Time/Project management

To encourage project management, each group must maintain a living time management document through the scope of the project. This consists of three comma-seperated value files:

- `management/milestones.csv` : A list of defined milestones, representing units of work that are planned, plus dependencies and completion status.
- `management/schedule.csv` : A week-by-week view of planned effort (hours to be spent) going into the project.
- `management/effort-log.csv` : A log of actual effort spent, and which milestone(s) it is applied to.

This tracking system is very ad-hoc and simple, in order to minimise the amount of complexity and to avoid the need for new tools. However, it contains the key data and models the main activities of many commercial tools used in software development such as JIRA, Asana, Fogbugz, and so on.

If you have already used such a system (JIRA, Trello, etc.), and your team would prefer to use it, then I am happy with that, though let me know that you are using it. However, I would not reccommend trying to learn a new tool while doing the project.

So the key ideas are that you should:

- Refine a project into individual deliverables/milestones, with an estimated amount of effort (time needed).
- Look at the amount of time available to actually work on and deliver those milestones
- Track people's work, in terms of how much effort was expended, and on which milestone.
- Occasionally re-evaluate the plan:
  - Were estimates optimistic or pessimistic?
  - Is there enough time left in the schedule to complete the planned milestones?
  - Do the milestones still make logical sense?
- Refine the plan as necessary:
  - If planned effort is less than needed effort, then either reduce the scope, or increase the planned effort.
  - If completed milestones give more information about upcoming milestones, then refine the effort estimate.
  - If completed milestones give more information about the road-map, then refine later milestones into sub-milestones.

• Ensure to keep refining along the way

For this particular coursework, the expectation is not that you do project management well, simply that you actually do it.

File contents

Each file should work together, and is designed to be machine readable. The following describes the fields and meaning of each column in the files, and is intended to encourage interoperability. If the identifiers don't quite match up, then it isn't a disaster - the main goal is to make sure that people are trying to actively manage their project.

Milestones:

- **MilestoneId** : a short unique name for the label, matching the regex `[a-zA-Z_]+`
- **Dependencies** : an optional list of milestones on which this milestone depends. The list is seperated by semi-colons (why?), so this should match the regex `([a-zA-Z_]+(;[a-zA-Z_]+)*)?` .
- **EstimatedEffort** : Number of hours expected to complete this milestone. This can be an integer or a decimal, matching the regex `[1-9][0-9]*([.][0-9]+)` . Usually you wouldn't expect a milestone to be much less than 0.5 hours (too detailed), nor much more than 8 hours (too coarse).
- **Lead** : Who is going to drive this milestone forwards? This isn't necessarily who will do the work, but it is the person who is responsible for managing completion. This should be an Imperial login matching `[a-z][a-z]*[0-9]+` . always wvl16

Completed : Once the milestone is complete, this should contain the commit hash where it is completed. As a convenience, you can use the word `this` to indicate that the milestone is completed in the same commit as the word `this` appeared in `milestones.csv` . This should match `[0-9a-fA-F]{40,64}|this` .

Description : A human level description of what the milestone means. This should be a short sentence that communicates what the goals are. This will be embedded in a csv file, which means that you should avoid using commas in this description (comma escaping in csv is not uniformly supported across tools). So a suggested pattern is "[a-zA-Z 0-9;_-]+" ?

Schedule:

- **Week** : Week number being discussed (pre-filled).
- **DateRange** : Dates covered by this week (pre-filled).
- **PlannedEffort** : Effort that is planned to be expended across both group members. ? I'm guessing hours
- **ActualEffort** : Effort that actually was expended across both group members.
- **OtherDeadlines** : Other deadlines and events that are expected, and might have an impact on the amount of effort that can be expended in that week.

Effort Log:

- **Date** : Date that the effort was expended. This can either be a literal date or date-time expressed in ISO 8601 format, or as a convenience it can be `this` , which indicates that it is the same date/time as the commit where the entry was added.
- **Effort** : How much effort was expended in hours (matching `[1-9][0-9]*([.][0-9]+)`).
- **Developers** : Developers who contributed to this effort, matching `([a-z][a-z]*[0-9]+(;[a-z][a-z]*[0-9]+)?)?` . If this is empty it will be assumed that it is the same developers as for the previous log entry.
- **Milestone** : Which milestone(s) this was contributing to, matching `([a-zA-Z_]+(;[a-zA-Z_]+)*)?`

Summary

- Milestones are over-arching part
  - complete this to understand total project
- Schedule
  - time planned vs actual effort
  - factor in other deadlines
- Effort log
  - evidence of effort $\leftrightarrow$ milestone

Expectations

The expectation is that you will update the project plan on at least a weekly basis, so there is at least one commit to the project management information per week. Even if no work has been done, you should make a commit to indicate that nothing has been done (so put an entry in the effort log with 0 hours).

You do not need to update the information at a very high granularity, unless you wish to - updating the information should take only a minute, and does not need to be done for every commit. However, the log would be expected to line up with commits in some way - in many cases, there will simply be evidence of code commited or deleted. In other cases, time will be spent discussing things, planning, or designing data-structures on paper. Try to capture those things, e.g.:

- Add a note describing the decision taken in the effort log (e.g. "Decided to break X into milestones X1; X2; X3".
- Add a note or text-file describing the understanding gained (e.g. "Spend time learning bison, and discovered the following commands/patterns/etc.")
- Create a small text file that describes technical decisions taken, and commit it into a `notes` or `docs` directory.
- Include a photo of diagrams or sketches that were drawn (though don't use the ultra-high quality raw version - compress them a bit).

If you have spent two hours on something, but can't point to how that effort was spent, and identify the outcomes, you may not be using your time well.

Assesment of project management

The main goal of this component is to get you to do some time and project management, so you need to build up evidence that you did it. The assessment will then be in a 20 minute oral+feedback session per team in the Summer term, where the team spends 10 minutes explaining the milestones, and how things were managed. The only thing needed for the oral will be the submitted repository (no presentation or other preparation is needed). The remaining 10 minutes will be feedback on the submitted compiler deliverable.

Levels of achievement are:

- A : demonstrated good understanding of how to plan and track progress, and showed good practises in terms of managing milestones and time. Outcomes from effort spent are clear.
- B : updated progress each week, with acceptable granularity in terms of time spent and milestone planning. Most effort has a clear outcome.
- C : progress not updated regularly, and weak link between effort spent and achieved outcomes.
- D : sporadic time management, with files updated well after effort was expended.