# Report

## Đánh số phần trăm progress

Ngay từ đầu dự án, nên có rule đánh % progress rõ ràng cho cả team, và **áp dụng cho tất cả công đoạn** (BD, DD, MK, CT, IT …)

**Ví dụ - Đối với task:**

1. Chưa start: *New* 0%
2. Started: *In Process* 5%
3. Sau khi hoàn thành:
   1. Nếu không có issue hoặc QA, và sẵn sàng cho khâu peer review, thì *Feedback* 80%
   2. Nếu còn nội dung TBD, issue, hoặc QA thì tùy vào khối lượng input/issue mà *Feedback* 60-80%
4. Sau khi hoàn thành peer review thì *Resolve* 90%
5. Sau khi khách hàng review
   1. Nếu OK thì finish 100%, và *Close* ticket.
   2. Nếu phải update và review lại thì vẫn để *Resolve* 90%

**Ví dụ - Đối với bug:**

1. Chưa start: *New* 0%
2. Started: *In Process* 10%
3. Tái hiện được bug: *In Process* 20%
4. Xác định được nguyên nhân: *In Process* 30%
5. Xác định được cách fix: *In Process* 50%
6. Fix xong cơ bản: *In Process* 70%
7. Fix xong hoàn chỉnh (chạy hết TC): *In Process* 80%
8. Re-test xong (sau khi merge branch bug vào branch chính): *Resolve* 90%
9. Khách báo ok và đóng bug: *Close* 100%

## Report template

* Phải đặt ra **template report** ngay từ đầu dự án. Ví dụ:
* Chỉ nên có **duy nhất một template report cho tất cả đối tượng**. Việc chia nhiều template report (vd: update task/bug trên Redmine thì theo template này, nhưng report cho Leader lại theo template khác, và report cho khách hàng thì lại theo template khác nữa) chỉ gây tốn effort và dễ nhầm lẫn thông tin.
* Một số template report được đúc kết trong quá trình làm dự án:
* Report theo ngày:

**1. Overal status**: [Tiến độ công việc chung: In process, delay, done]

**2. Today tasks**:

[Liệt kê các task làm trong ngày hôm nay và phần trăm progress của từng task. Ví dụ: ]

- Task #12345 (Close 100%): Tạo DD cho tính năng abc

- Task #56789 (In progress 20%): Code cho tính năng abc

**3. Next tasks**:

[Liệt kê các task sẽ làm trong ngày hôm sau. Ví dụ: ]

- Task #56789 (tiếp tục): Code cho tính năng abc

- Task #98765: Tạo test case cho tính năng abc

**4. Issues**:

[Liệt kê các issue chưa giải quyết được liên quan đến task đang làm. Cố gắng chi tiết]

**5. Risks**:

[Nếu issue mà ở mức nguy hiểm, gây ảnh hướng lớn, thì ghi vào đây]

**6. Others**:

[Các vấn đề khác, không liên quan đến các task được giao. Ví dụ: meeting, sự cố máy móc/điện mạng, …]

* Report theo task/issue/defect:

**Nếu là task:**

**I. Description**: Task này làm gì? (cụ thể hơn so với title của ticket)

**II. Overal status**: Mục Progress Percentage trên Redmine/Backlog…

**III. Dự kiến hoàn thành**: Muc duedate trên Redmine/Backlog…

**IV. Chia nhỏ task**: Chia task càng nhỏ càng tốt và đánh progress cho từng mục.

**V. Issue**: Đang gặp những issue/risk/limitation nào thì liệt kê hết ra. Mỗi issue có một action cụ thể.

**VII. Tài liệu tham khảo**: Liệt kê path một số tài liệu chính dùng để tham khảo khi làm task

**Nếu là defect:**

**I. Description**: Bug này là gì? (cụ thể hơn so với title của ticket)

**II. Overal status**: Mục Progress Percentage trên Redmine/Backlog…

**III. Dự kiến hoàn thành**: Mục duedate trên Redmine/Backlog…

**IV. Chức năng**: Bug này của chức năng gì? Ticket ID của chức năng đó là gì (hoặc dùng chức năng Related Ticket của Redmine)

**V. Root cause**:

**VI. Solution**:

**VII. Branch & commit ID**:

**VIII. Nguyên nhân gây bug**: Design sai? Test thiếu TC? Môi trường test sai?

**IX. Có cần check/test ảnh hưởng không**: Yes/No + lý do

**Other Templates**: *Project Management/Report Templates.xlsx*

**Notes:**

* Tìm ra nguyên nhân != phán đoán nguyên nhân 🡺 phải ghi rõ ràng trong report là "phán đoán bao nhiêu %". Nếu có nhiều hơn một phán đoán, liệt kê ra.
* Cách chính xác nhất để đánh giá tiến độ là có template % cụ thể (step nào tương ứng bao nhiêu %) + break down issue thành nhiều sub-issues và đánh % từng sub-issue.

## Quy định khung giờ report

Phải quy định rõ ràng **deadline để daily report** (vd: trước 5h chiều).

Đừng quên, nhiều member có **tâm lý delay việc report khi chưa có progress**. Nên leader phải có nhiệm vụ giúp member cảm thấy thoải mái khi report, để họ tự tin report đúng giờ (dù progress không có).

## Submit report ở đâu

Tùy dự án và tùy công ty, quy định nơi submit report và lưu trữ report cụ thể. Một số option:

1. Chat thẳng lên group team
2. Lưu trên wiki của Redmine

## Tips

* Dù task là investigation hay study, cũng **phải có output file** cụ thể, không thể report suông.

# Meeting

Meeting quá nhiều không bao giờ là tốt.

Tùy dự án và tùy giai đoạn, có **chiến lược meeting** sao cho hiệu quả nhất. VD, có nên daily meeting hay 2 lần/tuần? Có nên meeting tất cả member cùng lúc, hay chia nhỏ theo từng team nhỏ?

Ngoài ra, nên **tạo meeting request** (trên Skype chẳng hạn) để nhắc nhở mọi người chống quên.

# Quản lý tài liệu/code

## Tài liệu trên SVN

Ngay TỪ ĐẦU DỰ ÁN, **nên quy định folder tree** cho SVN.

Ví dụ:

*Folder structure for SVN*

|  |  |  |
| --- | --- | --- |
| **Folder** | **Path** | **Description** |
| Root folder | URL here | Root folder of the SVN for the project |
| ref | ./ref | This folder contains reference for source code and documents related to the project. |
| pm | ./pm | This folder contains project management information. It includes:   * estimate * meeting * plan * working\_rule |
| deliverables | ./deliverables | It is major folder for project working. It includes following major sub folders:   * trunk * branches * release |
| trunk | ./deliverables/trunk | This folder contains stable versions. |
| branch | ./deliverables/branches | Team member makes a copy of the trunk in the branches folder, and do your own work in a working copy of the branch you created.  For file naming for a new branch, please refer to Table 2‑5. |
| release | ./deliverables/release | This folder contains release versions. |

Khi team member commit bất kì tài liệu nào lên SVN thì phải commit đúng vào folder quy định sẵn.

**Ví dụ (tạo task và commit task):**

1. Member A được giao task tạo BD cho function nào đó (vd: task có Redmine ticket ID là #12345).
2. Member A bắt đầu task bằng việc tạo riêng folder cho task trên (vd: branches/Task\_12345\_MakeBDForABC) và commit tất cả những gì liên quan đến BD mà mình làm được lên folder này.
3. Khi member B review task của member A, member B cũng phải tạo review record file trong folder trên (sub-folder là "review").
4. Sau khi quá trình peer review hoàn thành và leader duyệt, member A copy folder của mình từ "branches" qua "release" (edit lại folder name, add thêm release note nếu cần, … sao cho phù hợp với quy định release).

**Note:**

* Tên file trên SVN không nên chưa dấu cách. Lý do:

NG: <https://csvn.cloud9-solutions.com/svn/s107.02/deliverables/branches/05_Tool/Back> Up CPRI\_MT Project VS 2019/CPRI\_MT\_Original\_Src/bin/L3\_Msg\_Info.dat

OK: <https://csvn.cloud9-solutions.com/svn/s107.02/mics/FromArtiza/20201102_アルチザ向け設計資産/CPRI-MT/03_source/bin/L3_Msg_Info.dat>

## Code trên Git

Ngay TỪ ĐẦU DỰ ÁN, nên **xác định rõ là sẽ dùng SVN hay Git để quản lý code**; đừng nhọc nhằn, dẫn đến rework sau này. Lời khuyên là nên dùng Git cho code.

**Quy định rõ ràng rule khi dùng Git**:

* Folder tree (rất quan trọng; nếu folder tree bị thay đổi lớn, sẽ gây khó khăn khi merge hay rebase)
* Ignore những file gì (commit file .gitignore ngay từ đầu)
* Ignore những changes gì (khác encoding, khác training spaces, …)
* Set EOL như thế nào (disable/enable tính năng [auto convert EOL](https://stackoverflow.com/a/39461324)?)
* Chia master branch và các feature branch (master chỉ chứa code sau khi fix hết bug, tất cả feature đều phải code trong branch riêng)
* Đặt tên branch (theo template gì). Ví dụ:
  + lte/develop
    - lte/feature/cpri-int-10
    - lte/fix/cpri-int-120
  + lte/release-ver1
  + nr/develop
    - nr/feature/cpri-int-125
    - nr/fix/cpri-int-625
  + nr/release-ver2
  + Đặt tên commit (theo template gì). Ví dụ: "[cpri-int-120] Fix …"
* Khi nào merge, khi nào rebase
* Commit code hằng ngày hoặc nhiều lần trong ngày (để tránh sự cố ngoài ý muốn)
* Không được delete commit (chỉ nên revert commit) và delete branch.
* Commit cả file attributes (read/write/execute permission)

**Tips:**

* Không phải member nào trong team cũng thành thạo Git, nên cần tổ chức 1 buổi **training về Git** ngày từ đầu dự án.

# Communication

Ngay TỪ ĐẦU DỰ ÁN, nên đặt ra các quy định liên quan đến communication cho toàn team.

## Chat

### Chọn phần mềm chat

**Skype** có lẽ là ứng dụng chat tốt nhất và phổ biến nhất cho công việc. Tuy nhiên, tùy vào đặc thù dự án, có thể cân nhắn thêm một số ứng dụng khác, như **MS Team** (hơi nặng và thừa thải tính năng, nhưng quản lý file tốt hơn Skype), **Telegram** (rất nhanh và nhẹ, nhưng chỉ nhắn tin cơ bản)

### Phân chia group chat

Mỗi dự án đều phải **tạo một group chat chung**. Ngoài ra, tùy vào số lượng member trong team và các phase của dự án, **có thể tạo thêm vài group nhỏ hơn**.

**Ví dụ:**

* Tách biệt group của BrSE và group của Dev
* Tách biệt group cho coding và testing

Tuy nhiên, đừng tạo quá nhiều group để tránh spam.

### Like hoặc Reply

Sau khi đọc notification, tất cả **member nên tương tác lại** (like hoặc reply) để leader biết được tình hình, xem có member nào bị miss notification hay không.

### Tránh spam

**Hạn chế enter liên tục** để tránh spam (notification liên tục).

## Email

Tiêu đề của mỗi email phải có **tiền tố** là tên dự án. Tiền tố này giúp đọc email nhanh hơn, cũng như để dễ dàng filter trên Outlooks.

**Ví dụ:**

|  |  |
| --- | --- |
| **Tiêu đề mail** | **Đánh giá** |
| [Artiza 5G] Customer feedback for No.18 | ĐẠT |
| Customer feedback for No.18 | KHÔNG ĐẠT |

**Lưu ý:**

* Chỉ dùng mail khi nội dung muốn gửi mang tính formal (làm evidence sau này). Trong đa số tình huống nội bộ trong team, chat vẫn tiện và nhanh hơn.

## Call

Đối với nhiều vấn đề, call là cách giải quyết tốt hơn nhiều so với chat. Do đó, đừng quên **gợi ý member call với nhau** khi cần thiết. Ví dụ: một member vừa điều tra một vấn đề nhức nhối mà các member khác cũng đang gặp phải; thay vì để các bạn chat với nhau, tại sao lại không khuyến khích call luôn cho tiện.

# Quản lý task

## Redmine

Redmine, nếu biết cách dùng, sẽ là **kênh tuyệt vời** để quản lý task và tiến độ task cho toàn team. Công cụ này cung cấp các tính năng: quản lý Task-Issues, Spent Time (thời gian làm mỗi task), Activity (recent-changed history about tasks), Calendar, Wiki, etc.

Một số kinh nghiệm dùng Redmine:

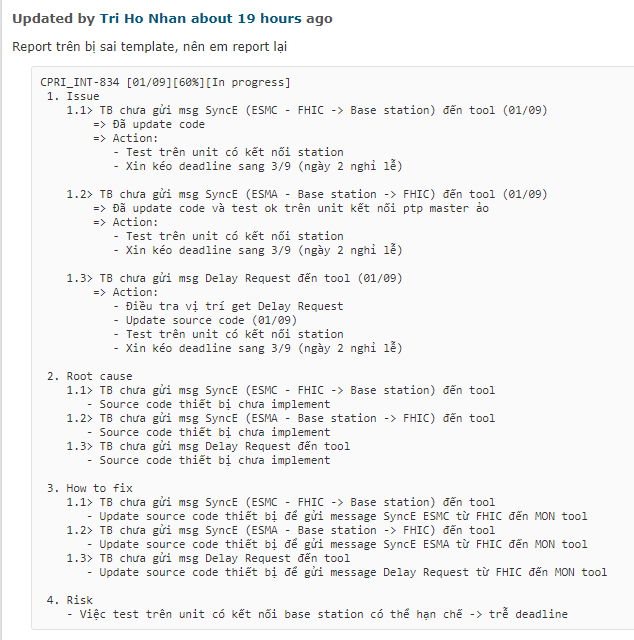
**Tips:**

* Luôn tạo **parent ticket** cho task. VD: Nếu parent ticket là fix bug, thì các child tickets (sub-tasks) là các bug cụ thể với ID cụ thể, và được assign cho các member cụ thể.
* Luôn đặt **tiền tố** vào tiêu đề của ticket. VD: [BD], [MK], …
* Việc **tạo ticket nên do đích thân leader làm**, và **assign cụ thể cho member**. Nếu để mỗi member tự tạo ticket theo ý của mình sẽ gây loạn, thiếu thống nhất.
* Yêu cầu member điền **đầy đủ thông tin cho mục Description** của mỗi ticket. Tốt hơn, quy định cụ thể **template cho từng loại task**. VD: BD cần các thông tin như document paths, limitations, …; MK cần cá thông tin như git branch, limitations, …; Fix bug cần các thông tin như root cause, solution, test result, …
* Quy ước đánh progress (tương tự mục [Report](#_Đánh_số_phần)).
* **Leader nên là người Close ticket**. Member chỉ nên được quyền **Resolve**.
* Đôi khi những thông tin quan trọng không đến từ specs hay design, mà đến từ những đoạn chat/mail của khách hàng. Do đó, **NHỚ lưu những thông tin này vào ticket luôn** để khỏi quên. BIẾN TICKET READMINE THÀNH PERSONAL NOTE!
* Tạo ticket theo task, chứ không theo ngày.
* **Sub task**: Mỗi task chỉ nên **kéo dài <= 3 ngày**. Nếu task quá lớn, hãy break down thì các sub-tasks.
* **Related task**: Các task có liên quan với nhau thì add chúng vào mục related task.

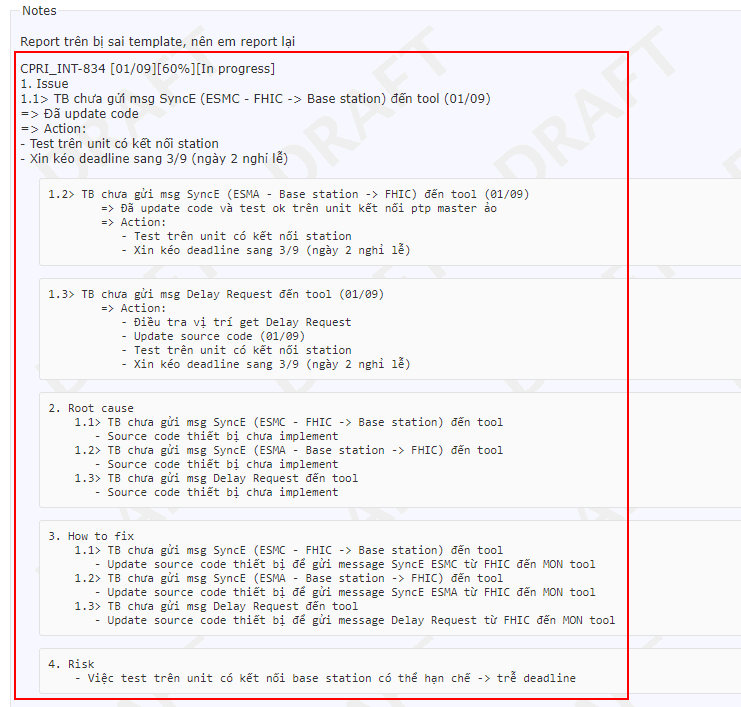


* **Meeting task**: Nếu tần suất meeting nhiều và sếp yêu cầu member phải luôn log đủ 8h/ngày, **tạo ticket riêng cho việc meeting** (chỉ cần 1 ticket là đủ) để log work.
* Mặc định, **comment trên Redmine không được quyền edit** (trừ khi member được cấp quyền edit comment). Nên dùng tính năng **Preview** để đảm bảo nội dung comment và format đúng mong muốn.
* Default text format trên Redmine không đẹp. Nên sử dụng **tính năng Preformated Text** để hiển thị đẹp. Ví dụ:

Khi dùng preformated text:



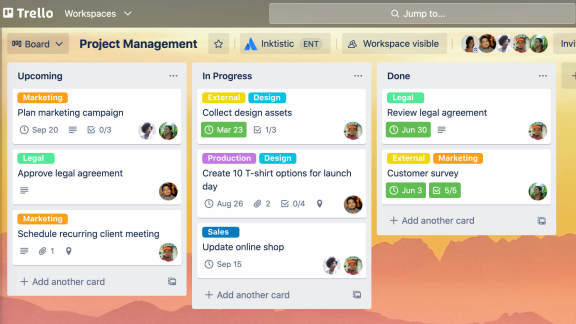
Khi dùng default format (có thể thấy text bị: không có dấu cách đầu dòng + hiểu nhầm format):



* Mục description và comment trên Redmine **rất bé**, khó làm việc. Nên tốt nhất, hãy **viết ra file text** với Notepad++ hoặc VSCode để đảm bảo nội dung chính xác. Rồi copy-paste vào Redmine.

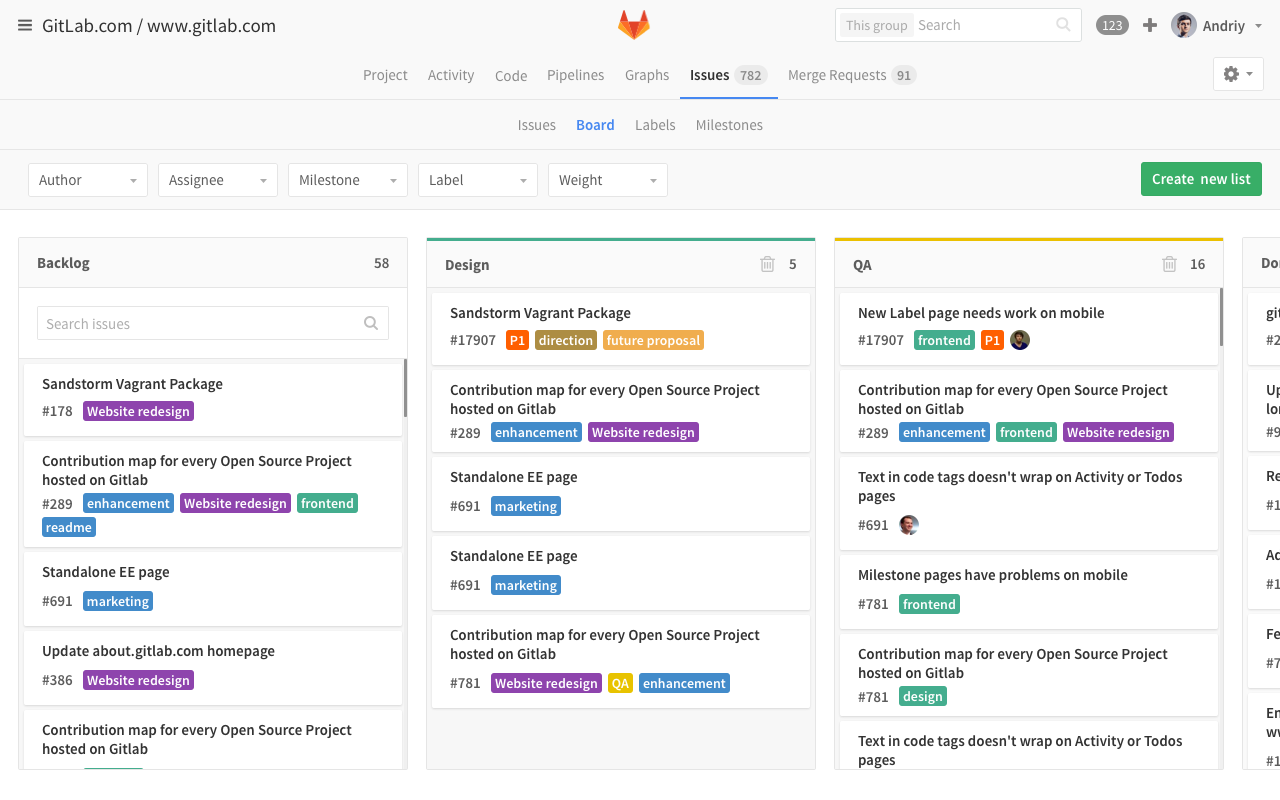
## Trello

Trello cung cấp tính năng Lists, Boards and Cards để quản lý progress của task một cách trực quan.



## Gitlab

GitLab chuyên dụng để quản lý source code (Repository, Code Review, Merge Request, Activity – recent-changed history about code, etc.). Nhưng công cụ này cũng hỗ trợ quản lý task rất hiệu quả với Tasks-Issues, Lists, Boards and Cards, Wiki, etc.



# Quản lý QA

QAs là một phần không thể thiếu trong mọi dự án. Do đó, ngay từ đầu, phải quy định **nơi lưu trữ QA** và **cách viết QA**.

**Làm rõ những điểm sau:**

* File nào chứa tất cả QA
* Template chung cho file QA:
  + Định dạng: Excel
    - Sheet 1: Chứa toàn bộ Q and A
    - Sheet 2+: Detailed info cho các Q cần trình bày thêm table/image …
* Template cho mỗi QA (number, feature category, content, asked date, who ask, answer, answered date, who answer, …)

**Tips:**

* Có thể chia QA thành 2 files. Một file internal (hỏi nội bộ team trước), một file external (nếu nội bộ team k trả lời được, thì hỏi khách hàng)

# Chia sẽ kiến thức

Làm task liên tục ngày qua ngày không phải là cách tốt nhất để giải quyết dự án. Nên **dành ra ít nhất một buổi mỗi tuần để leader và tất cả member ngồi lại, và chia sẽ những kiến thức hay** và mẹo vặt giúp công việc tốt hơn.

Ngoài ra, khuyên khích member **tạo guideline hoặc video**. Từ đó, dễ dàng share cho các member khác.