 *(Lớp TN01)*
Phạm Trần Minh Trí - 2313622 *(Lớp TN01)*

THÀNH PHỐ HỒ CHÍ MINH, 12/2025

Mục lục

Danh sách hình ảnh

Danh sách bảng

Danh sách thành viên

TT	Họ và tên	MSSV	Công việc phụ trách	% công việc
1	Nguyễn Thái Học	2311100	- Section 3, 4 - -	100%
2	Lê Nguyễn Kim Khôi	2311671	- Section 5, 6 - -	100%
3	Nguyễn Thiện Minh	2312097	- Section 5, 6 - -	100%
4	Huỳnh Đức Nhân	2312420	- Section 3, 4 - -	100%
5	Phạm Trần Minh Trí	2313622	- Section 1, 2 - -	100%

Bảng 1: Danh sách thành viên

Mở đầu

Ví dụ code listing:

```
def hello():
    print("Hello World!")
```

Listing 1: Hello World

Ví dụ đính kèm ảnh:

Hình 1: Logo

Reference 1, 1

1 Requirements Engineering

1.1 Stakeholders

Đối với hệ thống Smart Event Reminder System (SERS), các stakeholder bao gồm:

- User (người dùng): Người sử dụng SERS, dùng vào việc tạo và quản lý các sự kiện
- Admin (quản trị viên): Người quản lý phần mềm hệ thống, các tài khoản user, và hệ cơ sở dữ liệu của hệ thống.
- Project team (nhóm dự án): Chịu trách nhiệm thiết kế và phát triển hệ thống SERS.
- Notification service (dịch vụ thông báo): Có nhiệm vụ gởi thông báo về các sự kiện đến user.

1.2 Functional Requirements & Non-Functional Requirements

1. Functional requirements:
   - **FR-01:** Người dùng có thể tạo một sự kiện, với tiêu đề, mô tả, ngày giờ, địa điểm, tần suất lặp lại (mỗi ngày, mỗi tuần, mỗi tháng).
   - **FR-02:** Người dùng có thể thêm hoặc cập nhật các chi tiết của một sự kiện.
   - **FR-03:** Người dùng có thể xóa sự kiện bất kỳ.
   - **FR-04:** Người dùng có thể xem danh sách các sự kiện sắp tới.
   - **FR-05:** Người dùng sẽ nhận được thông báo nhắc nhở từ hệ thống trước khi sự kiện diễn ra (15 phút).
   - **FR-06:** Cho phép người dùng tạo liên kết URL để chia sẻ sự kiện cho người dùng khác xem.
2. Non-functional requirements:
   - **NFR-01:** Giao diện cần dễ hiểu, dễ sử dụng để cho một người dùng mới có thể nhanh chóng đăng nhập và tạo sự kiện mà không cần đọc hướng dẫn.
   - **NFR-02:** Hệ thống cần hiển thị danh sách các sự kiện trong vòng 1 giây khi người dùng có dưới 100 sự kiện trong database.
   - **NFR-03:** Dịch vụ thông báo cần gửi nhắc nhở đến người dùng 15 phút trước khi sự kiện bắt đầu, với tỉ lệ gửi thông báo thành công 99%.

1.3 User stories

- **US-01:** *Là một* sinh viên, *tôi muốn* **tạo** và **xem** danh sách hạn nộp bài tập sắp tới *để* sắp xếp thời gian học tập tốt hơn.
- **US-02:** *Là một* gia sư, *tôi muốn* **nhận thông báo** kịp thời về các buổi dạy kèm *để* không quên lịch dạy của tôi.
- **US-04:** *Là một* bệnh nhân, *tôi muốn* có nhắc nhở **lặp lại** hằng tuần *để* giúp tôi nhớ loại thuốc cần uống trong ngày.
- **US-03:** *Là một* thành viên câu lạc bộ, *tôi muốn* **chia sẻ** sự kiện cá nhân cho bạn bè *để* mọi người có thể biết đến thời gian và địa điểm của sự kiện và tham gia.

- **US-05:** *Là một* nhân viên văn phòng, *tôi muốn* đồng bộ dữ liệu về sự kiện của mình trong hệ thống lên Google Calendar *để* xem lịch tiện lợi hơn

1.4 Acceptance criteria & MoSCoW

Dựa theo phương pháp MoSCoW [2], ta xếp loại độ ưu tiên các user story thành 1 trong 4 nhóm: Must have, Should have, Could have, Won't have.

ID	User Story	MoSCoW	Acceptance Criteria
US-01	Tạo và xem danh sách bài tập (Sinh viên)	Must Have	(1) Hệ thống hiển thị form nhập liệu với các trường: Tiêu đề, Mô tả, Ngày giờ, Địa điểm. (2) Nút "Lưu" bị vô hiệu hóa hoặc báo lỗi nếu bỏ trống trường "Tiêu đề" hoặc "Ngày giờ". (3) Sau khi lưu thành công, sự kiện mới phải xuất hiện ngay lập tức trong danh sách "Sắp tới". (4) Ngày giờ của sự kiện phải là một thời điểm trong tương lai.
US-02	Nhận thông báo dạy kèm (Gia sư)	Must Have	(1) Hệ thống gửi thông báo đúng 15 phút trước giờ sự kiện. (2) Nội dung thông báo hiển thị đúng tiêu đề của sự kiện. (3) Nếu người dùng đang offline, thông báo hiển thị khi họ đăng nhập lại.
US-04	Nhắc nhở lặp lại uống thuốc (Bệnh nhân)	Should Have	(1) Trong form tạo sự kiện có tùy chọn "Lặp lại": Hàng ngày, Hàng tuần. (2) Nếu chọn "Hàng tuần", hệ thống tự động tạo các bản sao sự kiện vào cùng giờ các ngày tiếp theo. (3) Khi xóa một sự kiện lặp lại, hệ thống hỏi người dùng muốn xóa "Chỉ sự kiện này" hay "Tất cả chuỗi".
US-03	Chia sẻ sự kiện (Thành viên CLB)	Could Have	(1) Màn hình chi tiết sự kiện có nút "Chia sẻ" hoặc "Get Link". (2) Hệ thống sinh ra URL duy nhất (ví dụ: sers.com/share/xyz123). (3) Khi truy cập URL không cần đăng nhập, người xem thấy Tiêu đề, Thời gian, Địa điểm (Read-only).
US-05	Đồng bộ Google Calendar (Nhân viên VP)	Won't Have	(1) Người dùng thấy nút "Sync to Google". (2) Hệ thống yêu cầu quyền truy cập (Mockup OAuth). (3) Sự kiện được gửi thành công đến API giả lập và trả về trạng thái "Synced".

2 Process Model and Planning

Nhóm quyết định lựa chọn mô hình Scrum để phát triển hệ thống SERS.

Lý do lựa chọn:

- Scrum phù hợp với nhóm 5 thành viên, giúp tăng cường giao tiếp và tính minh bạch thông qua các sự kiện Daily Scrum (dù là online hay offline).
- Thời gian phát triển ngắn (vài tuần) đòi hỏi khả năng phản hồi, thích ứng nhanh với các thay đổi. Việc chia nhỏ dự án thành các Sprint 1 tuần giúp nhóm kiểm soát tiến độ tốt hơn so với mô hình Waterfall.
- Hệ thống được định nghĩa dựa trên các User Stories (như đã phân tích ở phần Requirements), tương thích với Product Backlog của Scrum.

Dự án kéo dài 2 tuần, được chia làm 2 Sprints.

- **Sprint 1 (Tuần 1):** Tập trung vào các tính năng cốt lõi (Core Features) - MVP.
- **Sprint 2 (Tuần 2):** Hoàn thiện các tính năng nâng cao, Notification và Refactoring.

2.1 Product Backlog & Planning Poker Estimation

Nhóm sử dụng kỹ thuật Planning Poker [7] để ước lượng độ phức tạp (Effort) cho từng User Story theo dãy số Fibonacci (0, 1, 2, 3, 5, 8, 13, 21).

ID	User Story	Priority	Story Points
US-01	Sinh viên tạo và xem danh sách sự kiện (CRUD)	Must Have	8
US-02	Gia sư nhận thông báo nhắc nhở (Notification)	Must Have	8
US-04	Bệnh nhân tạo nhắc nhở lặp lại (Recurring Events)	Should Have	5
US-03	Chia sẻ sự kiện qua URL (Sharing)	Could Have	3
US-05	Đồng bộ Google Calendar	Won't Have	-
		Total Points	**24**

Bảng 2: Product Backlog với ước lượng điểm

2.2 Sprint Backlogs

Sprint 1: Core Mechanics (Velocity dự kiến: 13 points)

Mục tiêu: Người dùng có thể tạo, lưu trữ và xem danh sách sự kiện.

- **US-01 (8 points):** Thiết kế Database (Mock), API thêm/xóa/sửa sự kiện, Giao diện danh sách.
- **Part of US-04 (5 points):** Xử lý logic lặp lại cơ bản (Backend logic).

Sprint 2: Notification & Polish (Velocity dự kiến: 11 points)

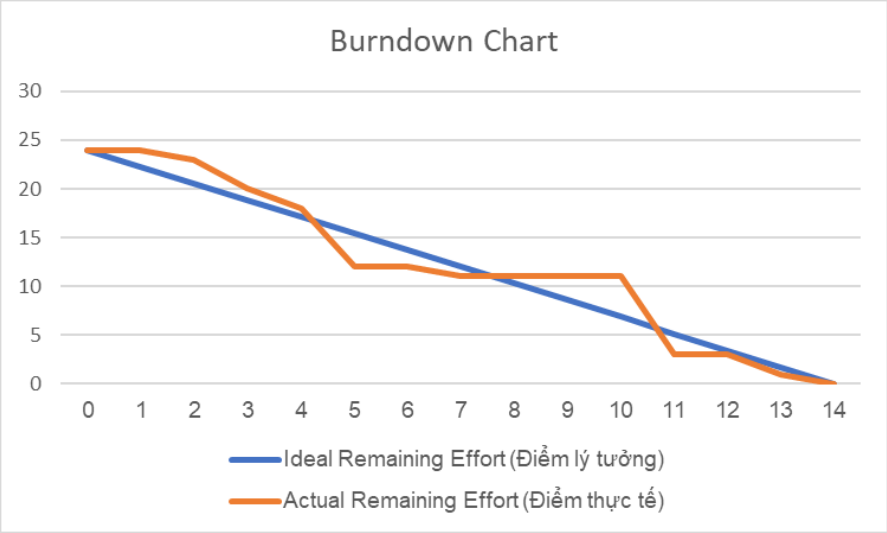
Mục tiêu: Hoàn thiện hệ thống nhắc nhở và tính năng chia sẻ.

- **US-02 (8 points):** Cài đặt Background Service kiểm tra giờ và gửi thông báo (Console log/Popup).

- **US-03 (3 points):** Tạo trang xem sự kiện public (Read-only view).
- **Testing & Documentation:** Viết test case và hoàn thiện báo cáo.

2.3 Burndown Chart (Simulated)

Biểu đồ Burndown [4] dự kiến cho thấy sự giảm dần của Story Points qua 14 ngày.



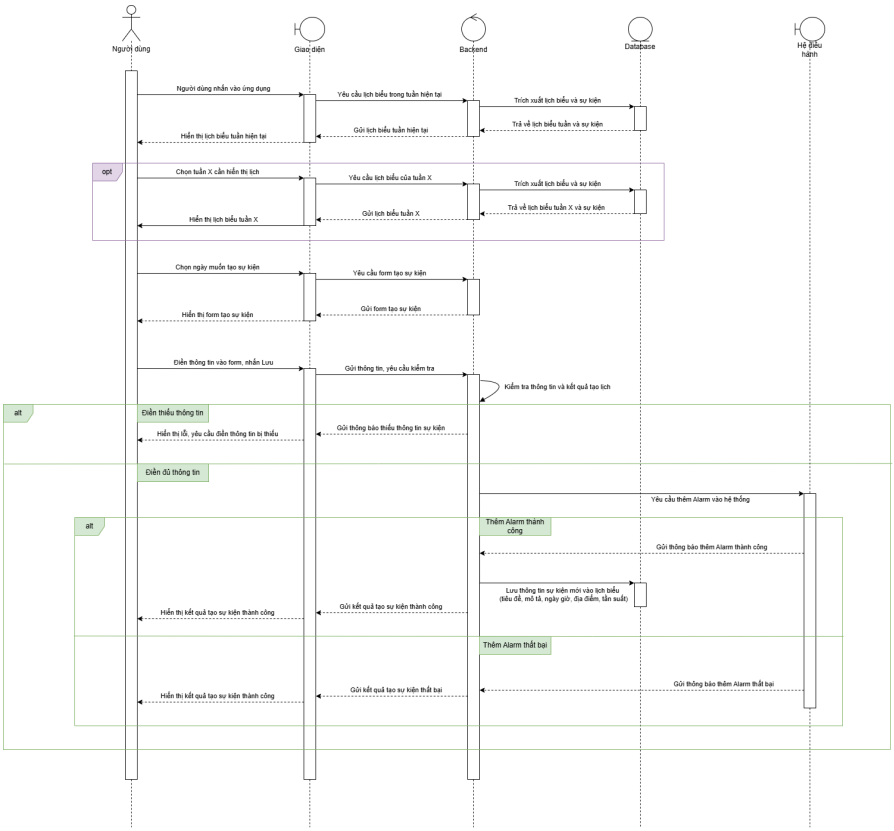
Hình 2: Burndown Chart cho 2 Sprints

3 System Modeling

3.1 Sequence diagram

Bên dưới là sơ đồ tuần tự cho usecase người dùng thêm một sự kiện mới vào lịch. Khi đó, phần mềm sẽ hiển thị biểu mẫu cho người dùng nhập thông tin, sau đó tiến hành tạo nhắc lịch và lưu thông tin sự kiện, cụ thể như sau:

- Khi người dùng nhấn vào ứng dụng, ứng dụng sẽ truy xuất và hiển thị lịch biểu cũng như các sự kiện có trong lịch biểu đó. Người dùng cũng có thể lựa chọn chuyển lịch biểu sang tuần khác tùy theo nhu cầu của họ.
- Khi ở giao diện hiển thị lịch, nếu người dùng muốn tạo một sự kiện mới, người dùng sẽ nhấn vào ngày muốn tạo. Khi đó, ứng dụng sẽ hiển thị ra biểu mẫu điền thông tin sự kiện, bao gồm các thông tin như: Tiêu đề, thời gian, địa điểm, tần suất....
- Sau khi người dùng nhập thông tin và nhấn "Lưu", ứng dụng sẽ lấy thông tin đó để tạo Alarm và đồng thời cũng sẽ lưu thông tin sự kiện vào database.
- Cuối cùng, người dùng sẽ nhận được hiển thị "Thêm sự kiện thành công" và chuyển về giao diện lịch biểu ban đầu.



Hình 3: Sequence diagram Thêm sự kiện vào lịch

3.2 State diagram

Sau đây là State diagram thể hiện vòng đời của một sự kiện. Một sự kiện có 5 trạng thái (state) là **Created**, **Scheduled**, **Notified**, và **Completed / Cancelled** - tương ứng với **Final State**.

Ban đầu, khi một sự kiện được tạo thành công bởi người dùng, sẽ bao gồm nhiều thông tin, như là tiêu đề, thời gian, địa điểm, tần suất, ... (Created State) . Khi người dùng chọn điều chỉnh các thông tin của sự kiện tương ứng, hệ thống sẽ cho phép đổi các thông tin đó, nhưng vẫn giữ đối tượng tương ứng.

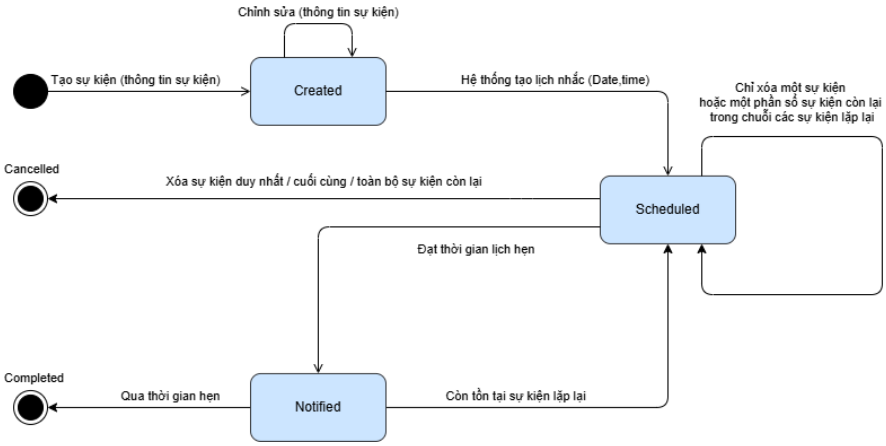
Sau khi người dùng tạo sự kiện thành công, hệ thống sẽ tự chuyển đổi sự kiện sang dạng chờ để nhắc hẹn (Scheduled State), dựa trên thời gian đã đặt lịch sẵn. Khi sự kiện sắp tới thời gian bắt đầu, hệ thống sẽ tự động thiết lập thông báo cho người dùng Notified State.

Sau khi sự kiện kết thúc, sẽ có hai luồng xử lý chính:

- Nếu sự kiện vẫn còn tồn tại các phiên bản lặp lại trong tương lai, sự kiện sẽ được chuyển về ngược lại trạng thái chờ để nhắc hẹn (Scheduled State).
- Nếu sự kiện đã hết thời hạn, sự kiện này sẽ được chuyển sang trạng thái kết thúc (Completed State - tương ứng với Final State).

Một tính năng quan trọng mà hệ thống cung cấp cho người dùng là khả năng xóa các sự kiện. Hệ thống có hai luồng xử lý chính:

- Nếu người dùng chỉ xóa một sự kiện trong nhiều sự kiện còn lại, sự kiện sẽ được reset lại về trạng thái chờ nhắc hẹn, với thời gian phù hợp (Scheduled State).
- Nếu người dùng xóa sự kiện duy nhất còn lại - tức là cuối cùng, hoặc tất cả chuỗi sự kiện còn lại từ thời điểm hiện tại, sự kiện này sẽ được cho kết thúc, về sang trạng thái Cancelled State - tương ứng với Final State.

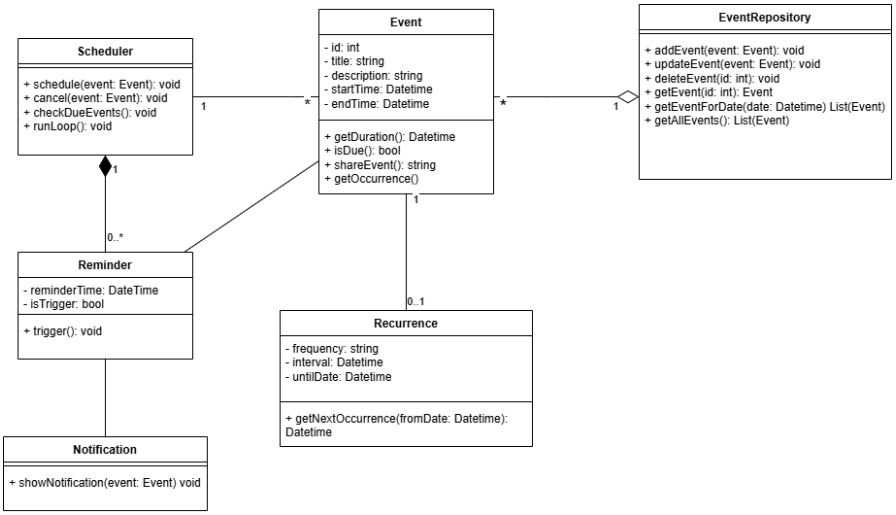


Hình 4: State diagram của sự kiện

3.3 Class diagram

Cuối cùng, đây là class diagram cho phần mềm SERS. Diagram gồm 6 class sau:

- **Event**: Chứa các thông tin về sự kiện như tiêu đề, thời gian, mô tả. Event sẽ được lưu trữ trong database và chỉ lưu trữ 1 lần đối với 1 sự kiện (kể cả sự kiện đó có tính lặp lại). Việc xử lý quy tắc lặp lại của event sẽ do class bên dưới đảm nhận.
- **Recurrence**: Mô tả quy tắc lặp lại của một Event. Tùy vào việc Event có lặp lại hay không, Event sẽ chứa Recurrence tương ứng.
- **EventRepository**: Lưu trữ danh sách các event vào database và cung cấp nó cho ứng dụng. Class scheduler sẽ tiến hành lấy danh sách các sự kiện từ EventRepository này.
- **Scheduler**: Tính toán và tạo lịch nhắc dựa trên Event và Recurrence. Thông qua logic của các hàm, schedluer sẽ tạo ra các Reminder để tiến hành nhắc nhở.
- **Reminder**: Đại diện cho một nhắc nhở của event. Khi tới giờ cần nhắc nhở, nó sẽ tự động gọi notification để gửi thông báo đến người dùng.
- **Notification**: Đây là phần chịu trách nhiệm hiển thị thông báo cho người dùng.



Hình 5: Class diagram

4 Architecture Design

Vì mục tiêu của nhóm chỉ là xây dựng một ứng dụng nhắc lịch sự kiện chạy cục bộ (local) với quy mô nhỏ và đơn giản, nhóm quyết định lựa chọn **Kiến trúc Phân lớp (Layered Architecture)** cho phần mềm SERS. Mô hình này chia ứng dụng thành các lớp riêng biệt, mỗi lớp đảm nhận một trách nhiệm cụ thể, giúp mã nguồn rõ ràng và dễ bảo trì. Dưới đây là architecture diagram của phần mềm:

Kiến trúc của SERS bao gồm 3 lớp chính là **Lớp Giao diện**, **Lớp Nghiệp vụ**, **Lớp truy cập dữ liệu**. Vai trò của từng lớp như sau:

- **Presentation Layer (Lớp Giao diện):** Đây là lớp tương tác trực tiếp với người dùng. Nhiệm vụ của nó là hiển thị lịch và các sự kiện, các form nhập liệu và các thông báo nhắc nhở (notification). Lớp này nhận yêu cầu từ người dùng và chuyển xuống lớp nghiệp vụ để xử lý, sau đó hiển thị kết quả trả về.
- **Business Logic Layer (Lớp Nghiệp vụ):** Đây là "bộ não" của ứng dụng. Lớp này chứa các logic cốt lõi như: tính toán thời gian nhắc nhở, xử lý logic thêm/sửa/xóa sự kiện, và kích hoạt cơ chế thông báo khi đến giờ. Nó không quan tâm dữ liệu được lưu trữ như thế nào hay hiển thị ra sao.
- **Data Access Layer (Lớp Truy cập Dữ liệu):** Lớp này chịu trách nhiệm giao tiếp với cơ sở dữ liệu cục bộ (ví dụ: SQLite hoặc file JSON/XML). Nhiệm vụ của nó là thực hiện các thao tác đọc, ghi, truy vấn dữ liệu sự kiện và cấu hình người dùng một cách an toàn và chính xác.

Với kiến trúc 3 lớp như trên, luồng dữ liệu trong SERS sẽ tuân theo quy tắc gọi từ trên xuống (Top-down) quen thuộc:

1. Người dùng thao tác trên GUI (ví dụ: tạo một sự kiện mới).
2. **Presentation Layer** thu thập dữ liệu đầu vào và gửi yêu cầu xuống **Business Logic Layer**.
3. **Business Logic Layer** kiểm tra tính hợp lệ của dữ liệu (validate) và thực hiện các xử lý nghiệp vụ cần thiết.
4. Nếu dữ liệu hợp lệ, yêu cầu lưu trữ được chuyển xuống **Data Access Layer**.
5. **Data Access Layer** thực hiện ghi dữ liệu vào bộ nhớ cục bộ và trả về kết quả thành công/thất bại ngược lên trên theo chuỗi gọi để thông báo cho người dùng.

Nhóm quyết định chọn kiến trúc phân lớp (Layered Architecture) vì 3 lý do chính sau:

- **Sự đơn giản (Simplicity):** Với mục tiêu là phần mềm chạy local đơn giản, kiến trúc này dễ triển khai và không đòi hỏi cấu hình phức tạp như Microservices.
- **Phân tách trách nhiệm (Separation of Concerns):** Việc tách biệt giao diện, logic và dữ liệu giúp việc phát triển và sửa lỗi dễ dàng hơn.
- **Dễ dàng kiểm thử (Testability):** Có thể kiểm thử riêng biệt lớp nghiệp vụ mà không cần phụ thuộc vào giao diện người dùng.

5 Implementation and Code Quality

6 Software Testing and Quality Assurance

7 Requirements Engineering

Tài liệu tham khảo

[1] Atlassian. *Product backlog vs. Sprint backlog: Top Differences*. https://www.atlassian.com/agile/project-management/sprint-backlog-product-backlog. Accessed: December 8, 2025. 2025.

[2] GeeksforGeeks. *MoSCoW Prioritization Technique in Product Management*. https://www.geeksforgeeks.org/product-management/moscow-prioritization-in-product-management/. Last updated: 18 Oct, 2025. 2025.

[3] IBM. *Use Case Specification Outline*. 2025. URL: https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors-next/7.0.3?topic=cases-use-case-specification-outline (visited on 10/02/2025).

[4] Megan Keup. *Burndown Chart: What Is It, Examples & How to Use One for Agile*. https://www.projectmanager.com/blog/burndown-chart-what-is-it. Published: Apr 1, 2025. 2025.

[5] KevinStratvert. *Figma Tutorial for Beginners*. 2025. URL: https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors-next/7.0.3?topic=cases-use-case-specification-outline (visited on 10/02/2025).

[6] Visual Paradigm. *What is Use Case Specification?* 2025. URL: https://www.visual-paradigm.com/guide/use-case/what-is-use-case-specification/ (visited on 10/02/2025).

[7] Mountain Goat Software. *Planning Poker: An Agile Estimating and Planning Technique*. https://www.mountaingoatsoftware.com/agile/planning-poker. Last updated: September 4, 2025. 2025.