

Bài 1:

User Story

“Là người dùng, tôi muốn tạo công việc mới để có thể theo dõi tiến độ công việc.”

Story Points ước lượng: 5 điểm Story (mức độ phức tạp trung bình).

Giải thích lý do

Tôi chọn 5 điểm vì tính năng này có mức phức tạp trung bình khi thực hiện.

Để tạo công việc mới, nhóm cần thực hiện các đầu việc như:

1. Thiết kế giao diện form nhập nội dung công việc.
2. Thực hiện validate dữ liệu (trống, sai định dạng...).
3. Xây dựng API backend để lưu dữ liệu vào cơ sở dữ liệu.
4. Xử lý lỗi khi người dùng nhập sai hoặc API trả về lỗi.
5. Hiển thị thông báo phản hồi sau khi tạo thành công.

Tính năng có một số rủi ro như xử lý quyền người dùng hoặc đồng bộ dữ liệu sau khi tạo công việc, nhưng nhìn chung không quá phức tạp.

Do vậy, mức 5 điểm trong thang Fibonacci phản ánh đúng nỗ lực và độ phức tạp trung bình của User Story này.

Bài 2:

Story Points ước lượng: 13

Vì tính năng thanh toán đơn hàng có độ phức tạp cao, nhiều rủi ro và khối lượng công việc lớn, liên quan đến nhiều hệ thống (backend, gateway, email, inventory).

Bài 3:

1) User Story

“Là người dùng, tôi muốn thêm/sửa/xoá sản phẩm trong giỏ hàng và xem tổng giá để có thể chuẩn bị thanh toán.”

Phạm vi (đã thỏa thuận cho lần ước lượng này):

- Thêm sản phẩm (kèm biến thể: size/color) từ trang sản phẩm.
- Cập nhật số lượng / xoá sản phẩm trong giỏ.
- Hiển thị danh sách item, subtotal, phí vận chuyển tạm (placeholder), tổng tiền.
- Lưu giỏ cho user chưa đăng nhập (localStorage) và merge cơ bản khi login.
- Không bao gồm coupon, voucher, tính phí vận chuyển phức tạp, hay inventory reserve nâng cao.

2) Thành viên tham gia

- Thành viên 1 — Product Owner
- Thành viên 2— Frontend dev
- Thành viên 3— Backend dev
- Thành viên 4— QA
- Thành viên 5— Fullstack / tích hợp

3) Lựa chọn lá bài (chưa mở)

Mỗi người chọn một điểm từ dãy Fibonacci.

- Thành viên 1→ 5
- Thành viên 2→ 8
- Thành viên 3→ 8
- Thành viên 4→ 5
- Thành viên 5→ 13

4) Mở bài & Thảo luận (tóm tắt)

- Thành viên 1(5): Nhìn ở góc độ sản phẩm, đây là tính năng cơ bản — thêm/xóa/update + hiển thị tổng → 5.
- Thành viên 2(8): Frontend có edge-cases (variant, debounce update, merge UI, localStorage) → 8.
- Thành viên 3(8): Backend cần API cho cart, merge logic, xử lý đồng thời → 8.
- Thành viên 4(5): Test case nhiều nhưng không phức tạp logic → 5.
- Thành viên 5(13): Lo ngại scope creep (coupon, inventory, edge-case login/merge) và rollback khi lỗi thanh toán → 13.

Điểm khác biệt chính: PO/QA tập trung vào chức năng cốt lõi; dev lo kỹ thuật, tích hợp, rủi ro concurrency & merge.

5) Thảo luận hướng tới đồng thuận

Nhóm quyết định **thu hẹp scope**: loại bỏ coupon, tính phí nâng cao, và các features liên quan đến inventory reserve. Ena chấp nhận giảm đánh giá xuống 8 sau khi scope bị giới hạn. Alice & Duy chấp nhận điểm 8 khi hiểu rõ rủi ro kỹ thuật.

6) Kết luận đồng thuận

Story Points cuối cùng: 8 điểm

Lý do: Với scope đã thỏa, tính năng là *core cart* nhưng vẫn bao gồm merge khi login + xử lý edge-case frontend/backend → complexity trung bình-cao phù hợp mức 8.

7) Nếu team muốn chia nhỏ (gợi ý)

Nếu 8 quá lớn để đưa vào 1 ticket, chia thành 2 story:

- **Story A — 3 điểm:** Thêm/sửa/xoá sản phẩm ở frontend + localStorage (không merge, không sync server).
- **Story B — 5 điểm:** API cart + merge khi login + đồng bộ danh sách + test tích hợp.

Hoặc chia sâu hơn:

- **3 điểm:** UI add/remove/update (no persistence).
- **5 điểm:** Persist to server + merge + QA/integration.

8) Acceptance Criteria (gợi ý, để QA dùng)

1. Người dùng có thể thêm sản phẩm (với biến thể nếu có) vào giỏ; hiển thị đúng tên/variant/price.
2. Người dùng có thể cập nhật số lượng; subtotal và tổng cập nhật tức thì.
3. Người dùng có thể xóa sản phẩm; subtotal cập nhật.
4. Giỏ lưu trên localStorage nếu chưa đăng nhập; khi đăng nhập, local cart merge với server cart theo rule (ví dụ: cộng quantity cho items trùng).
5. API trả lỗi hợp lệ khi product không tồn tại; frontend hiển thị thông báo lỗi rõ ràng.
6. Xử lý trường hợp update/checkout đồng thời (basic): không xảy ra double-add hoặc data inconsistency rõ rệt.
7. Các test cases chính (add/update/delete/merge) phải pass.

9) Bảng tóm tắt Planning Poker (bản để ghi nhanh)

Thành viên	Vote trước thảo luận	Lý do ngắn
Alice (PO)	5	Focus product: core feature
Binh (FE)	8	Variant, localStorage, merge UI
Cuong (BE)	8	API, merge logic, concurrency
Duy (QA)	5	Test nhiều nhưng logic đơn giản
Ena (Full)	13 → 8	Lo ngại scope creep → giảm sau khi thu hẹp scope

Kết quả cuối: 8 Story Points

Bài 4:

1) User Story

“Là người dùng, tôi muốn đăng nhập vào ứng dụng ngân hàng trực tuyến để có thể truy cập tài khoản và xem thông tin tài chính của mình.”

Phạm vi tính năng (giới hạn cho bài tập này):

- Form đăng nhập: username/password.
- Validate thông tin đầu vào.
- Xác thực với server (API login).
- Xử lý sai mật khẩu, tài khoản bị khóa.
- Lưu token/phiên đăng nhập (JWT/session).
- Auto-logout sau khi hết phiên.
- Bảo mật bắt buộc: mã hóa HTTPS, hạn chế brute-force.
- Không bao gồm: OTP / MFA / FaceID / Reset password (tách user story khác).

2) Thành viên tham gia (ví dụ)

- **Alice – Product Owner**
- **Binh – Frontend Dev**
- **Cuong – Backend Dev**
- **Duy – QA**
- **Ena – Bảo mật / Fullstack**

3) Mỗi thành viên chọn một lá bài (chưa mở)

- Alice → **3**
- Binh → **5**
- Cuong → **8**
- Duy → **5**
- Ena → **13**

4) Mở bài & Giải thích lý do

Alice (3 điểm)

- Tính năng đăng nhập là cơ bản, mọi ứng dụng đều có.
- Chỉ gồm form + API + xử lý lỗi -> đơn giản hóa theo góc nhìn PO.

Binh (5 điểm)

- FE phải làm form, validate, error states, loading states, lưu token, redirect → vừa đủ phức tạp → 5.

Cuong (8 điểm)

- Backend ngân hàng yêu cầu security cao:
 - Kiểm tra số lần đăng nhập sai
 - Lock account
 - Logging
 - Token issuance + refresh
- Không phức tạp như onboarding nhưng vẫn tương đối nặng → 8.

Duy (5 điểm)

- QA cần test nhiều case sai mật khẩu, account locked, expired session → nhưng logic vẫn ổn → 5.

Ena (13 điểm)

- Góc độ bảo mật:
 - Cần chống brute-force
 - Rate limiting
 - Hash password đúng chuẩn (BCrypt/Argon2)
 - Audit logging
- Hệ thống ngân hàng có yêu cầu bảo mật cao → Ena đánh 13 do security risk.

5) Giải thích sự khác biệt

- **PO** nhìn tính năng đơn giản → 3.
- **FE & QA** nhìn workload vừa phải → 5.
- **BE & Security** thấy nhiều rủi ro bảo mật + yêu cầu xử lý backend phức tạp → 8–13.
- Sự chênh lệch chủ yếu đến từ **security vs functional scope**.

6) Thảo luận và điều chỉnh phạm vi (scope)

Nhóm thống nhất phạm vi như sau để ước lượng:

Không bao gồm MFA/OTP

Không bao gồm hệ thống chống fraud nâng cao

Không bao gồm login bằng sinh trắc (FaceID/TouchID)

Chỉ bao gồm đăng nhập cơ bản + hạn chế brute-force + account lock + encryption/password hashing + lưu token

→ Điều này giúp giảm độ phức tạp backend và bảo mật.

Sau khi thảo luận:

- Ena giảm từ 13 → **8**
- Cuong giữ **8**
- Binh & Duy chấp nhận **8** vì backend security quan trọng.
- Alice đồng ý tăng từ 3 → **8** dựa trên rủi ro bảo mật.

7) Kết luận đồng thuận

Story Points cuối cùng: **8 điểm**

Lý do:

- Đăng nhập ngân hàng yêu cầu bảo mật cao hơn các app thông thường.
- Backend có logic lock tài khoản, rate-limit, audit log.
- FE có validate và xử lý lỗi nhiều trạng thái khác nhau.
- QA phải test nhiều case sai mật khẩu, lock, expire token.
- Mức 8 phản ánh đúng mức độ phức tạp trung bình–cao.

8) Bảng tóm tắt Planning Poker (để nộp)

Thành viên	Vote ban đầu	Lý do	Vote cuối
Alice (PO)	3	Nhìn tính năng cơ bản	8
Binh (FE)	5	FE validation + UX	8
Cuong (BE)	8	Logic bảo mật backend	8
Duy (QA)	5	Nhiều case nhưng testable	8
Ena (Security)	13	Bảo mật ngân hàng nhiều yêu cầu	8

Kết quả đồng thuận: **8 Story Points**

Bài 6:

Lập kế hoạch Sprint cơ bản — hoàn chỉnh, cụ thể, dễ nộp

Dưới đây là một **ví dụ chi tiết** (tiếng Việt) cho Sprint Planning cơ bản dựa trên **ứng dụng quản lý công việc**. Bạn có thể dùng trực tiếp vào báo cáo hoặc nộp bài.

1) Thông tin Sprint (giả định)

- Thời lượng Sprint: **2 tuần (10 làm việc)**

- Team: **4 developer** (giả sử)
- Mục tiêu chung: hoàn thiện chức năng cốt lõi để người dùng quản lý task cơ bản.

2) Xác định Mục tiêu Sprint (Sprint Goal)

Sprint Goal:

"Cung cấp trải nghiệm quản lý công việc cơ bản cho người dùng: đăng nhập, tạo và xem danh sách task."

Lý do: tập trung vào luồng người dùng cơ bản (đăng nhập → tạo task → xem task) giúp ứng dụng có giá trị tối thiểu và có thể demo cho stakeholder.

3) Chọn User Stories từ Product Backlog

Chọn **ít nhất 2** User Stories liên quan trực tiếp tới Sprint Goal:

US-A — Đăng nhập (Login)

- *As a user, I want to log in so I can access my tasks.*
Vì sao chọn: xác thực là điều kiện bắt buộc để bảo vệ dữ liệu người dùng; nhiều tính năng khác phụ thuộc vào đăng nhập.

US-B — Tạo công việc (Create Task)

- *As a user, I want to create a task so I can track my work.*
Vì sao chọn: lõi của ứng dụng quản lý công việc; cho phép người dùng bắt đầu sử dụng app.

US-C (bổ sung, ưu tiên thấp hơn nhưng hữu ích) — Xem danh sách công việc (List Tasks)

- *As a user, I want to view my tasks so I can see what to do.*
Vì sao chọn: liên kết trực tiếp với tạo task; cần cho demo luồng end-to-end.

4) Phân chia mỗi User Story thành các task nhỏ (task-level breakdown)

US-A — Đăng nhập

1. Thiết kế UI màn hình login (wireframe + thiết kế đơn giản).
2. Viết frontend form (validation cơ bản).
3. Backend: API authentication (endpoint login, verify creds, trả token/JWT).
4. Lưu trữ session/token trên client (localStorage/cookie).
5. Xử lý lỗi & hiển thị thông báo (wrong password, user not found).
6. Unit test backend login + integration test flow (frontend→backend).
7. Kiểm thử thủ công + fix bug.

US-B — Tạo công việc

1. Thiết kế UI form tạo task (title, description, due date, priority).

2. Frontend: form + validation.
3. Backend: API tạo task (POST /tasks).
4. Database: migration/schema cho task (title, description, status, due_date, owner_id).
5. Hiển thị toast/notification sau khi tạo thành công.
6. Test API + test UI flow.
7. Kiểm thử thử công + fix bug.

US-C — Xem danh sách công việc

1. Thiết kế UI list view (list + simple filter/status).
2. Backend: API fetch tasks (GET /tasks?user=me).
3. Frontend: gọi API, render list, xử lý loading/error.
4. Thêm khả năng refresh/auto-update khi tạo task.
5. Test + kiểm thử.

5) Ước lượng công việc theo T-shirt Sizes (Small, Medium, Large) — kèm lý do

Giải thích mapping kích cỡ (ví dụ tham khảo):

- **Small (S)** \approx 1–4 giờ (đơn giản, ít rủi ro)
- **Medium (M)** \approx 1–2 ngày (8–16 giờ)
- **Large (L)** \approx 3+ ngày (24+ giờ, có sự phức tạp hoặc nhiều integration)

US-A — Đăng nhập \rightarrow Medium (M)

Lý do: gồm frontend + backend + security (token/session) + testing. Không quá phức tạp nhưng cần tích hợp (auth) và xử lý lỗi an toàn.

Phân tasks với kích cỡ:

- Thiết kế UI: S
- Frontend form + validation: M
- Backend API auth + JWT: M
- Lưu token client: S
- Xử lý lỗi & thông báo: S
- Test backend + tích hợp: M

US-B — Tạo công việc \rightarrow Small–Medium (S \rightarrow M) (chốt: M)

Lý do: logic CRUD đơn giản nhưng gồm frontend, backend, và DB migration; có validate nên cần thời gian testing.

Phân tasks:

- Thiết kế UI form: S
- Frontend form + validation: S
- Backend POST /tasks: S → M (tùy DB)
- DB migration/schema: S
- Notification: S
- Test: S

US-C — Xem danh sách công việc → Small (S)

Lý do: fetch + render; nếu chưa có pagination/complex filter thì khá đơn giản.

Phân tasks:

- Thiết kế UI list: S
- Backend GET /tasks: S
- Frontend render + loading: S
- Refresh after create: S
- Test: S

6) Ước lượng tổng hợp (ví dụ ước lượng giờ quy đổi từ T-shirt)

Áp dụng mapping để ước lượng thô:

- US-A (M) ≈ **12–16 giờ**
- US-B (M) ≈ **10–14 giờ**
- US-C (S) ≈ **4–6 giờ**

Tổng ước lượng: ~26–36 giờ.

Với team 4 dev và 2 tuần Sprint (10 ngày), đây là khối lượng nhẹ — khả năng hoàn thành cao. (Nếu team có velocity cũ, dùng velocity để điều chỉnh phạm vi.)

7) Phân bổ công việc gợi ý

- Dev 1: Backend auth (US-A backend) + API tasks (US-B backend)
- Dev 2: Frontend login + token handling (US-A frontend)
- Dev 3: Frontend create-task + UI list (US-B + US-C frontend)

- Dev 4: DB migration, test automation, bugfix, integrate frontend/backend

(Phân công tùy theo skill; đảm bảo pair testing hoặc review code.)

8) Definition of Done (DoD) cho mỗi User Story

- Code đã được review và merged.
- Unit tests và integration tests chạy pass.
- Manual acceptance test (kịch bản chính) pass.
- Tài liệu/notes deployment (nếu cần).
- Demo cho PO/Stakeholder.

9) Điều chỉnh rủi ro & dự phòng

- Nếu đăng nhập phức tạp (OAuth, SSO) → nâng mức to **L**.
- Nếu backend chưa có chuẩn API → thêm buffer 1–2 ngày.
- Nên reserve ~10–15% capacity cho bugfix, code review, meeting.

10) Kết luận ngắn gọn

- **Sprint Goal:** hoàn thiện luồng đăng nhập → tạo → xem task.
- **Chọn User Stories:** Đăng nhập (M), Tạo task (M), Xem danh sách (S).
- **Phân rã:** tasks rõ ràng (UI, frontend, backend, DB, test).
- **Ước lượng:** tổng ~26–36 giờ → phù hợp Sprint 2 tuần cho team 4 dev.
- **DoD** và phân bổ công việc đã chỉ rõ.

Bài 5:

1. Product Backlog (6 User Stories mẫu)

ID	User Story
US1	Là người dùng, tôi muốn đăng nhập để truy cập hệ thống.
US2	Là người dùng, tôi muốn tạo một task để ghi lại công việc.
US3	Tôi muốn chỉnh sửa thông tin task.
US4	Tôi muốn xóa một task khi không cần nữa.
US5	Tôi muốn đánh dấu một task là hoàn thành.
US6	Tôi muốn xem danh sách toàn bộ task của mình.

2. Ước lượng Story Points (dùng dãy Fibonacci: 1, 2, 3, 5, 8, 13, 21)

Sử dụng phương pháp Planning Poker: cả team giơ thẻ và thống nhất điểm.

User Story	Poker kết quả	Story Points chốt	Giải thích
US1 – Đăng nhập	5, 8, 5, 5 → chốt 5	5 SP	Có backend + validate + token
US2 – Tạo task	3, 5, 5, 3 → 5	5 SP	CRUD + UI + validate
US3 – Chỉnh sửa	3, 3, 5 → 3	3 SP	Tương tự tạo nhưng đơn giản hơn
US4 – Xóa task	2, 1, 2 → 2	2 SP	Một endpoint + confirm
US5 – Đánh dấu hoàn thành	3, 3, 2 → 3	3 SP	Update trạng thái
US6 – Danh sách task	8, 8, 13 → 8	8 SP	Fetch + filter + giao diện list

Tổng SP của toàn Backlog = $5 + 5 + 3 + 2 + 3 + 8 = 26$ SP

3. Velocity của nhóm (giả định dựa vào Sprint trước)

Sprint	Điểm hoàn thành
Sprint 1	14 SP
Sprint 2	18 SP
Sprint 3	16 SP

→ Velocity trung bình = $(14 + 18 + 16) / 3 = 16$ SP

Sprint mới tối đa chỉ nên chọn ~16 Story Points.

4. Xác định mục tiêu Sprint

Sprint Goal: “Hoàn thành các chức năng cốt lõi cho việc quản lý task: đăng nhập, tạo task và xem danh sách task.”

Lý do: đây là 3 tính năng tạo thành luồng sử dụng cơ bản của ứng dụng (login → tạo task → xem task).

5. Chọn các User Stories cho Sprint dựa trên Velocity = 16 SP

Chọn theo độ ưu tiên + liên quan Sprint Goal:

User Story	SP
US1 – Đăng nhập	5
US2 – Tạo task	5
US6 – Danh sách task	8

Tổng = 5 + 5 + 8 = 18 SP → vượt Velocity
→ Điều chỉnh: bỏ bớt hoặc giảm phạm vi.

Phương án chọn hợp lý nhất:

User Story	SP
US1 – Đăng nhập	5
US2 – Tạo task	5
US3 – Chỉnh sửa task	3
US5 – Đánh dấu hoàn thành	3

Tổng: 5 + 5 + 3 + 3 = 16 SP (khớp Velocity)

6. Phân rã User Stories thành Task + ước lượng thời gian

Giả định một ngày làm 6 giờ thực tế

US1 – Đăng nhập (5 SP)

Task	Ước lượng
Thiết kế UI login	3 giờ
Tạo form + validate frontend	4 giờ
API login + JWT/Token	6 giờ
Lưu token + redirect	2 giờ
Xử lý lỗi	2 giờ
Test & fix	3 giờ

Tổng: 20 giờ

US2 – Tạo Task (5 SP)

Task	Ước lượng
UI form tạo task	4 giờ
Validate form	2 giờ
API tạo task	5 giờ
Lưu vào DB	3 giờ

Task	Ước lượng
Test backend & frontend	3 giờ
Fix bug	2 giờ

Tổng: 19 giờ

US3 – Chính sửa Task (3 SP)

Task	Ước lượng
UI edit task	3 giờ
API update task	3 giờ
Test & fix	2 giờ

Tổng: 8 giờ

US5 – Đánh dấu hoàn thành (3 SP)

Task	Ước lượng
Update trạng thái hoàn thành	2 giờ
API update status	2 giờ
Test	1 giờ

Tổng: 5 giờ

7. Tổng thời gian và phân bổ vào Sprint

- **US1: 20h**
- **US2: 19h**
- **US3: 8h**
- **US5: 5h**

→ Tổng Sprint = 52 giờ

Bài 7:

User Story

“Là một người dùng, tôi muốn đăng ký tài khoản để có thể sử dụng ứng dụng.”

1. Chia User Story thành các Task nhỏ

Dựa trên quy trình phát triển phần mềm + hành vi người dùng + kỹ thuật frontend/backend, User Story này được chia thành các Task sau:

Task 1: Thiết kế UI Form đăng ký (Registration UI Design)

- Thiết kế layout gồm: Email, Mật khẩu, Xác nhận mật khẩu, Tên người dùng.
- Thiết kế các thông báo lỗi: email không hợp lệ, mật khẩu quá ngắn...

Task 2: Xây dựng Form giao diện đăng ký (Frontend Form Implementation)

- Tạo HTML/React/Vue form.
- Validate đơn giản: trường không được để trống.

Task 3: Thêm kiểm tra tính hợp lệ nâng cao (Form Validation nâng cao)

- Validate email đúng chuẩn.
- Validate mật khẩu đủ độ mạnh.
- Kiểm tra mật khẩu và confirm password trùng khớp.

Task 4: API đăng ký tài khoản (Backend Register API)

- Tạo endpoint POST /register.
- Lưu user vào database.
- Hash mật khẩu trước khi lưu (bcrypt).
- Kiểm tra email đã tồn tại chưa.

Task 5: Kết nối frontend với API backend

- Gửi form data từ frontend đến API.
- Nhận phản hồi success/error.
- Hiển thị thông báo tương ứng cho người dùng.

Task 6: Gửi email xác thực (Optional – tùy mức độ yêu cầu)

- Tạo API gửi email xác minh.
- Gửi mã kích hoạt sau khi đăng ký.

(Nếu không yêu cầu email verify thì có thể bỏ task này.)

Task 7: Test toàn bộ luồng đăng ký

- Test UI (manual).
- Test API (Postman).
- Test validation.

- Test luồng end-to-end (nhập → gửi → nhận phản hồi).

Task 8: Fix bugs & cải thiện UX

- Sửa lỗi validate.
- Cải thiện UI: loading, disable nút đăng ký...

2. Vì sao chọn các task này?

Task	Lý do
Thiết kế UI	Cần có trước để cả frontend + backend làm theo đúng yêu cầu.
Form frontend	Giao diện là điểm người dùng tương tác đầu tiên.
Validation	Tránh spam API và bảo vệ chất lượng dữ liệu.
Backend API	Là lõi xử lý dữ liệu đăng ký; bắt buộc phải có để lưu user.
Kết nối frontend – backend	Hoàn thiện luồng đăng ký, tạo giá trị cho người dùng.
Email verify	Trong nhiều ứng dụng thực tế là yêu cầu bảo mật quan trọng.
Testing	Đảm bảo User Story đạt Definition of Done.
Fix bugs + UX	Giúp luồng đăng ký mượt và đáng tin cậy.

Những task này được chọn vì:

- Chia theo các bước kỹ thuật (UI → Frontend → Backend → Integration).
- Chia theo hành vi người dùng (nhập thông tin → submit → nhận phản hồi).
- Mỗi task độc lập tương đối, có thể làm trong 1–2 ngày.
- Giúp nhóm dễ dự đoán tiến độ, review và test.

3. Ước lượng độ phức tạp Task (T-shirt Size)

Task	Ước lượng	Giải thích
Thiết kế UI đăng ký	S	Đơn giản, chỉ UI cơ bản
Xây form frontend	M	Có nhiều trường & sự kiện
Validate nâng cao	M	Regex, điều kiện mật khẩu
Backend API đăng ký	M–L	CRUD + hash + kiểm tra email
Kết nối frontend – backend	S	Chỉ gọi API và nhận phản hồi

Task	Ước lượng	Giải thích
Gửi email xác thực	M	Cần kết nối SMTP/Service khác
Testing	S-M	Test UI + API + end-to-end
Fix bug & cải thiện UX	S	Công việc nhỏ, ngắn

Nếu cần Story Points, có thể quy đổi (tham khảo):

- S → 1 SP
- M → 3 SP
- L → 5 SP

4. Tổng ước lượng (tham khảo Story Points)

Task	SP
UI đăng ký	1
Form frontend	3
Validate nâng cao	3
API đăng ký	5
Frontend-backend connect	1
Email verify	3
Testing	3
Fix bug	1

Tổng Story Points ≈ 20 SP

Bài 8:

Sprint 1

- User Story 1: 5 SP (Hoàn thành)
- User Story 2: 8 SP (Hoàn thành)
- User Story 3: 3 SP (Không hoàn thành → không tính)

Velocity Sprint 1 = 5 + 8 = 13 SP

Sprint 2

- User Story 4: 13 SP (Hoàn thành)
- User Story 5: 5 SP (Hoàn thành)
- User Story 6: 8 SP (Không hoàn thành → không tính)

Velocity Sprint 2 = 13 + 5 = 18 SP

2. Velocity trung bình của nhóm

$$\text{Velocity trung bình} = \frac{13 + 18}{2} = 15.5$$

Làm tròn xuống khi lập kế hoạch Sprint (theo thực tế Scrum):

Velocity trung bình = 15 SP

3. Lập kế hoạch cho Sprint tiếp theo

Giả sử Product Backlog có các User Story sau (tương tự cấu trúc backlog chuẩn):

User Story	Điểm Story
US7	8 SP
US8	5 SP
US9	3 SP
US10	2 SP
US11	13 SP
US12	5 SP

Nhiệm vụ: chọn các User Story có tổng điểm ≈ 15 SP (Velocity trung bình).

4. Lựa chọn User Stories cho Sprint tiếp theo

Dựa trên Velocity = 15 SP, một lựa chọn tối ưu là:

- US7 = 8 SP
- US8 = 5 SP
- US10 = 2 SP

Tổng:

$$8 + 5 + 2 = 15 \text{ SP}$$

Đủ 15 SP – phù hợp hoàn toàn với Velocity trung bình của nhóm.

5. Kết luận

Sprint	Velocity
Sprint 1	13 SP
Sprint 2	18 SP

Velocity trung bình 15 SP