MMP group meeting 1

JIM FINNIS

Dates

- Project outline: Mon, 8th Feb 2021
- Mid-project demo: between Thu, 11th Mar 2021 and Wed, 17th Mar 2021
- Hand-in for technical work and report: Fri, 30th Apr 2021.
- Final demos: Mon, 10th May 2021 and Fri, 28th May 2021 (but hopefully only the first two weeks of that time)

Process

- https://teaching.dcs.aber.ac.uk/docs/2021/MMP/manual/process/index.html
- This **ENTIRE DOCUMENT** should be your Bible!
  - https://teaching.dcs.aber.ac.uk/docs/2021/MMP/manual

Backup and version control

- Use both
  - Have a **repository** (probably with git)
  - This should contain **both code and documents**
  - Make sure it's **backed up** – typically you will have the local repository in the same place as your project, and a remote repo offsite to which you "push" regularly.

Time management

- Prioritising
  - What's **important** and what's **urgent.**
  - Do things that are **both,** first.
  - Don't do things that are **neither.**
- Set goals
  - Create milestones in your schedule: "by March 1st I want to have done…."
- Scheduling around the goals
  - Work out which bits need to be done first, guess how long they'll take
  - Which work depends on which other work
  - Allow time for things to go wrong!

Time management

- Obvious things
  - Don’t put off difficult work – do it earlier if you can
  - Don’t just **stop** if you get stuck – ask for help. I can’t do the work for you, but I can talk you through problems.
  - Don’t leave things until the last minute.
  - Keep writing as you go
- Online tools
  - I use kanbanflow.com , which is a simple Kanban board. You don’t need to be using Kanban to use this.
  - There are lots of others!

Writing up

- **Keep a blog!** This is something you all must do
- Not all MMP students have to do this, **but you do**
- It's not assessed – but you will find it **very very useful** when you write up
- It also helps me know how you're doing
- Every time you do some work, you should add a quick note:
  - What you are working on
  - How it's going
  - Anything you think you might want to read in the future!
  - Keep technical stuff (UML diagrams, specifications) elsewhere (to avoid cluttering it, and to avoid UAP!)
- https://users.aber.ac.uk/tof7/mmp/#blog
- https://ryangouldsmith.uk/blog/

Hannah Dee on the Blog

“I view it as part of the process of organisation, and it doesn't matter if they miss a week or if all that week contains is a couple of relevant links or youtube videos.

“The lack of progress thing is a separate issue - if they can't get something basic to work then I want to know that so I can help. Part of the project process (I think) is realising that some weeks you'll make progress on the software/research side of it, which is great. And some weeks you'll get stalled so should chip away at the background reading, or the writeup, or the testing, or ... and you can knock out 2 paragraphs of blog on any of these topics.”

Writing up

- LaTeX or Word?
- I prefer LaTeX myself –
  - Better reference management tools
  - Easier to manage a large document (my thesis was 400+ pages with over 250 references)
  - Better equations (although you may not need these!)
- But you're fine to use Word
- We provide templates: https://teaching.dcs.aber.ac.uk/docs/2021/MMP/manual/report-and-technical-work/report-templates.html

Writing up

- Sections
  - Some ideas here: https://teaching.dcs.aber.ac.uk/docs/2021/MMP/manual/report-and-technical-work/structure.html
  - I will want to see a **requirements specification** and a **design specification** for traditional projects, and detailed descriptions of each iteration for more Agile methodologies

Testing

- What is your **testing strategy?**
  - Full test-driven development with **unit tests** (JUnit for example)
    - Some like it, some don't.
    - Can be good for some parts of the code but not others (such as UI)
  - Traditional develop – then – test
    - Write out a **set of tests** your code must pass
    - There should be a **lot** of these, broken into sections for each part of the program
    - Report the results in a table: test number, test description, PASS/FAIL, notes.
    - Regularly do these tests, even if they already pass – you could have a **regression**, where new code breaks code that used to work.

UML tools

- Lucidchart (search templates for “UML”, there’s one for “UML class”)
  - Really nice, but only 3 diagrams allowed!
- Draw.io
  - Not quite as nice, but does UML class diagrams fairly well
- PlantUML
  - Takes a thing that looks a bit like a Java file and turns it into a diagram
  - Pain to get to work, but saves time once you’ve got it working
  - Only mentioning it because it’s what I use

Every week

- Brief progress report including….
- Show us something you've done
  - Prototype / "spike work" https://en.wikipedia.org/wiki/Spike_(software_development)
  - Piece of code
  - Piece of design
  - Note to the two music player students: careful here!
- Or tell us something you've found out
  - New tech
  - New library
  - New data structure
  - Something you're stuck with
- Everyone helps everyone else stay on track