COSC2299/2428 Software Engineering: Process and Tools
RMIT University

Major Assignment, Part 1 – Semester 1 2016 – DRAFT
Due: 9AM Monday April 18 2016

The current assignment specification is considered DRAFT until Week 3. Please send any issues, problems or requests for clarification to the lecturer ASAP so that the spec may be fixed/updated.

Assignment Overview

You are required to develop an application matching the specification below, using appropriate process and tools, in particular those supporting collaborative project and code development. The program may be implemented in Java but does not need to be: i.e., you may use a different language and framework if you prefer.

This is a team assignment; Part 1 is worth 10% towards your final grade.
Team members will be required to indicate how and how much each team member contributed to the final submission.

Marks for individual team members may be adjusted to reflect level and quality of contribution, as indicated by peer assessments and logs from collaboration tools.

Team Issues

Teams will be self-organised; support for this will be provided the first tutorial, and by the course coordinator if required. Teams should consist of 3 or 4 members. There are no bonus marks for a small team. **Anyone who is unable to form a complete team must contact the lecturer by March 15**!!
Any issues that arise within the team should be resolved by the team if possible; if this is not possible, then this should be brought to the attention of the course coordinator as soon as possible.

Academic Integrity

The submitted assignment must be your own team's work. No marks will be awarded for any parts which are not created by your team.
Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students (other than those in your team), internet or other resources ***without proper reference***. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that your team should always create its own assignment even if you have very similar ideas to other teams. Penalties may be applied in cases of plagiarism.

What to do

Part 1 of the Major Assignment is to analyse a set of requirements, design and implement a system. You are only expected to implement a small subset of the complete system that is to be implemented this semester (in further parts). The extra functionality will be explained further in later specifications. Despite consisting of only partial functionality, you are expected to implement it correctly and perform a proper release. This means that the program you provide must:

1. run correctly;
2. have an appropriate level of testing. This means you should have checked the functionality of your system against the specifications below. Any errors discovered should be fixed. You should have a list of final acceptance tests that you make whenever you build your software;
3. contain appropriate user documentation that states how to install and compile the software, as well as how to use it;
4. be easy to use (i.e. have an intuitive user interface).

Your submission will contain not only the program and associated documentation, but evidence (e.g., logs) of the use of appropriate collaboration tools, including a task management tool such as Trello or JIRA; a communication tool such as Slack or HipChat; a code management tool such as Git; and any other output as appropriate (e.g., the output of unit testing).

The Requirements are listed below (in Client Requirements).

Things you are advised to do include:

1. Identify User Stories for the system. You should document these and use them to form the basis of your task plans. **These will not be directly assessed BUT you will need to show that you have written tasks for your task planning**—i.e., the actual use cases / scenarios won't be assessed but if you don't have any you will almost certainly lose marks associated with use of Trello/JIRA.
2. Create a class diagram of the problem which includes appropriate attributes, methods and relationships.
3. Your user documentation should be constructed as HTML documents, and include screen shots of your program. Images should be in one of the formats: jpeg, gif, png. You should check that your HTML files are rendered correctly on a Unix based machine as well as when using a Windows based browser to ensure they do not contain malformed HTML.(A very common mistake made by Windows users is to use the '\' character as a directory separator when specifying file paths. The HTML standard requires the use of the '/' character instead.) You can check your HTML at http://validator.w3.org. Passing validation at this site is is recommended but not required.

Client Requirements

You are required to retrieve data from Australia's Bureau of Meteorology website (www.bom.gov.au) (see http://www.bom.gov.au/catalogue/data-feeds.shtml for details on getting the information) for weather stations around Australia and its Territories (including the Australian Antarctic Territory).
For example a list of sites for the Sydney area is at
http://www.bom.gov.au/weather/nsw/observations/sydney.shtml

Links from this page take you to the recordings for individual stations.

Functional requirements

1. The system should allow users to select weather stations they want to keep track of (their favourites) - these should be stored by the program for when the program starts again.
2. The weather stations and the associated data are displayed as a **table** in a separate window. (The word "table" replaces the previous word "chart".)
3. When a favourite is chosen, the program should get as much weather information about it as possible from the BOM site. The information you retrieve should be stored in by your program so that a record of temperatures can be created and viewed.
4. The minimum information that should be presented...
    a. Each location records a number of different measurements - these should all be captured from the web site and stored/presented to the user. For example (Laverton, Victoria) can be found here: http://www.bom.gov.au/climate/dwo/IDCJDW3043.latest.shtml.
    b. The program should be able to graph the temperature history for each site in the favourites list, including the max, min, 9am and 3pm temperatures.
5. The program should have either a refresh button (to check if updated information is available) or a continually running background thread that automatically refreshes the information. The information should be automatically refreshed when the program starts up.
6. Management of windows: You should record the screen position of any windows when the application quits, so that the windows will reappear in the same location when the program restarts. You will need to think about how your program will cope when it is run on another computer with a different number of monitors or monitor sizes.
   IF you are implementing the program as a web or mobile app, then you will not necessarily have windows at different locations. However, your app must restart in the same state as when it was last closed: e.g., the same "favourites" selected and the same charts opened.

Submission Instructions

This assignment is to be submitted by 9AM Monday 18th April 2016. It should also be demonstrated by your team to your lab assistant in Week 7.

Include the following in all submissions:

- Names and student number for all team members;
- A short description of the contributions of each team member to the submission, including a % contribution of each team member (these should add up to 100%) and a statement of ***what*** part of the submission each team member was responsible for.

Marking criteria for the assignment will be discussed in class and published on Blackboard, but marking criteria will include:

- how well your software meets the requirements at the demonstration;
- quality of the software produced, including quality of documentation/comments and adherence to programming standards;
- adherence to process and use of the planning and collaboration tools;
- how consistently you work throughout the project, based on weekly meetings/demos and details found in your code repository.

Marks for individual team members may be adjusted to reflect level and quality of contribution, as indicated by peer assessments and logs from collaboration tools.

All assignment **submission is electronic** – instructions will be given on Blackboard. Assignments must be submitted by the due date/time – **late assignments** may not be accepted at all.

You are required to manage your time, and required to work consistently before the assignment is due. As you can submit your work electronically you should do so any day a change is made to your project.
If you are ill, and are unable to progress on your work, you should submit a request for special consideration. We will then calculate the amount of time you were unable to progress on your work as a proportion of the time available for the assignment, as well as assessing the work you have submitted. In general teams should be able to cope with the temporary loss of a team member for a short time. If a team member is absent for any length of time, you should consult your lecturer or head tutor.
We may then make an appropriate adjustment to your team's result.
Illnesses of 1 or 2 days are unlikely to affect your work greatly, and are thus unlikely to be considered justifiable reasons for your work not progressing.
If you are ill and unable to work for either a major period (>50%) or the entire period of the assignment part, you should contact your student advisor to discuss your situation.

A consequence of not submitting an assignment/part on time is that your team will get 0 marks for that assignment/part.