Ex01:

1. Ước lượng Story Points: *5 điểm Story*

2. Lý do chọn 5 Story Points

Tôi chọn **5 điểm Story** vì mức độ phức tạp của User Story này nằm ở **mức trung bình**:

🔶 Lý do chi tiết:

1. Có nhiều bước thực hiện

- Cần tạo giao diện để nhập thông tin công việc (tiêu đề, mô tả, deadline…)
- Cần xử lý dữ liệu khi người dùng nhấn tạo công việc
- Cần hiển thị thông báo khi tạo thành công

2. Độ phức tạp vừa phải

- Không quá đơn giản như 1–2 điểm (chỉ hiển thị hoặc thao tác nhỏ)
- Không quá phức tạp như 8–13 điểm (không có logic phức tạp, không liên quan nhiều hệ thống khác)

3. Có một chút rủi ro

- Dữ liệu nhập có thể sai, cần kiểm tra
- Giao diện và logic cần phối hợp đúng

Nhìn chung, nhiệm vụ này **vừa phải**, không khó nhưng cũng không quá dễ, phù hợp với **5 điểm Story** trong dãy Fibonacci.

3. Mức độ phù hợp với Sprint

- User Story rõ ràng mục tiêu và lợi ích với người dùng
- Dễ hiểu, không dùng thuật ngữ khó
- Đủ chi tiết để đội phát triển ước lượng và đưa vào Sprint

Ex02:

1. Ước lượng Story Points: *8 điểm Story*

2. Lý do chọn 8 Story Points

Tôi chọn **8 điểm Story** vì tính năng thanh toán đơn hàng có **mức độ phức tạp cao hơn mức trung bình**, gồm nhiều bước liên quan đến xử lý dữ liệu và kiểm tra thông tin.

Lý do chi tiết

1. Nhiều bước xử lý

- Kiểm tra thông tin giỏ hàng
- Nhập thông tin người mua (địa chỉ, số điện thoại…)
- Chọn phương thức thanh toán (COD, chuyển khoản…)
- Xác nhận đơn hàng và tạo hóa đơn

2. Độ phức tạp cao

- Cần kiểm tra dữ liệu đầu vào
- Cần xác nhận tổng tiền đã tính chính xác
- Có thể phải kết nối với cổng thanh toán (nếu có)

3. Rủi ro cao hơn

- Dễ xảy ra lỗi khi xử lý đơn hàng
- Thông tin thanh toán phải chính xác
- Có nhiều trường hợp ngoại lệ (thiếu hàng, sai thông tin…)

4. Không quá lớn đến mức 13 SP

- Tính năng này không yêu cầu thuật toán phức tạp
- Không cần tích hợp nhiều hệ thống bên ngoài (nếu chỉ là bài cơ bản)

Vì vậy, mức **8 Story Points** phù hợp nhất để thể hiện độ phức tạp và khối lượng công việc.

3. Mức độ phù hợp với Sprint

- User Story rõ ràng về mục tiêu và lợi ích
- Người dùng dễ hiểu và nắm được mục đích
- Có thể đưa vào Sprint để đội phát triển phân rã thành các task

Ex03:

1. Các thành viên chọn điểm Story

Giả sử nhóm có 4 thành viên và mỗi người chọn 1 lá bài từ dãy Fibonacci:

Thành viên	Điểm Story chọn
Thành viên A	5
Thành viên B	8
Thành viên C	8
Thành viên D	13

2. Thảo luận và giải thích sự khác biệt

Thành viên A chọn 5 điểm

- Nghĩ rằng tính năng chỉ cần: tạo giỏ hàng, thêm/xóa sản phẩm, cập nhật số lượng.
  → Công việc ở mức trung bình.

Thành viên B & C chọn 8 điểm

- Cho rằng tính năng hơi phức tạp:
  - Cần tính lại tổng tiền
  - Xử lý khi thêm/spam nút thêm sản phẩm
  - Kiểm tra sản phẩm hết hàng
    → Mức độ phức tạp cao hơn mức trung bình.

Thành viên D chọn 13 điểm

- Cho rằng tính năng liên quan nhiều logic:
  - Kiểm tra tồn kho
  - Đồng bộ giỏ hàng theo tài khoản người dùng
  - Lưu giỏ hàng khi người dùng thoát ứng dụng
    → Rủi ro lớn, nhiều tình huống cần xử lý.

3. Kết luận sau khi thảo luận

Sau khi đội thảo luận, cả nhóm đồng ý rằng:

- Không cần làm các tính năng nâng cao như đồng bộ đa thiết bị hay lưu giỏ hàng tự động (như thành viên D lo lắng).
- Nhưng giỏ hàng vẫn có độ phức tạp hơn mức trung bình (như thành viên A đánh giá).

Do đó, nhóm thống nhất chọn: *8 Story Points*.

4. Lý do chọn 8 Story Points

- Có nhiều chức năng con: thêm sản phẩm, xóa sản phẩm, thay đổi số lượng.
- Phải tính toán tổng tiền và hiển thị lại ngay lập tức.
- Cần kiểm tra tình trạng sản phẩm: còn hàng / hết hàng.
- Mức độ phức tạp cao nhưng chưa đến mức rất lớn như 13 SP.

Ex04:

User Story

"Là người dùng, tôi muốn đăng nhập vào ứng dụng ngân hàng trực tuyến để có thể truy cập tài khoản của mình một cách an toàn."

1. Các thành viên chọn điểm Story

Giả sử nhóm có 4 thành viên và mỗi người chọn 1 lá bài trong dãy Fibonacci:

Thành viên	Điểm Story chọn
Thành viên A	3
Thành viên B	5
Thành viên C	8
Thành viên D	5

2. Thảo luận và giải thích sự khác biệt

Thành viên A chọn 3 điểm

- Cho rằng tính năng đăng nhập khá đơn giản: nhập username & password, kiểm tra thông tin, thông báo kết quả.

Thành viên B chọn 5 điểm

- Nhận định rằng đăng nhập trong ứng dụng ngân hàng phải xử lý nhiều bước hơn bình thường:
    - Kiểm tra đa lớp (multi-layer validation)
    - Bảo mật cao hơn các ứng dụng thông thường
    - Kiểm soát mật khẩu sai quá số lần quy định

Thành viên C chọn 8 điểm

- Cho rằng việc đăng nhập ngân hàng rất phức tạp vì có thể bao gồm:
    - Xác thực 2 lớp (OTP)
    - Khóa tạm thời tài khoản khi nhập sai nhiều lần
    - Mã hóa dữ liệu đăng nhập
- Nhiều rủi ro → ước lượng cao hơn.

Thành viên D chọn 5 điểm

- Đồng ý rằng cần nhiều bước kiểm tra bảo mật nhưng không đến mức quá phức tạp như thành viên C nghĩ.

3. Kết luận sau khi thảo luận

- Nhóm thống nhất rằng **tính năng đăng nhập ngân hàng có độ phức tạp cao hơn đăng nhập thông thường**, nhưng không cần thực hiện hết các yếu tố nâng cao như mã hóa cấp độ hệ thống hoặc xử lý đa nền tảng.
- Hai thành viên chọn 5 SP và lập luận hợp lý, cân bằng giữa độ phức tạp và phạm vi của User Story.

Kết quả cuối cùng: *5 Story Points*.

4. Lý do chọn 5 Story Points

- Có nhiều bước kiểm tra hơn đăng nhập thông thường (ví dụ: giới hạn sai mật khẩu, bảo mật).
- Có thể phải xử lý xác thực thêm (ví dụ: OTP).
- Có rủi ro về bảo mật và dữ liệu.
- Tuy nhiên, phạm vi User Story chỉ là **"tạo tính năng đăng nhập"**, không bao gồm tất cả nghiệp vụ phức tạp của ngân hàng → không quá lớn để lên 8 SP.

Ex05:

1. Product Backlog mẫu cho ứng dụng quản lý công việc

Dựa trên đề bài, nhóm xây dựng danh sách các User Story và ước lượng Story Points:

User Story	Mô tả	Story Points
US1	Tạo công việc mới	5
US2	Chỉnh sửa công việc	3
US3	Xóa công việc	2
US4	Hiển thị danh sách công việc	3
US5	Tìm kiếm công việc	5
US6	Đăng nhập hệ thống	5
US7	Đồng bộ dữ liệu tài khoản	8

2. Planning Poker – Quy trình thảo luận

Nhóm thực hiện Planning Poker để ước lượng mức độ phức tạp.

Ví dụ quá trình ước lượng cho User Story "Tạo công việc mới"

Thành viên	Lá bài chọn	Lý do
A	5	Cần UI + xử lý dữ liệu
B	8	Có thể thêm validate nâng cao
C	5	Phức tạp trung bình
D	5	Không nhiều rủi ro

Kết quả cuối cùng: **5 Story Points**

Nhóm thực hiện tương tự cho các User Stories khác và thống nhất như bảng ở trên.

3. Mục tiêu Sprint

Dựa trên độ ưu tiên + khả năng thực hiện, nhóm đưa ra mục tiêu cho Sprint:

🎯 Mục tiêu Sprint:

"Hoàn thành các chức năng cốt lõi: tạo công việc, chỉnh sửa công việc, xóa công việc và đăng nhập hệ thống."

4. Phân chia User Story thành các Tasks

US1 – Tạo công việc mới (5 SP)

Task	Ước lượng (giờ)
T1: Thiết kế giao diện form	4h
T2: Xử lý thêm công việc mới	6h
T3: Validate dữ liệu	3h
T4: Test & sửa lỗi	3h

US2 – Chỉnh sửa công việc (3 SP)

Task	Giờ
T1: UI chỉnh sửa	3h
T2: Xử lý update	4h
T3: Kiểm tra dữ liệu	2h

US3 – Xóa công việc (2 SP)

Task	Giờ
T1: Xóa công việc	3h
T2: Xử lý confirm xóa	2h

US6 – Đăng nhập hệ thống (5 SP)

Task	Giờ
T1: Tạo UI đăng nhập	4h
T2: Xử lý đăng nhập	5h
T3: Kiểm tra tài khoản sai / đúng	3h
T4: Test & sửa lỗi	3h

5. Sử dụng Velocity để lập kế hoạch Sprint

Giả sử **Velocity của Sprint trước = 13 Story Points**.

→ Nghĩa là nhóm có khả năng hoàn thành khoảng **13 SP** trong Sprint tiếp theo.

Tổng điểm các User Story được chọn:

- US1: 5 SP
- US2: 3 SP
- US3: 2 SP
- US6: 5 SP

Tổng SP cần cho Sprint = 15 SP

Nhưng Velocity chỉ cho phép 13 SP → nhóm cần điều chỉnh.

Quyết định cuối cùng:

- Giữ: US1 (5 SP) + US2 (3 SP) + US3 (2 SP) + US6 (5 SP)
  → Tổng = **15 SP**

Nhóm thống nhất vẫn giữ 15 SP vì:

- Một vài tasks có thể hoàn thành nhanh hơn ước lượng.
- Đội có thêm 1 thành viên mới hỗ trợ nên khả năng hoàn thành cao hơn.

Hoặc nếu muốn an toàn hơn, nhóm có thể **loại US6 (5 SP)** và giữ 10 SP.

Ex06:

1. Mục tiêu Sprint

Mục tiêu Sprint:
"Hoàn thành các tính năng quản lý công việc cơ bản cho người dùng, bao gồm tạo công việc mới và xem danh sách công việc."

Lý do chọn mục tiêu này:

- Đây là những chức năng cốt lõi nhất của ứng dụng quản lý công việc.
- Mang lại giá trị trực tiếp cho người dùng ngay trong Sprint đầu tiên.
- Dễ triển khai, ít phụ thuộc vào các chức năng khác.

2. Chọn User Stories từ Product Backlog

User Story 1 – Tạo công việc mới

- **Là người dùng**, tôi muốn **tạo một công việc mới**, để quản lý các việc cần làm.
- **Lý do chọn:** Đây là chức năng thiết yếu, giúp người dùng bắt đầu sử dụng ứng dụng.

User Story 2 – Hiển thị danh sách công việc

- **Là người dùng**, tôi muốn **xem danh sách công việc**, để theo dõi các công việc đã tạo.

- **Lý do chọn:** Gắn liền với User Story 1, và giúp người dùng quản lý các công việc đã nhập.

2 User Story này **kết hợp hoàn chỉnh** một luồng chức năng: *Tạo* → *Xem*.

3. Phân chia User Stories thành các Tasks nhỏ

User Story 1: Tạo công việc mới

Task	Mô tả
T1	Thiết kế giao diện form tạo công việc
T2	Xử lý logic thêm công việc vào hệ thống
T3	Validate dữ liệu (không được để trống, độ dài hợp lệ)
T4	Hiển thị thông báo tạo thành công/thất bại
T5	Test và sửa lỗi

User Story 2: Hiển thị danh sách công việc

Task	Mô tả
T1	Tạo giao diện danh sách công việc
T2	Kết nối API để lấy danh sách công việc
T3	Hiển thị danh sách theo thứ tự thời gian
T4	Thêm loading/empty state nếu không có dữ liệu
T5	Test và sửa lỗi

4. Ước lượng công việc theo T-shirt Sizes

User Story 1 – Tạo công việc mới → Size: M (Medium)

Lý do:

- Có nhiều task nhỏ như UI, xử lý logic, validate.
- Độ phức tạp trung bình, rủi ro thấp.

User Story 2 – Hiển thị danh sách công việc → Size: S (Small)

Lý do:

- Chủ yếu là lấy dữ liệu và hiển thị lên giao diện.
- Ít logic phức tạp, thời gian thực hiện ngắn.

Ex07:

1. Chia nhỏ User Story thành các Task

Task 1: Thiết kế giao diện form đăng ký

- Tạo giao diện gồm các trường: Họ tên, Email, Mật khẩu, Xác nhận mật khẩu.
- Thêm nút "Đăng ký" và liên kết "Đăng nhập".

Task 2: Xử lý validate dữ liệu trên giao diện

- Kiểm tra email hợp lệ, mật khẩu trùng khớp, các trường không trống.
- Hiển thị lỗi tương ứng cho từng trường.

Task 3: Tạo API đăng ký tài khoản

- Xây dựng API nhận dữ liệu từ form.
- Kiểm tra trùng email, mã hóa mật khẩu, lưu dữ liệu vào database.

Task 4: Kết nối giao diện với API đăng ký

- Gửi dữ liệu từ form tới API.
- Nhận phản hồi và hiển thị thông báo thành công/thất bại.

Task 5: Tạo màn hình thông báo và điều hướng

- Hiển thị thông báo "Đăng ký thành công".
- Điều hướng người dùng sang trang đăng nhập hoặc Dashboard.

Task 6: Viết test và sửa lỗi

- Kiểm thử form, API, validate và các luồng hoạt động.
- Sửa lỗi trước khi đưa vào Sprint Demo.

2. Lý do lựa chọn các task

- Các task được chia theo **quy trình phát triển thực tế**: UI → Validate → API → Kết nối → Thông báo → Test.
- Mỗi task **độc lập**, có thể gán cho từng thành viên trong nhóm Scrum.
- Task nhỏ giúp **dễ quản lý tiến độ**, tránh quá tải cho một Sprint.
- Đảm bảo toàn bộ luồng đăng ký từ **giao diện → xử lý → lưu dữ liệu → thông báo** đều được triển khai đầy đủ.

3. Ước lượng các task (T-shirt Size)

Task	Ước lượng	Lý do
Task 1: Thiết kế giao diện	**S (Small)**	Giao diện đơn giản, ít thành phần
Task 2: Validate dữ liệu	**M (Medium)**	Cần xử lý logic nhiều trường và hiển thị lỗi
Task 3: Tạo API đăng ký	**M (Medium)**	Liên quan database + mã hóa mật khẩu
Task 4: Kết nối API	**S (Small)**	Gửi request + nhận phản hồi, ít phức tạp
Task 5: Thông báo & điều hướng	**S (Small)**	Logic đơn giản
Task 6: Test và sửa lỗi	**M (Medium)**	Cần test nhiều trường hợp và xử lý lỗi

Ex08:

1. Tính Velocity của nhóm

Sprint 1

- User Story 1: 5 điểm (**Hoàn thành**)
- User Story 2: 8 điểm (**Hoàn thành**)
- User Story 3: 3 điểm (**Không hoàn thành**)

Velocity Sprint 1 = 5 + 8 = 13 điểm

Sprint 2

- User Story 4: 13 điểm (**Hoàn thành**)
- User Story 5: 5 điểm (**Hoàn thành**)
- User Story 6: 8 điểm (**Không hoàn thành**)

Velocity Sprint 2 = 13 + 5 = 18 điểm

Velocity trung bình

Velocity truung bình = (13+18) / 2=15.5≈16 điểm

Velocity trung bình của nhóm ≈ 16 điểm/Sprint

2. Áp dụng Velocity để lập kế hoạch Sprint tiếp theo

Giả sử Product Backlog có các User Stories sau:

User Story	Điểm
User Story A	5
User Story B	8
User Story C	3
User Story D	13
User Story E	2

Với **Velocity = 16**, ta chọn các User Stories sao cho:

Tổng điểm ≤ 16
Công việc vẫn đủ quan trọng và phù hợp mục tiêu Sprint

User Stories được chọn

1. User Story B – 8 điểm

Chức năng lớn, quan trọng, phù hợp ưu tiên.

2. User Story A – 5 điểm

Gắn liền với B, giúp hoàn thiện tính năng.

3. User Story C – 3 điểm

Nhỏ, phù hợp để lấp đầy Velocity.

Tổng điểm Sprint:

8+5+3=168 + 5 + 3 = 168+5+3=16

Khớp hoàn hảo Velocity trung bình 16 điểm

Ex09:

1. Ước lượng công việc bằng Story Points

- **User Story được chọn:** "Tạo công việc mới"
- **Story Points ước lượng:** 5 điểm
- **Lý do chọn 5 điểm:**
  - User Story này yêu cầu tạo giao diện thêm công việc, kiểm tra dữ liệu đầu vào, lưu vào database và hiển thị trên danh sách công việc.
  - Độ phức tạp trung bình: không quá khó, nhưng có nhiều bước liên quan đến UI và backend.
  - Theo thang Fibonacci (1, 2, 3, 5, 8, 13), mức 5 là hợp lý cho trung bình – hơi phức tạp nhưng không quá lớn.

2. Planning Poker

- **Các User Stories và ước lượng nhóm:**
  1. "Tạo công việc mới" – 5 điểm (như trên)
  2. "Quản lý danh sách công việc" – 8 điểm (cần sắp xếp, lọc, xoá, cập nhật trạng thái)
  3. "Gửi thông báo khi công việc sắp hết hạn" – 3 điểm (phần logic đơn giản, chỉ trigger alert)
- **Thảo luận:**
  - Nhóm có thể có sự khác biệt trong đánh giá mức độ phức tạp: UI dễ nhưng backend phức tạp hơn.
  - Sau khi thảo luận, thống nhất số điểm cho từng story dựa trên kinh nghiệm và độ phức tạp.

3. Lập kế hoạch Sprint dựa trên Velocity

- **Giả sử Velocity của nhóm:** 13 điểm/ Sprint
- **Chọn các User Stories phù hợp:**
  - "Tạo công việc mới" – 5 điểm
  - "Quản lý danh sách công việc" – 8 điểm
  - Tổng = 13 điểm → phù hợp với khả năng nhóm trong 1 Sprint
- **Sprint Plan:**

User Story	Story Points	Task cụ thể
Tạo công việc mới	5	Thiết kế UI, validate input, lưu vào DB, hiển thị danh sách
Quản lý danh sách công việc	8	Tính năng xem, sửa, xoá, lọc, cập nhật trạng thái

- **Nhận xét:**

- Lựa chọn hợp lý, phù hợp với năng lực nhóm.
- Đảm bảo tổng điểm Story trong Sprint không vượt quá Velocity.
- Giải thích rõ ràng lý do lựa chọn từng Story Point giúp minh bạch trong lập kế hoạch.

Ex10:

1. User Story được chọn

- **User Story:** "Tạo tính năng giỏ hàng trong ứng dụng bán hàng trực tuyến"
- **Mô tả:** Người dùng có thể thêm sản phẩm vào giỏ hàng, xem danh sách sản phẩm trong giỏ, cập nhật số lượng, và xóa sản phẩm.

2. Chia nhỏ User Story thành các Task

Task	Mô tả công việc	Story Points	Lý do ước lượng
Tạo giao diện giỏ hàng	Thiết kế form giỏ hàng, hiển thị danh sách sản phẩm	3	Task UI vừa phải, không quá phức tạp
Xử lý API giỏ hàng	Tạo các endpoint: thêm, sửa, xóa, lấy giỏ hàng	5	Backend có logic, validation, database
Thêm sản phẩm vào giỏ	Logic thêm sản phẩm, kiểm tra tồn kho	2	Task đơn giản, ít bước
Cập nhật số lượng sản phẩm	Logic update số lượng sản phẩm trong giỏ	2	Chỉ cần validate và cập nhật database
Xóa sản phẩm khỏi giỏ	Logic xóa sản phẩm và cập nhật UI	2	Task đơn giản, ít rủi ro
Kiểm tra tính năng giỏ hàng	Unit test & QA chức năng thêm, sửa, xóa sản phẩm	3	Phức tạp vừa phải, cần test UI + backend

- **Tổng Story Points:** 17

3. Lập kế hoạch Sprint dựa trên Velocity

- **Giả sử Velocity của nhóm:** 15 điểm/ Sprint
- **Kế hoạch Sprint:**
  - Chọn các task sao cho tổng Story Points ≤ 15
  - **Sprint Plan:**

Task	Story Points	Ghi chú
Tạo giao diện giỏ hàng	3	UI chuẩn, dễ quản lý
Xử lý API giỏ hàng	5	Backend chính
Thêm sản phẩm vào giỏ	2	Task nhỏ

Task	Story Points	Ghi chú
Cập nhật số lượng sản phẩm	2	Task nhỏ
Xóa sản phẩm khỏi giỏ	2	Task nhỏ
Tổng	14	Dưới Velocity, an toàn

- **Các task còn lại (Kiểm tra tính năng giỏ hàng 3 SP) có thể lên Sprint tiếp theo**