Module Handbook

CS2103 Vs CS2103T

Same lectures, same exam. Separate tutorials, separate project grading. If not specifically mentioned, whatever is stated for CS2103 applies to CS2103T as well.

Lectures

Timing/venue: Fri 1400-1600 ICube Auditorium. Lectures start 1400 sharp and ends at 1535 (25 minutes to reach next lecture).

Attendance: Attendance for the first lecture is compulsory. Attendance for the remaining lectures is highly recommended. You can earn participation marks by attending lectures and handing in answers to the pop quizzes given during the lecture.

Webcast: All lectures will be webcast. However, some things are not captured well in the webcast recording. Please treat the webcast as a 'backup' for you to catch up anything missed during the lecture. Do not use it as an excuse to skip lectures.

Handouts: Each lecture comes with 1-4 handouts. These handouts will be uploaded to the IVLE by Wed midnight of the same week. You are welcome to, but not required to, read through them before attending the lecture. Some handouts also contain worked examples.
Read before tutorials: You are required to read the relevant handouts and give some thoughts to how to

apply those theories in your project before attending the corresponding tutorial. If you have any followup questions about handouts, please ask during tutorials (if there is spare time) or post in IVLE forums.

Slides: Since we are giving separate handouts, lecture slides will not be the traditional content-filled slides that you print and use as a reference for exams. Instead, handouts will serve as the main exam reference while slides will play only a supporting role during lecture delivery. Since you are going to ask for the slides anyway, selected slides will be released AFTER the lecture. Some slides cannot be released due to copyright reasons.

Topics: Please refer to the "Schedule" page to learn which lecture will cover which topics.

IDE workshop

This semester we have organized an additional workshop to help you get started with development tools. It will be held in Week 2. Eclipse/VisualStudio. Includes a bit of help on GUIs. Attendance eligible for participation marks. Please refer 'Schedule' page for more info.

Tutorials

Timing/venue: Starts on Week 3. Please refer the Schedule page for further details on each tutorial.

Choosing a tutorial: Note that all project team members should attend the same tutorial. That is because, unlike in previous semesters, most of the tutorials discussions are connected to the project. E.g. Your team will be regularly asked to explain to the tutorial group how you apply a particular theory in your own project and the other teams in the group to discuss your approach.
Furthermore, each tutorial slot is designated either as 'for C++' or 'for Java' (Refer the Schedule page to find which on is for which language). Teams doing their project in C++ are to attend one of the former while those using Java are to attend one of the latter. Those using C# can attend either type.

Participation marks: Tutorial attendance is counted for participation. Active participation in discussions (e.g. explaining to the class how your team did something), tutorial submissions (if a particular tutorial requires you to submit something) and tutorial preparation (if a particular tutorial requires you to prepare something beforehand) too will be counted for additional participation marks.
No participation marks for multitaskers: Doing unrelated things (e.g., social networking, IM etc.) while sitting in the tutorial will disqualify you from participation marks. More information about participation marks is explained later in this document.

Text book

There is no need to buy a text book this semester. Lecture handouts should be enough to give you the required coverage of our topics.

Programming language

Our learning materials (lecture slides, tutorials, etc.) use Java code examples. But you may use Java, C++, C# or pseudocode in your tutorial/exam answers. If you do not understand the code in the exam paper, please do not hesitate to ask the invigilator.
The project can be done using Java, C++ or C#. But note that we do not provide any learning resources or technical help for C# (supporting two languages is already hard enough). However, you are welcome to raise C# help requests in IVLE forums in case someone from our teaching team can help you. If you are choosing C#, try to form teams on your own and register early. It will be difficult to find enough C# teammates in your own tutorial if you wait till tutorial bidding is over. For more info on team formation, refer the 'Forming teams' sub topic later in this document.
We strongly recommend you to use Eclipse or VisualStudio for module related programming work. Since all team members will work on the same code base and project tutors too have to go through the code for

testing and evaluation, it is troublesome to use multiple IDEs.

Project

CS2103 project is structured to resemble the initial stages of a non-trivial real-life software project. In this project you will conceptualize a product and build a scaled-down version of the product, and have it ready to be continued by someone else.

Background story

In this project, we assume the following background story: You are teaming up with several of your university buddies to create a free desktop utility that you hope will attract many users. The product, if it becomes popular, will strengthen the SoC brand image and boost your own resume. Furthermore, future CS2103 students will continue developing your product to make it even better.

Note: Having a background story gives you a real-world context to base your decisions. For example, instead of asking us "should we have a GUI or not?" you will be asking yourself "if we are releasing this to the public as per our background story, does it make sense to release it without a GUI?" (see the FAQ below for more information about having a GUI).

The product

In this semester, you will develop a 'todo manager' tool especially targeting power users on Windows. i.e., *A todo manager for geeks on Windows*.

Problem outline: We are bombarded with 'things to do' continuously. The pile of todo items accumulate and weighs heavily on our mind at times. Sometimes things go out of control and we forget to do certain things on time.
Solution outline: It is good if every todo item is put into a systematic process that tracks them and helps us to decide what to do and when to do things so that we can keep our mind clear of todo items.
Target audience: There are MANY todo managers around. Calender software too can act as todo managers. When entering such a crowded product space, we should target a specific way of managing todo items by a specific type of users. Therefore, we target user like Jim whose 'todo management' workflow is described below.

Most of Jim's todo items arrive as emails. This is how Jim process his emails.
1. Decides what the follow up action required by that email.
    2a. If it can be done immediately, he does it right away (e.g., just reply to the email) and 'archive' the email (i.e. move it out of inbox).
    2b. If it cannot be done immediately, he schedules the follow up action in his calendar and archives the email. If he cannot decide a good time to do the action, he simply schedules it in a relatively free area in his calendar.
3. When Jim is free to do some work, he looks at his calendar and picks up the most appropriate thing to do. Once the task is done, he deletes the item or mark it as 'done'. If there is a further follow up action required, he schedules it in the calendar.
4. Jim periodically reviews the calendar to pick items that could not be completed and need to be rescheduled or discarded as 'cannot do'.
5. Todo items not arising from email are dealt similarly by entering them in the calendar.

As you can see from the above workflow, Jim's inbox is almost empty most of time. He no longer worries about an inbox full of emails he has to deal with at any time he login to email. He also does not have to keep any todo items in his mind because everything is recorded in the calendar somewhere. However, here are some 'pain points' of his workflow.

* Jim is currently using Google calendar. Since he store/retrieve/edit todo tasks so frequently, he find the online calendar too slow to work with. He'd rather have a desktop software he can summon quickly. Even better (but not a must) if he can activate the software by pressing a keyboard shortcut. He is quite impressed by how Google Desktop Quick Search feature can be activated by pressing

the control key twice quickly.
* Entering an event in a calendar is a pain because it takes several clicks. Jim would rather type than use the mouse because he can type faster than fiddling with the mouse. For example, to be able to type in commands such as "add July 10, 5-6, project meeting" (such as allowed by GCal's 'quick add' feature). He is not against GUIs, but feels most tasks can be done faster using text commands if the software is designed well. However, GUI can still play a part as a way to give visual feedback or for doing tasks too complex to do using text commands.
* A calendar is not very good in capturing todo items that need to be done before a specific time, or after a specific date.
* While it is good that all Jim's todo items are currently online and available to access from any computer, he almost always uses his own laptop to access them and sometimes from places where there is no ready Internet connectivity.
* Jim finds it is painful to do certain things important to Jim's workflow but not so important in a regular calendar. e.g. looking for suitable slot to schedule an item, searching for specific todo items, marking an item as done, deciding what todo item to do next, postponing an item.

In this project, you will try to solve some of Jim's problems to make his workflow even more efficient and painless. i.e. to make his life better. You don't have to solve all problems mentioned above and there may be other problems you can solve but not mentioned above. Good software delight users by solving problems users didn't even realize they had. You can also draw inspiration from the Todo.txt project, but try to do better. A GUI is optional. The data should be stored in a human readable text file. Integration with GCal (or other Calendar software or ToDo manager software) is optional too.
You are welcome to solve this problem by using a different workflow, if that workflow is better than the one suggested.

You should also comply with the following constraints. The purpose of these constraints is to increase comparability among submissions and to maximize applicability of module content in the project.

- **Constraint-Worthwhile**: The software should be something worth building. That is, it should have some value to someone. However, it need not be commercially viable.
- **Constraint-Desktop**: The software should work on a stand-alone desktop without network/Internet connection. It should not be a mobile app or a cloud-based application. If it is a client-server software, it should work when both the client and the server is run in the same desktop. However, you are allowed to have mobile/web-based extensions as long as the core functionality can be used on a standalone desktop.
- **Constraint-Standalone**: The software should work stand-alone. It should not be simply a plug-in to another software. However, you can build optional extensions that integrates your application with other existing software. Furthermore, you are allowed to build extensions that can plug into your software.
- **Constraint-No-Database**: The software should not use relational databases. (reason: Using relational databases reduces the scope for applying OO techniques). Data storage must be done using files you create yourself or using object serialization.
- **Constraint-OO**: A significant part of the software should follow the Object-oriented paradigm. However, some parts of the application can be non-OO.
- **Constraint-Windows**: The software should work on the Windows platform.
- **Constraint-No-Installer**: The software should work without requiring an installer.
- **Constraint-External-Software**: The use of a third-party framework/library may be allowed only if
  *it is free,
  *does not require any installation by the user of your software,
  *does not violate other constraints,
  and subject to prior approval by your supervisor. We will not allow third-party software that can interfere with the learning objectives of the module. Please email your project-tutor for approval before you start using the library.
- **Constraint-Opensource**: The source code and all other submitted documents may be made

- **Constraint-Novel**: In general, choosing to build something that is very similar to an existing software is allowed but discouraged. Instead, consider building an interesting variation of an existing software if you cannot think or a completely novel application.

Note that developing a bigger product (more functions, more code) does not necessarily earn you more marks. Often, well-executed projects producing small yet elegant products can score more than those who struggle through the project and produce larger yet half-baked products. Working hard is good, working smart is better.

Given below are some sample products produced during a previous semesters.

- Sample set 1 (AY09/10-Sem I) : Note that in that semester, all students were required to produce a variation of Sudoku and above constraints were not applied. Weightage for the project in that semester was 30%.
- Sample set 2 (AY09/10-Special Sem): In that semester, students were allowed to produce any product (because of the small class size). However, they had only 5 weeks to produce it and the weightage was 35%.
- Sample set 3 (AY10/11-Sem 1): In that semester, some students developed a product similar to the current project. However, the weightage was only 35%.

Note that this semester's project weightage is much higher (50%) and the level of help given to you is much higher. Naturally, we expect better (not necessarily bigger) products this time.

Note that C++ projects will be graded separately from Java and C# projects. This is to account for the perceived difference in the learning curve in C++ technologies compared to Java and C#.
CS2103T projects will be assessed separately from CS2103 projects. This is to account for the difference in workload.

Project deliverables

Here is the outline of the project structure and the deliverables at various milestone of your project. The subsections further down explain each deliverable in detail.

- First, a coding exercise to freshen up your coding skills and prepare you for the project.
    - CE1: This is an individual exercise. You can start this before you team up.
- Next, you will specify what you propose to build, resulting in the following deliverable.
    - Project proposal
- Iteration 1 comes next, in which you implement a working version (named V0.1) of the product that contain a selected subset of features. Deliverables at the end of this iteration are
    - Product V0.1
    - Project manual V0.1
    - Product demo (live)
- At this point, you will do another coding exercise to affirm the skills you honed during iteration 1. It also gives you a taste of what it is like to work in a big project with many developers and many users.
    - CE2: This too is an individual exercise.
- During iteration 2, you will improve the functionality delivered in the previous iteration and add some more functionality. Deliverables at the end of this iteration are
    - Product V0.2 (executable, source code, blurb, screen shot)
    - User guide for V0.2, Developer guide for V0.2
    - Product demo (live + video)

Please see the "Schedule" page to find out submission deadlines. Given below are the detailed description of each submission.
In addition to the above 'hard' deliverables, there is a 'soft' deliverable called project participation (worth 5%). This is a measure of how well you are prepared for, and participate in, discussions during tutorials. This means your project team should work as a team when preparing for tutorials and during tutorial discussions. We expect all team members to take part in discussions, not just one or two members dominating discussions.

CE1: TextBuddy

Write a CLI (Command Line Interface) program called TextBuddy using Java/C++/C# to manipulate text in a file. Here is a sample output for the program. Text in purple are commands entered by the user.

```
c:> TextBuddy mytextfile.txt
Welcome to TextBuddy. mytextfile.txt is ready for use
command: add little brown fox
added to mytextfile.txt: “little brown fox”
command: display
1. little brown fox
command: add jumped over the moon
added to mytextfile.txt: “jumped over the moon”
command: display
1. little brown fox
2. jumped over the moon
command: delete 2
deleted from mytextfile.txt: “jumped over the moon”
command: display
1. little brown fox
command: clear
all content deleted from mytextfile.txt
command: display
mytextfile.txt is empty
command: exit
c:>
```

You may make any reasonable assumptions about program behavior not evident from the above sample. State all your assumptions in the header comment of the TextBuddy main file.
Things that might earn you a Kudos (What’s a Kudos? See ‘K/K mode Grading’ in this document):

- Implement other commands such as ‘sort’, ‘search’ etc.
- In addition to the text UI described above, implement a GUI that has a text box to enter the command and another area to display the response.

submission:

1. Zip up the entire project folder, name it "["TutorialGroupID"][“Your full name”][CE1]".zip (e.g. [M10][Li Jien][CE1].zip) and upload to the appropriate IVLE folder. No *.rar files please.
   Also include testinput.txt, expected.txt files used for testing (put these at the same folder where TextBuddy.class or TextBuddy.exe is.
2. Copy paste your code and input.txt to a single document (use this template), format it for human reading (hint: make font size small until text no longer overflow to the next line. Small font size is ok, we can zoom-in to read), and submit as a single pdf to the appropriate IVLE folder. File name is similar to the zip file but replace ‘.zip’ with ‘.pdf’.

   Please follow the naming convention precisely as we subject your submissions to some automated processing.
   If re-uploading an updated version, please delete the previous version first. IVLE does not overwrite the previous version.

evaluation: K/K basis. Code quality will be considered in grading, in addition to the functionality. Resist the temptation to plagiarize. All submissions will be subjected to automated plagiarism detection and also will be published to the class.

CE2: OSS exercise

This is a confidence building exercise scheduled around half-way into the semester. After you have some experience with the basic tools of the trade, the project will encourage you to tinker around in a big project with a large code based written by a large group of developers.
Specifically, you will work with code from a popular open source project. You will download code from the project and fix a bug documented in a given bug report. More details will be given near to the deadline.

Project proposal

This is a description of the product you plan to build. Use this file as a template. Things to include:

- **Cover page:** Contains Team details and product name.
- **Product vision**: Describe what the product would be like when it is released as V0.2. This should

be from the user's point of view and can take the form of a 'quick start guide' for users. Use UI prototypes (can be simple sketches drawn on whiteboard/paper and scanned/photographed or created using a tool such as Powerpoint, HTML, or Balasmiq). The important thing is to help the reader visualize what the product will look like and convince them it is a cool and useful product. State what you honestly believe you can do, based on information you have at this point. A bit of guessing is inevitable.You will not be penalized if you cannot deliver what you promised. It is perfectly OK to change the project scope along the way.

- **Development plan**: This section is for a technical audience. You may include details such as which features will be delivered in which version, other major milestones, rough sketch of an architecture and other technical details that you would like our feedback on. This section is not evaluated. It is simply a chance for you to get early feedback.

suggested length: 3-4 pages excluding cover page.
submission: Name the file as "["team number"][Proposal]".pdf (e.g. [C13][Proposal].pdf) and upload to the appropriate IVLE folder. No doc or docx files please. Note that there are no spaces anywhere in the file name. Please follow the naming convention precisely as we subject your submissions to some automated processing. Only one member need to upload the proposal to IVLE.
As always, you are allowed to contact project tutor early and get feedback in advance. There is no need to wait till the proposal deadline.
evaluation: K/K basis.

-- [Start of V0.1 deliverables] ---

Project manual V0.1

Your project manual is to contain following sections:

- **Cover page**: As before. Use the same template. Change '[Proposal]' to '[V0.1]' in the page header.
- **Credits:** Acknowledgements of any third party materials used (if any). Omit this section if you did not use any third party materials.
- **User guide**: This part is for potential users to learn how to use your product. If your product is very intuitive to use (try to make it so), this section does not have to be very detailed or lengthy. Suggested length: 3 pages
- **Developer guide**: You will write this section as if it will be given to another team to develop the product. That is, the intended audience for this document is another team that will depend on this document to tell how the product has been designed and implemented. The document contains just enough information required for them to take over the project. The basic questions you should ask yourself when writing this document are:
  - What do we know about the system at this point?
  - Which part of that knowledge will be useful for the next team who will be taking over?
  - What is the best way to pass that knowledge to them? Note that some information can be passed on in the form of code without any additional documentation.

  Given below are some things you could consider including. Note that you are not required to include them all. Neither you are restricted to what is listed below.
  - Architecture
  - important APIs
  - Design descriptions (class diagrams, sequence diagrams, notable algorithms ...)
  - Code examples
  - Instructions for testing

  Suggested length: 5 pages excluding cover page.

See "Project Teams" page if you are not sure of your team number or the project tutor.

submission: Submit a single pdf file that contain all sections described above. Name the file as "["team number"][V0.1]".pdf (e.g. [W13J1][V0.1].pdf) and upload to the appropriate IVLE folder.

evaluation: K/K basis.

Product V0.1

This is a working system containing the functionality you selected to deliver at this milestone.
submission: A zip file containing the Visual Studio or Eclipse project from which you can build and run the program. Name the file "["your team number"][V0.1]".zip (e.g., [F10S3][V0.1].zip), and upload to the "[V0.1] Product" IVLE folder. No *.rar files please.
evaluation: Based on a demo and feedback session. Grading is on K/K basis.

V0.1 demo and feedback session

The main objective of this session is to give you feedback on your progress and future directions. In addition, we use it to evaluate your V0.1 submission. The whole team need to be present at the demo. At the same time, we will also discuss your manual and give feedback.
It is an informal session. There is no dress code. We do not require a formal presentation from you, but if you want, you may bring some slides prepared to explain things. Bring your own laptop for the demo. To save precious time, please be ready with the laptop booted up and the executable and the manual ready to open. Since most sessions are scheduled back-to-back, be there ahead of time.

--- [Start of V0.2 deliverables] ---

Developer guide V0.2

This is an *improved and updated* version of the 'developer guide' section of V0.1 project manual. It now describes the system at this milestone. Imagine you are handing over the project to another team at this point. This new team has never seen the previous version of the document and will rely totally on this version to understand the project. You have flexibility over what to include in this version of the document. However, given the purpose of the document, it is appropriate to repeat a refined version of most of the stuff you put in the previous version of the document.
suggested length: 10
submission: Similar to the previous version of the document. Update the file name and the page header appropriately.

Product V0.2

This is a working system containing the functionality you are able to deliver at this milestone. Note that what you deliver may differ from what you initially promised in the proposal, as long as you have a good reason for it. It may contain an improved version of the functionality delivered before as well as some new functionality.
submission:

- A zip file containing source files. Format and submission similar to V0.1. Use 'V0.2' instead of 'V0.1' when naming the file.
  Note that each source files should start with a header comment that describes the purpose of the file and the main authors' name. In case a file was written by multiple authors, you may mention up to two authors who contributed most to the file. The main author is the person who contributed most to that file. If you are using Visual Studio to develop, please include the whole project folder.
- A demo video that showcases the current functionality of the product. Remember to bring out how users can use the product to solve their problems rather than simply describing features of the product. E.g. describe a scenario of a user using the product, starting from the very first use (what to do first), and going through typical scenarios of usage.
  Create the video as if it is targeted for potential users (not teaching team). Please do not mention module name/code, school name, lecturer name etc. in the video (there is no need to advertise the product as done in a school project).
  Name the file using your team number. e.g., [T11C2][V0.1].mpg. You can use free tools such CamStudio, Debut or Jing for screen-recording the demo. You may use any video format that is supported by YouTube (e.g., mp4, wmv, avi,...). Just a screen recording is enough. Text annotations and voice narration are not necessary (but OK to have). A brief guide for creating video product demos is here.
  Upload to the "[V0.2] Video" IVLE folder. Note the 50Mb upload limit imposed by IVLE.

Additional files required for publishing your product (an example can be seen in [this page]):

- An executable version (an executable jar file or an exe file) that we can simply double-click to run your product. If you did not manage to create a single-file executable, at least make it such that 'unzip and double-click to run'. However, do not include source files. Name it "["your team number"][V0.2]".exe/jar/zip e.g., [T11C2][V0.2].exe and upload to the "[V0.2] Publish" IVLE folder.

- User guide meant for users of the product. This can contain two sections:
  (a) Quick Start Guide: Written in 'step-by-step tutorial style' explaining how to go about from the point of downloading to making productive use of the application in the most basic and typical fashion.
  (b) User Manual: Written in reference style to explain all nitty gritty details of all features, including advanced usage.
  If applicable, mention 'known issues' too.
- A screenshot that represents your product. This will be used when publishing your product in our web page (see this page for an example). Name it "["your team number"][V0.2]".png (e.g., [T11C2][V0.2].png), and upload to the "[V0.2] Publish" IVLE folder. png format is preferred to avoid quality loss from image compression.
- A blurb. This is a short description of your product, to be used when we publish your product. Word limit is 150. No line breaks please. Everything should be in a single paragraph. Upload it as a text file named "["your team number"][V0.2]".txt (e.g., [Q12][V0.2].txt) to the "[V0.2] Publish" IVLE folder.

Here is an example submission for the team T11C2:

IVLE folder	Files
V0.2 Product	[T11C2][V0.2].zip
V0.2 Video	[T11C2][V0.2].avi
V0.2 Dev guide	[T11C2][V0.2]DevGuide.pdf
V0.2 Publish	[T11C2][V0.2]UserGuide.pdf [T11C2][V0.2].exe [T11C2][V0.2].png [T11C2][V0.2].txt

V0.2 demo and Q&A session

This is a closed-door session, 30 minutes long, attended by evaluators and the team only.
The whole team is required to be present but it is not necessary that every member take part in the demo.

- Demo: You get 10 minutes to demo the product using the executable submitted. The scenarios you demonstrate should be chosen judiciously so that you cover the full range of your product's functionality. Rehearse the steps well and ensure you can finish the demo within the 10 minutes.
  - Dress code - none.
  - Punctuality - be there on time or else!
  - Use of slides - You may use slides during the demo, but not required.
- Q&A: Evaluator may ask you questions about the project, directed to the whole team or to a specific team member. These questions aim to check how well you know the part of the project that you claim to have done.
  Be prepared to answer these questions:
  - What are the strong points of your project?
  - What did you learn from this project?
  - What did you contribute to this project?
  - Show us the code you wrote.

V0.2 evaluation

V0.2 evaluation will NOT be on K/K basis (as this is your final submission, stricter grading criteria will be applied). Given below is the marking scheme. Note that you score for each component **will be 0%** if you are below what is mentioned under [score: 1%] for that component.
Total: 25%

a) Product features [max score: 5 %] - How good is your software as a product?
Evaluated based on the demo and acceptance testing done by the evaluator.

- Amount of work done
  - [score: 1%] Meets expectations.
  - [score: 2%] Exceeds expectations. i.e., an 'impressive' amount of work.

- Cohesiveness of features (do the features make up a 'whole' product instead of a random mix of features?)
    - [score: 1%] Just-enough features to consider it a usable product. Minor features may be missing.
    - [score: 2%] a complete and usable product for a given target user base (no need to have 'every feature under the sun').
- Full score bonus: extra 1% for getting max 4% for above

b) Design [max score: 5 %] - How good is your internal design? Evaluated solely based on what we can learn from the developer guide.

- Conformance with the top-down approach
    - [score: 1%] Good effort to apply top-down, but not entirely successful.
    - [score: 2%] Almost entirely top-down, except for occasional localized bottom-up optimizations
- Conformance with the OO paradigm
    - [score: 1%] Mostly OO, but occasional evidence of non-OO practices.
    - [score: 2%] Fully OO.
- Full score bonus: extra 1% for getting max 4% for above

c) Implementation [max score: 5 %]- How good is your implementation? Evaluated based on an inspection of the code. Note that if an individual's code is significantly below the average code quality of the product, that person will be penalized individually.

- [score: 3%] Reasonably good quality code for the most part.
- [score: 5%] Top quality code for the most part, easy-to-understand.

d) Testing [max score: 5 %] - How good is your testing? Evaluated based on Q&A session and the developer guide. If you have an automated test driver, you may be asked to run it on the spot.

- How much effort you have invested in the testing process
    - [score: 1%] Some effort on systematic testing.
    - [score: 2%] Comprehensive systematic testing with an ATD.
- How well you have tested the product
    - [score: 1%] Only minor bugs.
    - [score: 2%] No apparent bugs.
- Full score bonus: extra 1% for getting max 4% for above

e) Documentation [max score: 5 %]- How well does your user guide and developer guide achieve their objectives? Evaluated based on the user guide and developer guide.

- User guide
    - [score: 1%] Barely good enough to get started.
    - [score: 2%] Top notch user guide, both as a quick start guide and a user manual.
- Developer guide
    - [score: 1%] Barely good enough to get started.
    - [score: 2%] Top notch developer guide. Just enough content and easy to understand.
- Full score bonus: extra 1% for getting max 4% for above

f) Outstanding achievements bonus [max score 2%] - Any 'over the top' good things the regular marking scheme (a-e) does not capture.

CS2103 projects Hall of Fame

We plan to induct 8-10 teams to the *CS2103 Projects Hall of Fame* (Here is the hall of fame from 2010 Aug batch) The selection is by a vote from the teaching team and is based on overall impression of the product (not the marks earned from V0.2 evaluation). The purpose is to encourage you to go beyond what is required by the module and do something really 'off the charts'. All winners will be listed in a *CS2103 Projects Hall of Fame* web page that will be available to the public for the foreseeable future. Do not worry if you do not make it to the HoF; your project will still be listed in 'All projects' page and you can still show it as proof of your achievements in CS2103.

Forming teams

- **For CS2103:** We allow you to form your own teams (as long as all members are in the same

tutorial). If you are unable to form teams on your own, not to worry, we'll help you form teams. It is safer to form teams on your own because when we form teams, you may be asked to change the tutorial if there aren't enough team-less students in a tutorial to form a team.
For CS2103T: Teams are formed by the CS2101 instructor.

- The default team size is four. Please do not form teams with five or more.
- All team members need to be in the same tutorial. The reason for this was explained in the 'Tutorials' section above. We know you do not like this restriction. However, one of the aims of this restriction is to reduce your workload by using tutorials for project related discussions. If you are worried about getting stuck with 'bad team mates', we have increased contact with project tutor and peer evaluations (compared to previous semester) to mitigate that risk. Furthermore, we will do our best to move your preferred team mates to the same tutorial.
- You can use the "Project-related discussion" IVLE forum to get in touch with others looking for a team.
- **Team registration for CS2103T students**: Your team formation will be handled by CS2101 side. When you form teams for CS2101, please ensure all members can also attend the same tutorial on CS2103T side. i.e. one of the Wed morning tutorials. Once your team is formed, register the same team on our side using IVLE project sign up.
- **Team registration for CS2103 students**:
  If you have already formed a four-person team on your own: Pre-book a place for your team in your preferred tutorial by registering your team via IVLE project sign up (starting from Week 1 Monday) . Slot allocation is on a first come first served basis. When you bid for tutorials, give the same slot as your first choice. Do not give any more choices and do not give it as a choice for other modules. If the system did not allocate you the requested slot, please notify Mengran (our TA) to transfer you manually.
  If you have only three members: Go ahead and register your team via IVLE project sign up (starting from Week 1 Monday). But note that you may be required to accept another member to you team later.
  If you have only two members: Do *not* register your team using IVLE sign up. Just bid for any tutorial that fits with your language choice. Once tutorial placements are confirmed you can form teams with others in that tutorial.
  If you are unable to form teams early: Just bid for any tutorial that fits with your language choice. Once tutorial placements are confirmed you can form teams with others in that tutorial.
- You need to elect one of your team mates as the team leader. All members should be put in charge of various aspects of the project. e.g., testing, documentation, deadlines, etc. 'In charge' does not mean 'do everything', just 'responsible for the successful execution of that aspect'. For any given aspect, there should be one person who is in charge.

Supervision

In this module, you have two tutors: *tutorial tutor* and the *project tutor*. While tutorial tutor will discuss project related matters during the tutorial, he will not be able to track your project over time or attend to problems specific to your team. That is the job of the project tutor. Your project tutor will most likely attend your tutorial. You can meet him before/after the tutorial, or any other time, as many times you need, subject to room in his schedule. In addition, your project tutor may call you for meetings if required. However, please note that it is not the project tutor job to chase you down and give help. It is up to you to get as much guidance from the supervisor as you need. Just to be clear, do not expect your project tutor to code or debug for you.

Teamwork

We know that most of you plan to do an equal share of the project work, some are even willing to do more than the fair share. However, some of you may be forced to contribute less than an equal share due to reasons of your own. We have no problem with less-than-equal-share of contributions. Those who are unable to contribute an equal share of the work will not get the same project grade as others in the team. Some situations where you are very likely to have a significantly lower grade:

- You did not do any coding (note that each team member has to do at least *some* coding).
- You did all the coding yourself (If you strongly feel that none of the other members write "good enough" code, please get permission from the supervisor before taking over the whole coding aspect).

- You regularly missed team meetings.
- You regularly failed to deliver what your promised.
- You often did not adhere to team decisions and regularly took critical decisions on your own without consulting the team.

We use the online peer evaluation system TEAMMATES to conduct three rounds of peer-evaluations to determine the level of contribution from each member. During peer-evaluations, you will be asked to estimate the contribution of each team member (including yourself) to the project so far. The system also allows you to give anonymous feedback to your teammates.

Peer evaluation round 1 will be used to give early feedback to those who the team thinks 'not contributing enough'. This round will not affect your grade.
Rounds 2 and 3 may affect your grade in the following way:

- If everyone in the team is in agreement about how much each one contributed, we award grades based on the contribution levels submitted through the peer evaluation. i.e., if everyone agrees you did more than others, you get a higher grade.
- If there are significant discrepancies in the peer evaluations (e.g., you say you did an [equal share + 20%] while your team says you did an [equal share - 10%]), we will investigate further before finalizing project grades. For such teams, we are likely to request some or all members to submit an additional individual report to itemize and quantify the contribution to the project. Overclaimers will be penalized stiffly. Please do not claim high just to 'see how'.

A final note on this issue : Please be sincere about how much you can do from the beginning, rather than let your team mates figure it out over the long run. Tell them how much you can do and what kind of work you are willing to do. If you are not open about it, they will still figure it out at the end, they will still report it (via the peer-evaluations), and your score will be adjusted all the same. Being open about it on the other hand can save everyone a lot of frustrations, unpleasantness, and bitterness.

Project communication
Keeping a record of communications among your team can help you, and us, in many ways. For that purpose, we use Google groups. After forming a team, please create a Google group for your project

- Add cs2103ProjectMonitor@gmail.com as one of the members (use the 'add members directly' option, not the 'invite members by email' option). That will help us to monitor your project discussions if there is such a need.
- Add your project tutor to the Google group. You can find your supervisor's email from the 'Teaching team' page.

Soon after you are assigned a team id (around week 3), please reconfigure your Google group as follows.

- The name of your group should start with "[cs2103aug11]["your team number"]". For example, "[cs2103aug11][W9J2]APlus" is an acceptable name.
- Configure the email address of the group to be **cs2103aug11XXXX@googlegroups.com** where XXXX is your team number. e.g., cs2103aug11W9J2@googlegroups.com. Group administrator can do this by navigating 'More' -> 'Settings' -> 'Group settings' -> 'General' -> and editing the relevant settings
- Set the subject prefix of your Google group to "["**cs2103aug11XXXX**"]" where XXXX is your team number. Group administrator can do this by navigating 'More' -> 'Settings' -> 'Group settings' -> 'Email delivery' -> and editing the 'Prefix' setting.
- As much as possible, CC your communications with the supervisor to your Google group. That way, you will have all your project related communication in one place.

Please follow the above configuration closely as we subject project emails to some automated processing.

Note that we do not monitor all the Google groups all the time. It is a tool we intend to use 'as needed'. If you expecting a direct answer from the lecturer or project tutor, please send a direct email as well.

All project-related communication is to be done in English, even if all team members understand another

language better than English.

If you had a project related discussion in person (i.e. not using email), assign one team member to post minutes (can be brief) of your discussion. i.e., tasks done, important decisions taken, etc. You can also post chat scripts from meetings held via IM. Keeping records of project communications in the Google group helps us keep tabs on what's happening in the team and will also serve as evidence if your team runs into teamwork issues later on. You don't have to report everything in the Google group, but don't let the group go completely silent either.

Module policies

About following instructions: When working with others, especially in a large class such as CS2103, it is very important that you adhere to standards, policies, and instructions imposed on everyone. Not doing so creates unnecessary headaches for everyone and sends negative signals about your working attitude. Therefore, we impose an automatic penalty for not following instructions strictly.

Policy on email response time:

I have a self-imposed policy on response time: I will respond to you within 24 hours if it was an email sent to me or a forum post directed at me. If you didn't get a response within that time, please feel free to remind me. It is likely that I did not notice your post or the email got stuck somewhere.

Policy on plagiarism:

- You are not allowed to share individual assignments with classmates directly.
- You are not allowed to share project-related things with other teams directly.

However, note that most submissions will be published to the class after the submission deadline, after which they can be 'reused' by others subject to the policy on reuse given below. While the main objective of this to let you learn from each other, it also acts as a very effective deterrent against plagiarism. If you copy from another team, it is very likely that someone else in the class will notice this and report to us even if we failed to notice it ourselves.

Policy on reuse:

In general, reuse is encouraged. However, note that reuse has its own costs (such as the learning curve, additional complexity, usage restrictions, and unknown bugs). Furthermore, you will not be given credit for work done by others. Rather, you will be given credit for *using* work done by others. Always get permission from the supervisor before you reuse things from elsewhere.

- You are allowed to reuse work from your classmates, subject to following conditions:
  - The work has been made public by us.
  - You clearly give credit to the source.
- You are allowed to reuse work from outside sources, subject to following conditions:
  - The work comes from a source of 'good standing' (such as an established open source project) and has been publicly available before the start of the module. This means you cannot reuse code written by an outside 'friend'.
  - You clearly give credit to the original author. Acknowledge use of third party resources clearly. e.g., in the welcome message, splash screen (if any) or under the 'about' menu. If you are open about reuse, you are less likely to get into trouble if you unintentionally reused something copyrighted (unknown to you).
  - You do not violate the license under which the work has been released. Please do not use 3rd-party images and soundtracks in your software unless they have been specifically released to be used freely. Just because you found it in the Internet does not mean it is free for reuse. Google search is not a source of free files.

Policy on help from outsiders:

In general, you are not allowed to involve outsiders in your project except your team members and the teaching team. However, It is OK to give your product to others for the purpose of getting user feedback. In addition, if you know an outsider who is willing to help you on some aspect, please talk to us first to see if that can be allowed. We can allow such help if it will not disadvantage other teams or if such help can be made available to all teams equally.

Tech help:

Some of our project tutors have volunteered to help you in technical problems if they can. If you need their help, please post your problem in the respective "tech help" forums in IVLE. You are discouraged from contacting the tech experts directly. This is so that whatever help being given is given openly so that other classmates can benefit from it. A solution is not guaranteed. However, they will do their best to help you.

FAQ

- **How does 'suggested length' (for document submissions) work?**
  By giving a suggested length we want to encourage you to write shorter documents that contain the right amount of details to communicate effectively whatever that you are trying to communicate. We do not read beyond this suggested length unless what we have read up to that point makes us want to read more. Therefore, it is in your interest to stick to the suggested length. But if you think you have more good stuff than that would fit in the page limit, go ahead and include them by all means.
- **Why don't we specify what exactly to put in submission documents?**
  Because there's no right answer. The best way to communicate something often depends on what is being communicated. Therefore, we aim to refine project documents iteratively. We believe the learning experience will be richer if we let *you* decide the best way to present your project information rather than just following our instructions blindly.
  In real-life projects, you are rarely told which diagrams to draw; that is a decision you have to make yourself.
- **Why do we number releases 0.1, 0.2 etc. instead of 1.0, 2.0 etc.?**
  Because what you do in this project is meant to be the initial part of a bigger project.
- **Why do we grade initial submissions for effort rather than quality?**
  Because we want you to learn from your mistakes without incurring penalties for doing so.
- **Doesn't publishing submissions unfair to the team who submitted them?**
  We publish your submissions only after giving you the credit your deserve for it. Besides, if you were the first to think of something your peers are willing to adopt later, that means you are already ahead of them and they are unlikely to earn more marks by adopting your ideas.
- **Must we have a GUI for the product?**
  Whether to make your UI textual (menu-based or command-based) or graphical depends on the type of the product. Often, products have both textual UIs and GUIs. However, you can defer the implementation of the GUI to a later iteration, to be done by whoever that will take over the project after you. Treat the GUI as just another feature that you deliver at some point in the project. .You are welcome to do a fancy GUI if you have expertise or willing to spend lot of effort on it. Otherwise, just doing a basic GUI or a text UI will not disadvantage you in any way. In fact, text UIs have regularly appeared among top scorer's in previous years.

Exams

- No midterm.
- Final exam - 2 hours, open book, essay type questions (no MCQ), covers materials from all lectures, handouts, and tutorials. Questions are likely to be similar to what was discussed during tutorials, but based on a different project/product.
  Due to increased emphasis on the project this semester, the exam is expected to be slightly easier than previous semesters.
  The exam will be easy for you if you attend lectures, participate in tutorials, and do a sincere job in the project. However, you will not be able to ace the exams by doing past papers just before the exams. In fact, the style of this semester's paper will be different to past papers.
  Sample exam questions to be given in week 13.

Grading

Grade breakdown

- [40%] Final exam.
- [50%] Project.
    - [4%] CE1 (grading: K/K based)
    - [4%] CE2 (grading: K/K based)
    - [5%] Project proposal (grading: K/K based)
    - [7%] V0.1 (grading: K/K based)
        - [4%] Product V0.1
        - [3%] Project manual V0.1
    - [25%] V0.2
        - Please refer the 'V0.2 evaluation' section for more details
    - [5%] Project participation
- [10%] Participation

  Here are ways to earn participation points.
    - Attend a lecture + answer pop quizz for that lecture (1 point per lecture), provided you arrived no later than 10 minutes into the lecture.
    - Attend and follow tutorial (1 point per tutorial), provided you stayed 40 minutes in the tutorial.
    - Other ad hoc opportunities offered during the module (e.g., submit a survey, answer a question during a lecture).

  To earn full 10%, you should earn 20 participation points (the maximum points achievable will be around 25). 19 points → 9%, 18 points → 8%, …and so on, until 10 points or fewer → 0%.

K/K mode grading

- We use *K/K mode* of grading during early stages of the module. This mode is applied when learning from mistakes is more important than the assessment.
    - We give full marks if we see evidence of "genuine effort", although the outcome may not be "optimal" in the eye of the evaluator.
    - We give a "Kita" if we cannot see a genuine effort. A Kita means "you are in danger of getting 0 for this submission, unless you improve a lot in the next submission".
    - Kita receivers can still earn up to 100% of the marks for that submission if they do a good job in the next submission.
    - We give a "Kudos" for those who did a great job, at least in some aspect of that submission. Kudos can come with extra participation marks.
    - Both Kita cases and Kudos cases are reviewed by the whole teaching team so that there is uniformity across the class.
- In summary
    - if you didn't hear from us, you did a reasonable job and you get full marks.
    - if you received a Kudos, you have gotten full marks and we think you did a great job.
    - if you received a Kita, you can still earn up to 100% of the marks if you do a *substantially* better job next time. If not, you will get zero marks.
- This mode of grading has the following benefits:
    - It reduces the subjectivity of grading, especially when multiple evaluators are involved. This is because most of the class receives the same grade.
    - It allows us to give negative feedback when necessary, without demotivating you too much. You can recover from a Kita without losing any marks.
    - It does discourage unhealthy competition among project teams during early stages of the project. That is because you need not be the "best in class" to earn full marks for a submission. However, those who are genuinely motivated will work harder to earn "Kudos".
    - It gives slow teams a chance to catch up.
    - It fits "no right answer" situations and "learning from trial-and-error' situations encountered during SE projects.

--- end of document ---