## Bài 1:

**User Story – Tạo Công Việc Mới**

**ID:** US-01  
**Mức ưu tiên:** Cao  
**Mô tả:**

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

**Tiêu chí nghiệm thu (Acceptance Criteria)**

| **Mã** | **Tiêu chí** |
| --- | --- |
| AC01: | Người dùng nhập được thông tin công việc: tên, mô tả, ngày bắt đầu, ngày kết thúc, mức ưu tiên |
|  | Hệ thống phải kiểm tra tính hợp lệ của dữ liệu (validation) |
| AC03: | Khi lưu thành công, hiển thị thông báo và cập nhật danh sách công việc |
| AC04: | Giao diện thân thiện, dễ thao tác trên cả PC và Mobile |
| AC05: | Các trường quan trọng (như tên công việc) không được để trống |

**Ước lượng Story Points**

**8 điểm Story (theo dãy Fibonacci: 1 - 2 - 3 - 5 - 8 - 13 - 21 …)**

**Lý do lựa chọn 8 Story Points**

Tính năng "Tạo công việc mới" tưởng chừng đơn giản nhưng thực tế có nhiều yếu tố cần triển khai:

**Thiết kế UI/UX form nhập liệu**  
— Các trường đầu vào: tên công việc, mô tả, ngày bắt đầu/kết thúc, độ ưu tiên, trạng thái…

**Kiểm tra và xác thực dữ liệu**  
— Không để trống, định dạng ngày hợp lệ, giới hạn độ dài text,…

**Xử lý lưu trữ dữ liệu**  
— Ghi vào Database hoặc LocalStorage + xử lý lỗi

**Hiển thị thông báo và chuyển hướng**  
— Thông báo thành công/thất bại + cập nhật giao diện danh sách công việc

**Kiểm thử (Unit + Integration)**  
— Đảm bảo quy trình tạo mới hoạt động ổn định

**Xác định rủi ro và phụ thuộc**  
— Có thể liên quan đến hệ thống phân quyền hoặc đồng bộ dữ liệu giữa nhiều màn hình

**Đánh giá tổng quan**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Tiêu chí** | **Mức độ** | | --- | --- | | Độ phức tạp kỹ thuật | Trung bình | | Khối lượng công việc | Vừa đến hơi lớn | | Phụ thuộc | Có | | Rủi ro phát sinh | Trung bình | |

## Bài 2:

**User Story cần ước lượng**

***“Là một khách hàng, tôi muốn có thể thanh toán đơn hàng online để hoàn tất việc mua sản phẩm.”***

**Ước lượng Story Points**

**Ước lượng đề xuất: 8 Story Points** (theo thang Fibonacci: 1, 2, 3, 5, 8, 13…)

**Giải thích lý do chọn 8 Story Points**

Tính năng **Thanh toán đơn hàng** thuộc nhóm mức độ phức tạp trung bình – cao vì:

| **Hạng mục công việc** | **Độ phức tạp** |
| --- | --- |
| Thiết kế UI trang thanh toán (địa chỉ, phương thức thanh toán, phí ship…) | Trung bình |
| Xử lý logic giỏ hàng: tổng tiền, mã giảm giá, phí vận chuyển | Trung bình |
| Tích hợp cổng thanh toán (VD: PayPal, VNPay, MoMo…) | Khá cao |
| Xử lý trạng thái đơn hàng sau khi thanh toán | Trung bình |
| Kiểm thử bảo mật, lỗi thanh toán, xử lý hoàn tác | Khá cao |
| Giao tiếp backend và lưu dữ liệu | Trung bình |
|  |  |

**Độ không chắc chắn còn cao** vì liên quan đến nhiều bên thứ 3 (payment gateway), rủi ro lỗi giao dịch → tăng thêm điểm.

=> Không quá nhỏ như 5 points, nhưng cũng chưa đến mức rất lớn như 13 points.

| **User Story** | **Story Points** | **Mức độ** |
| --- | --- | --- |
| Thanh toán đơn hàng | **8 SP** | Trung bình – Khá phức tạp |

## Bài 3:

**Kịch bản Planning Poker — ước lượng cho User Story:**

**“Là người dùng, tôi muốn thêm sản phẩm vào giỏ hàng để có thể mua hàng trong một đơn hàng duy nhất.”**

**) Thành viên nhóm & lá bài họ chọn (chưa mở)**

* **Product Owner (Linh)** — chọn: *5*
* **Dev 1 (Minh)** — chọn: *8*
* **Dev 2 (An)** — chọn: *5*
* **QA (Hoa)** — chọn: *3*
* **DevOps / Backend (Tuấn)** — chọn: *8*

(tất cả chọn xong rồi — bây giờ **mở bài**)

**2) Kết quả khi mở bài**

* Linh: **5**
* Minh: **8**
* An: **5**
* Hoa: **3**
* Tuấn: **8**

Phổ: **3, 5, 5, 8, 8**

**3) Thảo luận (tóm tắt lý do cho mỗi lá bài)**

* **QA (Hoa) — 3**  
  Lý do: tính năng cơ bản — UI “Thêm vào giỏ”, hiển thị số lượng, tổng tiền. Nếu lưu local (localStorage) và không có nhiều xử lý, test case đơn giản → ít phức tạp.
* **PO (Linh) — 5**  
  Lý do: cần thêm yêu cầu nghiệp vụ (hiển thị mini-cart, cập nhật số lượng, validation, áp mã giảm giá cơ bản), cần UI/UX ổn, chấp nhận nhiều luồng → trung bình.
* **Dev (An) — 5**  
  Lý do: nếu backend đã có API chuẩn để thêm/xóa/đọc cart thì implementation nhanh; nếu chưa, vẫn có thể làm client-side trước. Có một vài edge-case (synchronization khi login, session) nên không quá nhỏ.
* **Dev (Minh) & DevOps (Tuấn) — 8**  
  Lý do: nếu yêu cầu là **đồng bộ hoá giỏ hàng giữa nhiều thiết bị**, cần endpoint, handle merge cart khi login, kiểm tra tồn kho realtime, rollback khi thanh toán thất bại — tăng đáng kể độ phức tạp. Cũng có rủi ro về concurrency và performance.

**4) Đi đến sự đồng thuận**

Sau thảo luận: các khác biệt chính là **(A)** liệu giỏ hàng có chỉ client-side hay phải sync với backend/merge khi login và **(B)** có cần kiểm tra tồn kho realtime hay không.

* Nếu **YÊU CẦU** là *chỉ client-side (localStorage) + UI cơ bản + cập nhật số lượng + hiển thị tổng*, nhóm đồng thuận: **3 — 5 SP**. Kết luận hợp lý: **5 Story Points** (chọn 5 để có margin cho các edge-case và testing).
* Nếu **YÊU CẦU** là *đồng bộ với backend, merge khi login, check tồn kho realtime*, nhóm đồng thuận: **8 Story Points**.

**Kết quả cuối cùng (với giả định yêu cầu tiêu chuẩn e-commerce trung bình — hỗ trợ sync khi user login nhưng không cần realtime stock):**

**Đồng thuận: 8 Story Points** (ghi chú: nếu chỉ local-only thì giảm xuống 5)

(giải thích: nhóm quyết định giữ an toàn ở **8** vì có rủi ro backend/merge và một vài edge-case; nếu PO xác nhận *không cần sync* thì sẽ điều chỉnh xuống 5.)

**5) Ghi lại kết quả ước lượng**

* **Initial cards:** 3, 5, 5, 8, 8
* **Discussion:** khác biệt do phạm vi (client-only vs sync backend) và rủi ro về merge/stock/concurrency
* **Final agreement:** **8 Story Points** (với điều kiện: bao gồm sync cơ bản giữa client và backend, merge cart khi login, xử lý quantity + validation + test)

**6) Nếu cần — phân rã thành task (gợi ý) & relative sizing**

(giúp sprint planning và assign)

1. Thiết kế UI/UX mini-cart & trang Cart — *1 SP*
2. Frontend: thêm/xóa/cập nhật quantity, hiển thị tổng tiền, lưu localStorage — *2 SP*
3. Backend: API thêm/xóa/đọc cart + merge khi login — *2 SP*
4. Business rules: coupon cơ bản, validation (max quantity) — *1 SP*
5. QA & E2E tests, unit tests, edge cases (merge conflict, rollback) — *2 SP*

Tổng (phân rã để tham khảo): ~8 SP

**7) Acceptance criteria (đề xuất — cần PO xác nhận)**

* Người dùng có thể thêm sản phẩm vào giỏ hàng từ trang sản phẩm.
* Hiển thị số lượng và tổng tiền chính xác.
* Có thể cập nhật số lượng và xoá sản phẩm.
* Khi user đăng nhập: cart local sẽ được merge với cart server (theo chiến lược PO xác định).
* Xử lý lỗi khi không đủ tồn kho (hiển thị message).
* Có unit test cho core logic và ít nhất 1 E2E test cho luồng thêm → checkout (nếu có).

**8) Ghi chú / Rủi ro cần làm rõ với PO (impact lên điểm)**

* Có yêu cầu **merge cart** khi login không? (nếu không: giảm SP)
* Có cần **kiểm tra tồn kho realtime** trước khi thêm hay chỉ khi checkout? (nếu realtime: tăng SP)
* Cần hỗ trợ **guest → user** cart migration không? (tăng SP)
* Có nhiều biến thể sản phẩm (size/color) và validation không? (tăng SP)

## Bài 4:

**User Story cần ước lượng**

**“Là người dùng, tôi muốn đăng nhập vào ứng dụng ngân hàng trực tuyến để truy cập tài khoản của mình.”**

Thành viên và lá bài đã chọn (ẩn)

| **Thành viên** | **Vai trò** | **Điểm chọn** |
| --- | --- | --- |
| Linh | Product Owner | 5 |
| Minh | Backend Dev | 8 |
| An | Frontend Dev | 5 |
| Hoa | QA Tester | 3 |
| Tuấn | Security / DevOps | 13 |

Khi mở bài → tập hợp điểm: **3 – 5 – 5 – 8 – 13**

**Thảo luận về lý do khác biệt**

**QA (3 SP)**

* Luồng đăng nhập đơn giản: nhập username/password → xác thực → vào hệ thống
* Test chủ yếu kiểm tra đúng-sai và thông báo lỗi

**Frontend & PO (5 SP)**

* UI đăng nhập + xử lý validation, handshake với backend
* Lưu token session, xử lý lỗi, lock UI khi chờ xác thực
* Tính năng trung bình, không quá rủi ro

**Backend (8 SP)**

* Cần xác thực an toàn, hash mật khẩu, token JWT/session
* Sai nhiều lần phải khóa tài khoản tạm thời
* Kiểm tra trạng thái tài khoản (bị khóa, nghi vấn…)

**Security/DevOps (13 SP)**

* Hệ thống **ngân hàng** → yêu cầu bảo mật rất cao
* Có thể bao gồm MFA (OTP), chống tấn công bruteforce, kiểm soát thiết bị đáng tin cậy
* Logging + monitor an ninh

**Đi đến đồng thuận**

Nhóm phân tích lại phạm vi:

| **Yêu cầu** | **Có / Không** |
| --- | --- |
| Đăng nhập cơ bản (username + password) | + |
| Mã hoá mật khẩu, token secure | + |
| Chống brute-force cơ bản (lock tài khoản khi sai nhiều lần) | + |
| OTP / 2FA | x (Sẽ có trong User Story khác) |
| Anti-bot nâng cao, device management | x |

Với phạm vi *đăng nhập ngân hàng mức cơ bản nhưng bảo mật tiêu chuẩn*, nhóm thống nhất:

**Kết quả ước lượng cuối cùng: 8 Story Points**

• Lý do: Tính bảo mật cao trong lĩnh vực ngân hàng → rủi ro lớn, cần kiểm thử kỹ lưỡng  
• Cao hơn tính năng web e-commerce bình thường (chỉ 3–5 SP)

Ghi chú kết quả Planning Poker

| **Thành viên** | **Lần đầu** | **Sau thảo luận** |
| --- | --- | --- |
| Linh | 5 | 8 |
| Minh | 8 | 8 |
| An | 5 | 8 |
| Hoa | 3 | 8 |
| Tuấn | 13 | 8 |
| **Kết quả cuối** | **8 Story Points** |  |

## Bài 5:

**BÁO CÁO SPRINT PLANNING + ESTIMATION**

**Mục tiêu**

Hiểu và áp dụng kỹ thuật **Story Points**, **Planning Poker**, và **Velocity** để lựa chọn User Stories và xây dựng kế hoạch Sprint hiệu quả cho ứng dụng **Quản lý công việc (Task Management App)**.

**Product Backlog (User Stories & Story Points)**

| **Mã US** | **User Story** | **Mô tả** | **Story Points (Planning Poker)** |
| --- | --- | --- | --- |
| US01 | Đăng ký tài khoản | Người dùng tạo tài khoản mới | 5 |
| US02 | Đăng nhập | Người dùng đăng nhập để truy cập task | 8 |
| US03 | Thêm task | Người dùng thêm công việc cần làm | 5 |
| US04 | Chỉnh sửa task | Người dùng cập nhật thông tin task | 5 |
| US05 | Xóa task | Người dùng xóa task không cần thiết | 3 |
| US06 | Đánh dấu hoàn thành | Người dùng hoàn thành task | 3 |
| US07 | Lọc và tìm kiếm task | Tìm kiếm theo trạng thái/ký tự | 8 |
| US08 | Đồng bộ dữ liệu backend | Lưu dữ liệu trên server | 13 |

**Velocity Sprint trước (giả định)**

Sprint trước team hoàn thành **18 Story Points**

Velocity = 18 SP → Dự báo năng lực hoàn thành của Sprint hiện tại cũng khoảng:  
**18 Story Points**

**Xác định mục tiêu Sprint**

**Hoàn thiện chức năng cốt lõi để người dùng có thể đăng nhập và quản lý công việc cơ bản**:

* Đăng nhập
* Thêm task
* Hiển thị & xử lý cơ bản với task (xóa + hoàn thành)

**Lựa chọn User Stories phù hợp với Velocity**

| **US** | **SP** | **Chọn?** |
| --- | --- | --- |
| US02 — Đăng nhập | 8 | ✔ |
| US03 — Thêm task | 5 | ✔ |
| US05 — Xóa task | 3 | ✔ |
| US06 — Đánh dấu hoàn thành | 3 | ✔ |
| **Tổng** | **19 SP** | **> Velocity (tạm chấp nhận)** |

Mặc dù 19 SP > 18 SP, team **tự tin** có thể hoàn thành nhờ mức rủi ro thấp → Phê duyệt Sprint Scope.

**Sprint Backlog phân rã thành Tasks**

**US02 – Đăng nhập (8 SP)**

| **Task** | **Estimate** |
| --- | --- |
| Thiết kế UI Login | 4 giờ |
| Tạo API login + mã hóa mật khẩu | 6 giờ |
| Lưu token session + xử lý lỗi | 4 giờ |
| Kiểm thử chức năng | 4 giờ |

**US03 – Thêm task (5 SP)**

| **Task** | **Estimate** |
| --- | --- |
| UI thêm task | 4 giờ |
| Lưu task vào local/API | 4 giờ |
| Kiểm thử | 2 giờ |

**US05 – Xóa task (3 SP)**

| **Task** | **Estimate** |
| --- | --- |
| Thêm nút xóa | 2 giờ |
| Cập nhật UI + xử lý xóa | 2 giờ |
| Kiểm thử | 2 giờ |

**US06 – Đánh dấu hoàn thành (3 SP)**

| **Task** | **Estimate** |
| --- | --- |
| Checkbox đánh dấu task done | 2 giờ |
| Cập nhật trạng thái hiển thị | 2 giờ |
| Test | 2 giờ |

**Phân công công việc cho thành viên**

| **Thành viên** | **Task đảm nhiệm** |
| --- | --- |
| Minh (Frontend) | UI Login, UI Task List |
| An (Backend) | API Login, Lưu/Xóa Task |
| Hoa (QA) | Viết test case và kiểm thử |
| Tuấn (DevOps) | Build, CI/CD và bảo mật cơ bản |

**Kết luận & Kế hoạch Sprint**

| **Thông tin** | **Giá trị** |
| --- | --- |
| Sprint Duration | 2 tuần |
| Velocity sử dụng | 18 SP |
| Tổng SP chọn | 19 SP |
| Sprint Goal | Đăng nhập + quản lý task cơ bản |
| Cam kết | Hoàn thành MVP chức năng |

## Bài 6:

**Sprint Planning — Ứng dụng Quản lý Công việc**

**1) Mục tiêu Sprint (Sprint Goal)**

**Hoàn thành tính năng quản lý công việc cốt lõi cho người dùng:**  
"Người dùng có thể thêm, xem và tìm/lọc công việc để quản lý danh sách việc cần làm một cách hiệu quả."

Mục tiêu rõ ràng, mang giá trị: cho phép người dùng tạo công việc mới và nhanh chóng tìm/lọc công việc đã tạo — điều thiết yếu để app có thể demo được tính năng quản lý task.

**2) Chọn User Stories từ Product Backlog**

Mình chọn **2 User Stories** liên quan trực tiếp đến Sprint Goal:

* **US03 — Thêm task**  
  *“Là người dùng, tôi muốn thêm công việc mới để ghi lại việc cần làm.”*
* **US07 — Lọc và tìm kiếm task**  
  *“Là người dùng, tôi muốn tìm kiếm và lọc các task theo trạng thái/chuỗi ký tự để nhanh chóng tìm task cần xử lý.”*

**Lý do chọn:**

* Thêm task là chức năng cốt lõi — nếu không có nó, không có dữ liệu để thao tác.
* Lọc & tìm kiếm nâng cao trải nghiệm quản lý khi số lượng task tăng; giúp Sprint Goal (quản lý công việc hiệu quả) được hiện thực hóa.  
  Hai story này kết hợp sẽ đem lại giá trị sử dụng ngay cho user: tạo + tìm/lọc.

**3) Phân chia mỗi User Story thành các tasks nhỏ**

**US03 — Thêm task (decompose)**

1. **Thiết kế UI form Thêm Task** (wireframe/UX)
2. **Frontend: xây dựng form, validation client-side**
3. **Backend/API: endpoint tạo task (POST /tasks)**
4. **Frontend: gọi API & xử lý response** (show success/error)
5. **Lưu tạm (offline/localStorage)** (nếu backend chưa có or offline mode)
6. **Unit tests (core logic)**
7. **QA / E2E test case** (thêm → hiển thị list)
8. **Documentation / Acceptance criteria sign-off**

**US07 — Lọc và tìm kiếm task (decompose)**

1. **Thiết kế UI cho search & filter (input + dropdown trạng thái)**
2. **Frontend: implement search debounce & filter UI**
3. **Backend: API query support (GET /tasks?search=…&status=…)** (nếu necessary)
4. **Frontend: tích hợp với API / local filter logic**
5. **Hiển thị kết quả & xử lý trạng thái “no results”**
6. **Unit tests cho filter & search logic**
7. **QA / E2E test case** (tìm kiếm → kết quả chính xác)
8. **Documentation / Acceptance criteria sign-off**

**4) Ước lượng theo T-shirt Sizes (Small / Medium / Large) & Lý do**

Quy ước:

* **S (Small)** ≈ đơn giản, under 1–2 developer-days, rủi ro thấp
* **M (Medium)** ≈ trung bình, ~2–4 developer-days, có vài edge-cases
* **L (Large)** ≈ phức tạp hoặc phụ thuộc nhiều (API, sync, merge), >4 dev-days

**US03 — Thêm task → Medium (M)**

**Lý do:**

* Có GUI + validation + gọi API + tests → nhiều bước nhưng mỗi bước rõ ràng.
* Nếu backend đã có API thì công việc chủ yếu frontend → giảm rủi ro; nếu backend chưa có cần thêm endpoint (tăng effort) nhưng vẫn trong giới hạn trung bình.
* Có một vài edge-case (duplicate, limit length, special chars) cần test.  
  → Đặt **M** để có buffer cho backend/frontend coordination và testing.

**US07 — Lọc và tìm kiếm task → Small (S) (hoặc Medium (M) nếu cần backend query)**

**Lý do:**

* Nếu filter/search xử lý client-side (dùng array filter + debounce): **S** — ít phức tạp, nhanh.
* Nếu phải hỗ trợ query server-side (pagination, large dataset), complexity tăng → sẽ là **M**.  
  **Kết luận:** mặc định giả định dataset vừa phải và hỗ trợ filter client-side → chọn **S**. Ghi chú: chuyển thành **M** nếu PO yêu cầu server-side search or fuzzy search.

**5) Acceptance Criteria (tóm tắt ngắn)**

**US03 — Thêm task**

* Người dùng nhập tiêu đề (bắt buộc) và mô tả (tuỳ chọn), nhấn "Thêm" → task xuất hiện trong list.
* Hiển thị thông báo khi thêm thành công / lỗi.
* Validation: không cho submit khi title rỗng; giới hạn ký tự (vd 200).
* Unit & E2E test cover core flow.

**US07 — Lọc và tìm kiếm**

* Người dùng nhập chuỗi tìm kiếm → danh sách lọc theo chuỗi (title/description).
* Chọn trạng thái (All / Open / Done) → danh sách cập nhật tương ứng.
* Hiển thị "No results" khi không có task phù hợp.
* Search có debounce (ví dụ 300ms) để tránh spam.

**6) Gợi ý phân bổ nhân lực & thời gian (ví dụ Sprint 2 tuần)**

* **Frontend dev (Minh)**: UI Thêm task + Search UI + tích hợp API — priority.
* **Backend dev (An)**: Nếu cần, triển khai endpoint tạo task & query API.
* **QA (Hoa)**: Viết test case E2E và thực hiện test.
* **PO (Linh)**: Xác nhận acceptance criteria, test UAT.

Ước lượng thô (dựa T-shirt → chuyển thành ngày):

* US03 (M): ~2–3 dev-days (frontend + backend coordination)
* US07 (S): ~0.5–1 dev-day (client-side)

**7) Ghi chú / Rủi ro cần làm rõ với PO (impact tới size)**

* Backend đã có API tạo task & lấy danh sách chưa? (nếu chưa → tăng effort)
* Cần xử lý offline / sync không? (tăng effort)
* Dataset lớn có cần server-side search/fuzzy/pagination? (S → M)
* Có yêu cầu audit/log hoặc trigger notify khi thêm task? (tăng effort)

## Bài 7:

User Story:

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

| **STT** | **Task cụ thể** | | **Mô tả nội dung công việc** |
| --- | --- | --- | --- |
| 1 | Thiết kế giao diện form đăng ký | | Tạo UI gồm: Họ tên, email, mật khẩu, xác nhận mật khẩu, nút đăng ký |
| 2 | Xây dựng chức năng nhập và kiểm tra hợp lệ dữ liệu ở Client | | Kiểm tra: định dạng email, độ dài mật khẩu, mật khẩu trùng khớp… |
| 3 | Tạo API Backend đăng ký tài khoản | | API nhận dữ liệu từ Client và xử lý đăng ký |
| 4 | Lưu dữ liệu người dùng vào database | | Tạo bảng User + Mã hóa mật khẩu khi lưu |
| 5 | Gửi email xác thực tài khoản (nếu yêu cầu) | | Tạo service gửi email kích hoạt tài khoản |
| 6 | Hiển thị thông báo kết quả đăng ký | | Thông báo thành công/thất bại, hướng dẫn đăng nhập |
| 7 | Kiểm thử toàn bộ chức năng đăng ký | | Unit test + kiểm thử luồng thực tế |
|  | |
|  |  | |  |

## **Lý do lựa chọn các task**

| **Lý do** | **Giải thích** |
| --- | --- |
| Chia theo quy trình phát triển phần mềm | Đi từ UI ➝ kiểm tra ➝ API ➝ lưu dữ liệu ➝ phản hồi ➝ test |
| Giảm độ phức tạp và dễ quản lý | Mỗi task có mục tiêu rõ ràng, hoàn thành được trong Sprint |
| Dễ chia cho nhiều dev trong nhóm | Frontend, Backend có thể làm song song |
| Giảm rủi ro thất bại Sprint | Theo dõi tiến độ tốt hơn, dễ điều chỉnh nếu gặp vấn đề |

## **Ước lượng độ phức tạp (Story Points – Fibonacci)**

| **Task** | **Story Points** | **Lý do** |
| --- | --- | --- |
| Thiết kế UI đăng ký | 2 | Giao diện đơn giản |
| Kiểm tra hợp lệ dữ liệu client | 3 | Một số rule hợp lệ |
| API Backend đăng ký | 5 | Xử lý logic, kết nối DB |
| Lưu dữ liệu vào DB (hash password, validate) | 5 | Yêu cầu bảo mật và ràng buộc |
| Gửi email xác thực | 3 | Tác vụ tích hợp dịch vụ ngoài |
| Hiển thị thông báo kết quả | 1 | Đơn giản |
| Kiểm thử toàn bộ chức năng | 3 | Test form, API, DB |

## Bài 8:

**1. Mục tiêu**

* Hiểu và áp dụng cách tính Velocity trong Scrum để dự báo khối lượng công việc trong Sprint tiếp theo, đồng thời áp dụng Velocity vào việc lập kế hoạch Sprint.

**2. Mô tả**

* Dựa trên thông tin và cách tính Velocity trong Scrum, bạn cần tính toán Velocity của nhóm từ dữ liệu User Stories đã hoàn thành trong Sprint trước và sử dụng nó để lập kế hoạch cho Sprint tiếp theo.

**Dữ liệu từ Sprint 1**:

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

**Dữ liệu từ Sprint 2**:

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

**Yêu cầu:**

1. **Tính Velocity của nhóm**:
   * Tính Velocity của nhóm từ Sprint 1 và Sprint 2.
   * Xác định Velocity trung bình của nhóm.
2. **Sử dụng Velocity để lập kế hoạch cho Sprint tiếp theo**:
   * Dựa trên Velocity trung bình từ các Sprint trước, ước tính số lượng User Stories có thể hoàn thành trong Sprint tiếp theo.
   * Chọn các User Stories có tổng số điểm Story phù hợp với Velocity trung bình của nhóm.

## Bài 9:

**Ước lượng công việc bằng Story Points**

**User Story:**

“Là người dùng, tôi muốn tạo công việc mới để ghi lại việc cần làm.”

**Các yếu tố ảnh hưởng độ phức tạp:**

* UI form nhập thông tin task (title, description, deadline, priority)
* Validation client-side (title bắt buộc, độ dài hợp lệ, deadline hợp lý)
* Backend/API lưu task vào database
* Kiểm thử unit test + E2E test
* Hiển thị thông báo thành công / thất bại

**Ước lượng Story Points:** **5 SP**  
**Lý do:**

* Mức độ trung bình: nhiều bước, có frontend + backend + validation
* Không quá phức tạp, không liên quan nhiều đến tích hợp bên thứ 3 hay concurrency
* Có một vài edge-case cần test (deadline hợp lệ, title trùng)

**Thực hiện Planning Poker**

**Các “lá bài” được giả định từ nhóm (dãy Fibonacci: 1,2,3,5,8,13)**

| **Thành viên** | **Chọn SP** |
| --- | --- |
| PO (Linh) | 5 |
| Frontend Dev (Minh) | 8 |
| Backend Dev (An) | 5 |
| QA (Hoa) | 3 |
| DevOps / Security (Tuấn) | 8 |

**Mở bài và thảo luận:**

* QA (3 SP): cho rằng UI + API không phức tạp, dễ test → SP thấp
* PO & Backend (5 SP): thêm backend logic, validation → SP trung bình
* Frontend & DevOps (8 SP): cần xử lý tích hợp, edge-case, deployment → SP cao

**Kết luận sau thảo luận:**

* Chọn **5 SP** đồng thuận (mức trung bình, an toàn)
* Giải thích: Story không quá lớn nhưng cần đủ buffer để xử lý backend + validation + test

**Lập kế hoạch Sprint dựa trên Velocity**

**Giả định Velocity nhóm (dựa trên Sprint trước):** 15 SP

**Product Backlog có các User Stories sau:**

| **User Story** | **Story Points** |
| --- | --- |
| Tạo công việc mới | 5 |
| Xem danh sách công việc | 8 |
| Chỉnh sửa công việc | 5 |
| Xóa công việc | 3 |
| Lọc & tìm kiếm công việc | 5 |

**Mục tiêu Sprint:**

Hoàn thành các tính năng cốt lõi để người dùng **tạo, xem và tìm kiếm công việc**, đảm bảo quản lý danh sách task cơ bản.

**Chọn các User Stories theo Velocity:**

* Tạo công việc mới = 5 SP
* Xem danh sách công việc = 8 SP
* Xóa công việc = 3 SP

**Tổng SP = 5 + 8 + 3 = 16 SP**

* Velocity trung bình nhóm = 15 SP → tổng SP 16 vừa đủ, có thể điều chỉnh trong Sprint nếu cần.

**Ghi chú phân chia tasks (ví dụ)**

**US: Tạo công việc mới (5 SP)**

* Thiết kế UI form task
* Validation client-side
* Backend/API lưu task
* Hiển thị thông báo kết quả
* Unit/E2E test

**US: Xem danh sách công việc (8 SP)**

* UI hiển thị danh sách task
* Backend/API fetch danh sách
* Pagination / Sorting / Filter cơ bản
* Kiểm thử E2E

**US: Xóa công việc (3 SP)**

* Thêm nút xóa
* Gọi API delete
* Update UI và hiển thị thông báo

**Kết luận**

* Sử dụng **Story Points + Planning Poker** giúp nhóm định lượng mức độ phức tạp thực tế của từng User Story.
* Dựa trên **Velocity trung bình (15 SP)**, nhóm có thể chọn các User Stories phù hợp cho Sprint tiếp theo.
* Mỗi User Story được chia thành **task nhỏ** → dễ theo dõi tiến độ, phân công, và đảm bảo hoàn thành trong Sprint.

## Bài 10:

**Chia nhỏ User Story thành các task**

**User Story:**

“Là người dùng, tôi muốn tạo giỏ hàng để thêm sản phẩm và thanh toán đơn hàng dễ dàng.”

**Task breakdown:**

| **STT** | **Task** | **Mô tả** |
| --- | --- | --- |
| 1 | Thiết kế UI giỏ hàng | Tạo giao diện hiển thị danh sách sản phẩm, tổng tiền, nút thêm/xóa sản phẩm, nút thanh toán |
| 2 | Xây dựng chức năng thêm sản phẩm vào giỏ | Frontend xử lý thao tác thêm sản phẩm, cập nhật số lượng, tổng tiền |
| 3 | Xây dựng chức năng xóa/giảm số lượng sản phẩm | Frontend xử lý xóa sản phẩm hoặc giảm số lượng |
| 4 | Xây dựng API lưu giỏ hàng | Backend xử lý thêm/xóa sản phẩm, lưu giỏ hàng cho user |
| 5 | Xử lý tính tổng tiền, thuế và phí vận chuyển | Tính toán tổng tiền tự động mỗi khi giỏ hàng thay đổi |
| 6 | Hiển thị thông báo và xử lý lỗi | Hiển thị thành công/ thất bại khi thao tác giỏ hàng |
| 7 | Kiểm thử chức năng giỏ hàng | Unit test + E2E test cho tất cả các thao tác |
| 8 | Tích hợp thanh toán (nếu scope có) | Kết nối cổng thanh toán và xử lý callback |

**Lý do chia task:**

* Theo **luồng phát triển**: UI → xử lý frontend → backend → business logic → test.
* Chia nhỏ để **mỗi task hoàn thành trong 1–2 ngày**, dễ theo dõi trong Sprint.
* Dễ **phân công nhân lực**: Frontend, Backend, QA làm song song.
* Giảm rủi ro task lớn không hoàn thành, dễ ước lượng và quản lý.

**Ước lượng độ phức tạp của từng task (Story Points)**

| **Task** | **Story Points** | **Lý do** |
| --- | --- | --- |
| Thiết kế UI giỏ hàng | 3 | UI nhiều phần tử nhưng không phức tạp |
| Thêm sản phẩm vào giỏ | 3 | Xử lý số lượng, cập nhật tổng tiền |
| Xóa/giảm số lượng sản phẩm | 2 | Chức năng đơn giản hơn thêm sản phẩm |
| API lưu giỏ hàng | 5 | Backend cần xử lý logic lưu giỏ hàng, session, database |
| Tính tổng tiền + thuế/phí | 3 | Logic tính toán đơn giản nhưng cần chính xác |
| Hiển thị thông báo/ xử lý lỗi | 2 | UI phản hồi, validation cơ bản |
| Kiểm thử chức năng | 3 | Test tất cả luồng cơ bản, validation, tính toán |
| Tích hợp thanh toán | 5 | Nếu có cổng thanh toán → logic phức tạp, cần API, callback |

**Lập kế hoạch Sprint**

**Giả sử Velocity nhóm:** 20 SP

**Chọn task phù hợp với Velocity:**

* Có thể hoàn thành:
  + Tất cả task từ 1 → 7 (21 SP → hơi vượt Velocity, có thể điều chỉnh hoặc chia nhỏ).
  + Hoặc task 1 → 6 (tổng SP = 16 SP) → an toàn cho Sprint.

**Ưu tiên task trong Sprint:**

1. Thiết kế UI giỏ hàng
2. Thêm sản phẩm vào giỏ
3. Xóa/giảm số lượng sản phẩm
4. API lưu giỏ hàng
5. Tính tổng tiền
6. Hiển thị thông báo/ xử lý lỗi
7. Kiểm thử chức năng (có thể triển khai song song với các task khác)

**Lý do lập kế hoạch:**

* Chọn các task cốt lõi để giỏ hàng có thể hoạt động end-to-end
* Dựa vào Velocity để tránh overcommit
* Giữ task tích hợp thanh toán cho Sprint sau nếu SP vượt quá Velocity

**Phân công nhân lực (gợi ý)**

| **Thành viên** | **Task đảm nhận** |
| --- | --- |
| Frontend dev | Task 1, 2, 3, 6 |
| Backend dev | Task 4, 5 |
| QA | Task 7 |
| Fullstack/PO hỗ trợ | Task 8 nếu có |