CS3283/4

Media Tech Projects

Lecture 1

Overview

14 Jan 2017

About MTP

- Software engineering project with a media focus
- Team of 4-5 students
- Apply CS2103 to build a cool large-scale software

At the end of the project, you should be able to

- apply rigorous and principled software development methodologies and techniques to develop **user-friendly, robust, secure**, and **efficient** software systems

- demonstrate strong communication and teamwork skills
- apply CS principles to analyse and formulate a problem, and to design + implement + evaluate the solution to the problem.

Workload

- 8 MCs = 20 hours per week per person
- Produce 100 hours of work per week

Schedule

- Week 2 - 4: Analysis and Design
- Week 5 - 9: Implementation and Testing
- Week 10 - 13: Evaluation and Report

Assessment

CA	Week	Individual	Team
CA1	1 - 4	0%	20%
CA2	5 - 9	40%	0%
CA3	10 - 13	20%	20%
Overall		60%	40%

Assessment

- Quality and amount of work done
- For individual: weekly work log and code commits
- For team: requirement and design documents, final report, presentation, videos, etc.
- Peer reviews

Contact Time

- Weekly lecture (1st half)
- Team meeting (twice a week)
- Project session (10 hours a week)

Website

- https://nus-mtp.github.io/1617
- Follow the repo https://github.com/nus-mtp/1617 for changes

GitHub Repo

- **Wiki** for documentation / work log
- **Projects** for task management
- **Issues** for bug tracking
- **Pull request** for code review
- **Code commit** history for assessment

GitHub Repo

- **Wiki** for documentation / work log
- **Projects** for task management
- **Issues** for bug tracking
- **Pull request** for code review
- **Code commit** history for assessment

Schedule

- Week 2 - 4: Analysis and Design
- Week 5 - 9: Implementation and Testing
- Week 10 - 13: Evaluation and Report

Development Process

Software as a Service

- Continuous deployment
- Continuous integration
- Test-driven development
- Multiple sprints

Software Requirement

- **What** your software will do (not **how**)
- Finish a draft by end of your 2nd session **next week**
- Continue to iterate through the rest of the semester
- Week 4 version will be graded

Software Requirement

- Talk to your users / client
- Do a survey
- Study similar software
- Come up with functional / non-functional requirements
- Document them

Design / Architecture

- **How** to meet the requirements
- First draft by end of second session in Week 3
- Continue to iterate through the rest of the semester
- Week 4 version will be graded

Design / Architecture

- What framework/language/algorithm/ etc to use
- Database schemas
- Software components and their interaction
- UI design

Project Planning

- Organized into (at least) 5 weekly sprints
- Prioritize the features
- Estimate the time needed
- No plan is perfect: adjust as you go

Coding

- Write code
- Test code
- Review code
- Deploy
- Repeat

Tools

- Github for code review
- Linting tools for static analysis
- Travis for continuous integration
- Heroku for continuous deployment

Evaluation

- **Robust**: integration testing, load testing, etc
- **Secure**: penetration testing, etc
- **Efficient**: profiling and optimization
- **User-friendly**: Nielsen's heuristic, think-aloud protocol, etc.

Deliverables

- GitHub webpage
- Explainer video
- Developer’s guide
- STePS poster and presentation
- Oral examination and demonstration

Software Requirements

Functional
vs.
Non-functional

1.

Define the goals of the project

2.

Communicate among team members and with customers

3.

Leads to test cases and evaluation criteria

4.

Helps to plan project and schedule tasks

5.

Reduce bugs

7

Better software design

8.

Serves as contract

Properties of Good Software Requirements

1.

Correct

2.

Valuable

3.

Easy to Read

4.

Design Free

5.

Attainable

6.

Complete

7.

Consistent

8.

Unambiguous

9.

Verifiable

10.

Atomic

Use Case

vs.

User Stories

Use Case

- **System**: X
- **Actor**: Manager
- **Precondition**: Manager has logged in
- **Gurantees**: Module is added to student registration record
- **MSS**:
    - 1. Manager enters student number
    - 2. Manager enters the module code
    - 3. Manager clicks add
    - 4. System reports that module is added to student's record
- **Extensions**:
    - 4.1 Systems reports that module is full
        - 4.1.1 Manager clicks OK
    - 4.2. Systems reports that student does not have one of the prerequisite.
        - 4.2.1 Manager clicks "overwrite" to add the module to student's record anyway

User Stories

- **As** a manager, **I want to** register a module to a student **so that** the module appears in the student's record **if** the module is not full and student meets the prerequisite of the module (but the prerequisite can be overridden)

Next Week

- Decide the functional and non-functional requirements
- Write it down in GitHub

Tips

- Use the members' time effectively
- Use the meeting time effectively (remember the 10-10 split in workload)