**This is the Final Milestone Submission Instruction Document and also should be used as a Template for Your Submission**

**Full Marks**: 100 (15% to final grade)

**Submitted by**

Iain Workman

Mitchell Corbett

John Mason

Matthew Galbraith

Ryan LaForge

**Submitted to**

The students of CMPT 370 (2013-14)

University of Saskatchewan

November 18, 2013, Saskatoon

***Abstract: In the abstract you are supposed to provide an overall summary of your project in a similar way to your milestone 2. Just provide a paragraph of not more than 200 words about your project. You can mention what the project is about, what its business values are and how successful you have been in working in this project. Also write about your feelings of working in this project. Abstract should be of font size 10 Times New Roman. All other text will be 12 Times New Roman. Section headings should be 13 Times New Roman and sub-section headings should be 12 Times New Roman. You should use standard paper size of US letter. Before you write the abstract you must provide a “cover page” similar to the “cover page” of this milestone. You must include the project title, group name, group ID, submission date, NSIDs/Names of the group members in the cover page. Also provide a Table of Contents of your project similarly to the one in this document. This final submission has a different marking scheme as provided in Appendix A. It is also mandatory that you use the word version of this document for your submission. If you have been working with Latex or other editors pls. let e know first and I will try to help out.*** (2 marks)

**\*\*\*Read the *abstract* carefully before working on this milestone\*\*\***

Note: If you need a latex version of this document, pls. feel free to send me an email.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table of Contents** | | | **Page No.** |
|  |  | |  |
|  | **Abstract** (2 marks) | | **2** |
|  |  | |  |
| **1.** | **Introduction** (2 marks) | | **5** |
|  |  |  |  |
|  | **1.1** | **System description** (0.25 mark) | **5** |
|  | **1.2** | **Business case** (0.25 mark) | **5** |
|  | **1.3** | **User-level goals for the system** (0.25 mark) | **5** |
|  | **1.4** | **User scenarios** (0.25 mark) | **5** |
|  | **1.5** | **Scope document** (0.25 mark) | **5** |
|  | **1.6** | **Project plan / rough estimates** (0.25 mark) | **5** |
|  | **1.7** | **User involvement plan** (0.25 mark) | **5** |
|  | **1.8** | **Low fidelity prototypes** (0.25 mark) | **5** |
|  |  | |  |
| **2.** | **Requirements and Early Design** (17 marks) | | **6** |
|  |  |  |  |
|  | **2.1** | **Summary Use Cases** (2+ 1 marks) | **6** |
|  | **2.2** | **Fully-dressed Use Cases** (1 marks) | **6** |
|  | **2.3** | **Use Case Diagram** (1 mark) | **6** |
|  | **2.4** | **Domain Model** (2 marks) | **6** |
|  | **2.5** | **Glossary** (2 mark) | **7** |
|  | **2.6** | **Supplementary Specification** (2 marks) | **7** |
|  | **2.7** | **System Sequence Diagrams** (3 marks) | **7** |
|  | **2.8** | **Operation Contracts** (2 marks) | **7** |
|  | **2.9** | **Obtaining User Feedback** (1 marks) | **7** |
|  |  |  |  |
| **3.** | **Updated Design and Testing** (14 mark) | | **8** |
|  |  |  |  |
|  | **3.1** | **System Operations** (1 marks) | **8** |
|  | **3.2** | **Sequence or Communication Diagrams with GRASP Patterns** (2 marks) | **8** |
|  | **3.3** | **Class Diagram** (3 marks) | **8** |
|  | **3.4** | **Unit Testing** (8 marks) | **8** |
|  |  |  |  |
| **4.** | **Reengineering** (22 marks + 2 bonus marks) | | **9** |
|  |  |  |  |
|  | **4.1** | **Code Smells** (7 + 6 marks) (Also 2 bonus marks for the second part) | **9** |
|  | **4.2** | **Refactoring** (5 marks) | **11** |
|  | **4.3** | **Gang of Four Design Patterns** (4 marks) | **11** |
|  |  |  |  |
| **5.** | **Complete Implementation and Product Delivery** (33 marks) | | **12** |
|  |  |  |  |
|  | **5.1** | **Naming Conventions** (1 marks) | **12** |
|  | **5.2** | **Commenting** (1 marks) | **12** |
|  | **5.3** | **Pretty-printing of the source** (2 marks) | **12** |
|  | **5.4** | **Usability Engineering** (7 marks) | **12** |
|  | **5.5** | **Excellency of the Complete Implementation** (15 marks) | **13** |
|  | **5.6** | **User Manual** (7 marks) | **13** |
|  |  |  |  |
| **6.** | **Project plan, Budget Justification and Performance Evaluation** (7 marks) | | **13** |
|  |  |  |  |
| **7.** | **Conclusion** (1 marks) | | **17** |
|  |  |  |  |
|  | **Acknowledgements** (1 marks) | | **17** |
|  | **How to Submit?** (0 marks, penalty of 2 marks if not followed) | | **17** |
|  | **References** (1 marks) | | **18** |

**1. Introduction** (2 marks)

After your provide the table of contents, this is the time for writing the Introduction of the report. Similar to your Milestone 3, you must provide the following about your submission. In particular, your group will submit an executive summary of your product. This should capture the expectations of the stakeholders, and should include the following:

**1.1 System description**

A high level description of the system from a business and user perspective. Make it very clear what the purpose of the system is and any important high level aspects of how it will accomplish this task.

**1.2 Business case**

What problems will this solve for your target audience? What competitive advantages does it offer? How can it improve efficiency, effectiveness or workers? If it is for entertainment, then what will make it compelling for users?

**1.3 User-level goals for the system**

State the main expectations and requirements the user will have for your system.

**1.4 User scenarios**

For every main user task that involves your system, describe a realistic scenario that shows when the user will use the system and what for. Do not include a detailed description of how the user will interact with the system, but rather describe how the system will assist the user in their workflow.

**1.5 Scope document**

Describe what your system will and will not do when it is finished. Be careful to make the scope match with what your estimated budget in hours can afford and be conservative with estimates.

**1.6 Project plan / Rough estimates**

Break down the project into individual tasks, and for each task, provide a rough estimate of the number of hours it will take to complete. This portion must be similar to your Milestone 3 submission and should not include the actual time spent or so. Your will provide the details about actual time spent in a later section below.

**1.7 User involvement plan**

You will be involving users throughout your development process. Write a plan for how you intend to involve users, including which users you will involve, how much of their time you will need, when you will need it, and why. Again, this could be kind of similar to your Milestone 3 submission.

**1.8 Low fidelity prototypes**

For each of the main interfaces, provide a pencil or whiteboard drawing of it showing your approach to design. Consider these UI designs preliminary, for discussion purposes, and by no means final. Mostly will be similar to your Milestone 3 submission. However, you are welcome to update this part that may reflect your current status of the project. Also address the comments you received from the markers.

**2. Requirements and Early Design** (17 marks)

We have been following RUP and many of the work we have done in the *Introduction* above is part of the *Inception Phase* of RUP. This is our time to further work on *Inception Phase* and in the same time we have to work on the *Elaboration Phase*. In particular, address the following issues and you are welcome to note any additional things that you have considered (in case). You must use UML drawing tools (of your choice) for the different UML diagrams. Whenever you are using or referring a third party thing, you must cite (see e.g. the *References* section below) that to your document.

**2.1 Summary Use Cases** (2 +1 marks)

(a) List the names of all the use cases of the system and provide a two line description for each of them.

(b) From the list above choose eight important use cases and write down their *summary use cases* if there are four members in your group. Similarly, choose six important ones and write their summary use cases if you are three in the group, i.e., two summary use cases per group member.

**2.2 Fully-dressed Use Cases** (1 marks)

Write down four *fully-dressed use cases* covering the major features of your project if you are four in your group. Similarly, if you are three in your group, provide three and so on. If your system does not have the scope of the required number of *fully-dressed use cases*, you are welcome to break down the big ones into smaller ones. You might also want to clearly mention why you do not have sufficient use cases in case you cannot break then out.

**2.3 Use Case Diagram** (1 mark)

Draw a *Use Case Diagram* for your project. You must show the changes that you have made compared to Milestone 4 submission. Show all changes with a different colour or list the new use cases below the diagram. You must use a UML drawing tool of your choice.

**2.4 Domain Model** (2 marks)

Build a *Domain Model* for your project. This domain model should be an updated model from your Milestone 4 and you must clearly indicate (e.g., showing with a different colour) of what changes have been made in this domain model compared to the earlier one. For each of the changes note what were the reasons behind the change. Instead of thinking in terms of design objects (e.g., Java class), consider in terms of real-world objects at this stage. Again, you must use a UML drawing tool of your choice.

**2.5 Glossary** (2 marks)

Build a *Glossary* defining the common terminologies of your project. Again, you must clearly indicate what terminologies have been added compared to Milestones 4.

**2.6 Supplementary Specification** (2 marks)

The *Supplementary Specification*captures other requirements, information, and constraints not easily captured in the use cases or Glossary, including system-wide "URPS+" (**u**sability, **r**eliability, **p**erformance, **s**upportability, and more) quality attributes or requirements. Read Section 7.4 of Chapter 7: Other Requirements from Larman and write down whether there are any supplementary specifications for your project. Note this supplementary specification was not asked in earlier milestones.

**2.7 System Sequence Diagrams** (3 marks)

Draw four *System Sequence Diagrams (SSDs)* if you are four in your group (and three if you are three) for the success scenarios of the major (those are fully-dressed above for example) *use cases* of your project. Follow all the instructions that were discussed in the lecture. Use a UML drawing tool of your choice. Are there any updates from your Milestone 4 submission? Note all the changes below the diagrams.

If your system does not have the scope of the required number ofSSDs, you are welcome to break down the big ones into smaller ones. You might also want to clearly mention why you do not have sufficient SSDs in case you cannot break then out.

**2.8 Operation Contracts** (2 marks)

Write down the operation contracts for the key operations (taken from the System Sequence Diagrams above) of your project. You should have two important operation contracts per group member.

**2.9 Obtaining User Feedback** (1 mark)

Write down the general process that you had been using to obtain user feedback in the *Inception* and *Elaboration* phases of RUP.

**3. Updated Design and Unit Testing** (14 marks)

At this stage, you were mostly done with the *Inception* phase and had been working on the *Elaboration* and *Construction* phases of RUP. It means that you had been heavily working on improving the design, implementing your system and testing. You should thus provide the following for this stage:

**3.1 System Operations** (1 mark)

Look at your system sequence diagrams (SSDs) from previous section and list all the system operations of your project. You should name at least four system operations per group member.

**3.2 Sequence or Communication Diagrams with GRASP Patterns** (2 marks)

Choose two important system operations per group member from the list above and draw sequence diagrams (or communication diagrams) for them (two diagrams per group member). You must consider GRASP patterns when drawing the diagrams and clearly indicate which GRASP pattern has been used for what. You must show the application of at least one GRASP pattern per group member. Write down the rationale of their applications in a separate paragraph below each of the diagrams. Of course, use a UML drawing tool of your choice.

**3.3 Class Diagram** (3 marks)

Draw a class diagram of your project. This class diagram must reflect your current implementation, i.e., there should have a direct mapping between the design objects and the actual classes in your project. The objects must include the attributes and methods reflecting your implementation. The class diagram will reflect your knowledge of GRASP patterns and the gang of four design patterns. Furthermore, clearly indicate the changes you made from your Milestone 5 submission. Of course, use a UML drawing tool of your choice.

**3.4 Unit Testing** (8 marks)

Given that you were mostly done with your design and you had been heavily working on implementing the system, you were required to do unit testing for each of the new functionalities you added. Report your experience of applying JUnit [3] in your project. Provide at least a couple of situations of how JUnit was used and whether it was useful for your project. Provide the name of the subject method/class to test, the sample testing data, and the source code of applying JUnit in your project along with any comments you have. Those of you are not using Java, use a corresponding widely used unit testing tool for this part (or ask me if you do not find one). In addition, provide the locations (in terms of full file names and begin-end line numbers) of these testing details in your source code. Yes, provide everything in this document and also provide the location in the source.

**4. Reengineering** (22 marks + 2 bonus marks)

At this stage your implementation is done (almost at the end of the *Construction* phase of RUP) and you would like to deliver the product to the customer. However, before you deliver the product, you have to make sure about the professional quality of your product. You thus now plan to improve the design of the project by refactoring and in the same time you would like them to be aware of some of the bad smells (in particular software/code clones) which might not be possible to remove at this stage. You are also interested to apply the gang of four design patterns for improving the design further. You should thus focus on the following issues:

**4.1 Code Smells** (7 marks + 6 marks) (Also 2 bonus marks for the second part)

There might be many different types of clone smells in your system, and duplicated code fragments (also called software clones or code clones) are the number one bad code bad smell. We thus must be (at least) aware of them. We also have to know what are the root causes of such code clones (e.g., why did you make that particular clone? Was there any way of avoiding that clone?)? We then have to decide whether such clones should be refactored or kept in the system? If we decide to refactor what are the possible risks in refactoring? First, detect the code clones from your project (or your TA will be doing it for you). If you want to detect the clones by yourself, you can use my clone detection tool NiCad [1] or you may use other tools available in the clone detection literature [2]. For example, you might use *CCFinder* [4] or *Simian* [5].***)*** Just send an email to me (with your group ID) and I will give you the clones from your project. Once the clones are detected, your job is to guess (***yes, just “your intuition”***) why you guys have created such clones? If you have a quick look on my technical report [6] (Section 2, pages 3-7), you will find the details of possible root causes of clones. In particular, for each of the clones you might consider the following taxonomy in deciding the possible reasons for their existence. If you believe there are some other reasons, you should also note that. Because you created that clones, you better know why you did so.

**Taxonomy of Root Causes of Code Clones** (derived from Roy and Cordy’s technical report [6] Section 2, pages 3-7):

1. *Development Strategy*
   1. *Reuse Approach* 
      1. *Simple reuse by copy and paste*
      2. *Forking*
      3. *Design reuse*
      4. *Functionalities /Logic reuse*
   2. *Programming Approach* 
      1. *Generative programming*
      2. *Merging similar systems*
      3. *Delay in restructuring*
2. *Maintenance Benefits*
   1. *Avoiding* 
      1. *Risk*
      2. *Unwanted design dependencies*
   2. *Ensuring* 
      1. *Robustness*
      2. *Better performance in real time programs*
   3. *Reflecting* 
      1. *Design decisions (e.g. cross-cutting concerns)*
3. *Overcoming Underlying Limitations*
   1. *Language Limitations* 
      1. *Lack of reuse mechanisms*
      2. *Abstraction creates complexity*
      3. *Abstraction is error-prone*
      4. *Significant efforts in making abstractions*
   2. *Programmers' Limitations*
      1. *Time Limitations*
      2. *Performance by LOC*
      3. *Lack of ownership*
      4. *Lack of knowledge in the domain*
      5. *Difficulty in understanding large systems*
4. *Cloning by Accident*
   1. *Language Paradigm* 
      1. *Protocols to interact with API and Libraries*
   2. *Programmers' Working Style* 
      1. *Programmer’s mental model*
      2. *Unknowingly implementing the same logic by different programmers*

The TA or I will provide you the clones where each clone class/pair will have an ID number. For each of the clone pairs/classes, ***you just need to say what were the possible reasons for that clone pair/class from the taxonomy above***. For example, for a clone pair with ID ***n1***, you believe that ***reusing the logic/functionalities*** was the primary reason for its existence, and it matches with ***1.a(iv)*** in the above taxonomy, your answer will be as follows:

|  |  |  |
| --- | --- | --- |
| **Clone Pair/**  **Class ID** | **Root causes** | **Further comments** (e.g., those causes not covered in the taxonomy above) |
| ***n1*** | 1.a(iv) | ….. |
|  |  |  |
|  |  |  |
|  |  |  |

As a part of Section 4.2 below, you will then try to refactor the obvious and easy clones from the system.

However, not all clones are refactorable and thus they should be kept in the system. In such cases, provide the rationale of why these clones are not refactorable (and what are the risks of refactoring). You might also get an idea from the taxonomy above for finding the reasons. In addition to the 6 marks assigned for this part, you might get **two more bonus marks** here if you do a good job! In particular, you might use a table as follows:

|  |  |
| --- | --- |
| **Clone pair/ Class ID** | **Why Not Refactorable/Risks in refactoring** |
| ***n2*** | Not possible to refactor because c# does not allow procedural abstraction (**just an example!**) |
|  |  |

**4.2 Refactoring** (5 marks)

We have talked about a significant number of refactorings in the lecture. You must consider all those with respect to your project. Indicate the applications of at least one such refactoring per group member in your code. You first have to say which refactoring, and then say where (full file names and begin-end line numbers of the location) in the source it was applied and then say why it was applied. How did this subject refactoring improve the design of your project? You should also make use of *Eclipse* [7] in doing these refactoring as discussed in the tutorial. You might get help from a tutorial of University of Manitoba [8]. You may use other IDE if you wish.

**4.3 Gang of Four Design Patterns** (4 marks)

You must make the best use of the Gang of Four Design patterns that have been discussed in the lecture and in the tutorial. In particular, you must at least use one of the design patterns in your implementation (one per group). If you have done so, write down the location of where the design pattern has been used. You might get help from a tutorial of University of Manitoba [9] *(However, I am NOT expecting you to use CodePro as discussed in this document)*.

**5. Complete Implementation and Product Delivery** (33 marks)

Your implementation has now been completed, several re-engineering tasks (e.g., refactoring) have now been applied and that the system is now fully tested. You are basically in the *Transition* phase of RUP. At this stage you must focus on the following issues if not done already in the previous milestones:

**5.1 Naming Conventions** (1 mark)

Follow proper naming conventions while coding. If this is not done already, you should follow Chapter 11: The Power of Variable Names of the text “*Code Complete 2*” by Steve McConnell or follow the tutorial slides posted in Moodle.

**5.2 Commenting** (1 mark)

If not done already, provide appropriate and useful comments in your code. In particular, for each class, provide comments about it (e.g., who has implemented that class, what it is and what it does). Similarly for each of the methods provide appropriate comments. You should also go to the line level commenting. You might find Chapter 32: Self-Documenting Code of “*Code Complete 2”* useful or follow the tutorial slides posted in Moodle.

**5.3 Pretty-printing of the source** (2 marks)

Use *Artistic Style* [10] and apply pretty-printing to your project. You must show the evidence (e.g., the running command, any problems you have been encountered and so) that you did use this tool.

**5.4 Usability Engineering** (7 marks)

As we discussed in the lecture, good user interfaces are very important for the success of a project. You must thus consider the issues we talked about the properties of a good user interfaces and attempt to follow all those in your project. You must design your system by balancing the *ease of learning* and *easy of use* usability properties. In particular, you must follow the following user interface design guidelines from the lecture:

1. Know the user
2. Minimize memorization
3. Optimize operations
4. Engineer for errors
5. Avoid modes
6. Be consistent

You should have a look on the lecture slides for details about the above points. Provide some examples (at least one example per group member) for each of the points about. For example, you can indicate the names/locations of the Windows where you have considered what design guidelines.

**5.5 Complete Implementation** (15 marks)

Provide the complete implementation not only for the success scenarios of the major use cases but also for the alternative/failure scenarios. This is your final submission and thus you must have the implementation completed. The marker will closely look at your implementation and will evaluate with respect to several parameters such as the size of the project (not in terms of lines of code but in terms of the number of interesting functionalities addressed), technical challenges considered and tackled, correctness, and the applications of GRASP patterns, refactoring, gang of four design patterns (only those discussed in the lecture and tutorial) and user-interface design guidelines. You must thus try to make the best use of these factors if you want to get good marks for this section. If you want, you can write your experiences in working on some interesting and challenging things (in case you did) that deserve the marker’s attention here. This is the section where the marker and **the instructor** will try to determine the ***Excellency*** level of a project.

*(Note that in other sections I have also assigned marks for many of these factors, but I just asked for a minimum requirement. Here, the marker will try to figure out whether you have made the best use of such factors in terms of the implementation).*

**5.6 User Manual** (7 marks)

Your product is now ready for delivering to the customer and thus you must have to have a good user manual for your system. Provide a good (both ease of learning and ease of use) user manual for your system. It should include the details of installing and using your system. You also have to provide sample data (both input and output) for running your system on an overall level (e.g., for all major use cases).

**6. Project plan, Budget Justification and Performance Evaluation** (REQUIRED + 7 marks)

The project plan should list the tasks that have been completed. For each of the tasks list the names of the individual(s) who worked along with the time they worked on the task. List the tasks that have been mostly done in groups and say the % contributions for each group member (in number of hours). You must use tabular format for proving the details of this section as follows:

(Use *NSID/LastName* format when you are mentioning a group member)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **List of Tasks** | | | **Completed by whom and % Contributions if done in group**  **(provide in number of hours)** | **Comments** |
|  |  | |  |  |
|  | **Abstract** | |  |  |
|  |  | |  |  |
| **1.** | **Introduction** | |  |  |
|  |  | |  |  |
| **2.** | **Requirements and Early Design** | |  |  |
|  |  |  |  |  |
|  | **2.1** | **Summary Use Cases** |  |  |
|  | **2.2** | **Fully-dressed Use Cases** |  |  |
|  | **2.3** | **Use Case Diagram** |  |  |
|  | **2.4** | **Domain Model** |  |  |
|  | **2.5** | **Glossary** |  |  |
|  | **2.6** | **Supplementary Specification** |  |  |
|  | **2.7** | **System Sequence Diagrams** |  |  |
|  | **2.8** | **Operation Contracts** |  |  |
|  | **2.9** | **Obtaining User Feedback** |  |  |
|  |  |  |  |  |
| **3.** | **Updated Design and Unit Testing** | |  |  |
|  |  |  |  |  |
|  | **3.1** | **System Operations** |  |  |
|  | **3.2** | **Sequence or Communication Diagrams with GRASP Patterns** |  |  |
|  | **3.3** | **Class Diagram** |  |  |
|  | **3.4** | **Unit Testing** |  |  |
|  |  |  |  |  |
| **4.** | **Reengineering and Mutation Testing** | |  |  |
|  |  |  |  |  |
|  | **4.1** | **Code Smells** |  |  |
|  | **4.2** | **Refactoring** |  |  |
|  | **4.3** | **Gang of Four Design Patterns** |  |  |
|  |  |  |  |  |
| **5.** | **Complete Implementation and Product Delivery** | |  |  |
|  | | | | |
|  | **5.1** | **Naming Conventions** |  |  |
|  | **5.2** | **Commenting** |  |  |
|  | **5.3** | **Pretty-printing of the source** |  |  |
|  | **5.4** | **Usability Engineering** |  |  |
|  | **5.5** | **Complete Implementation** |  |  |
|  | **5.6** | **User Manual** |  |  |
|  |  |  |  |  |
| **6.** | **Project plan, Budget Justification and Performance Evaluation** | |  |  |
|  |  |  |  |  |
| **7.** | **Conclusion** | |  |  |
|  |  |  |  |  |
|  | **Acknowledgements** | |  |  |
|  | **References** | |  |  |
| **Total % Contributions (in hours) of the group members** | | | **NSID1/LastName1:** |  |
| **NSID2/LastName2:** |
| **NSID3/LastName3:** |
| **NSID4/LastName4:** |
| **Total (in hours)** | | |  |  |

You must also provide another table by showing who has done what (i.e., who was responsible for which task) and how much help was taken from whom for the major tasks of your project as follows. **Also list all the tasks (e.g., implementation related) for which your group or you spent a significant amount of time and the ones you would like to draw marker’s attention.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Group Member** | **Tasks Responsible For** | **% Contributions if not done alone, and then say who helped and how much** | **Comments** |
| **NSID1/LastName1** | Task1 | Spent 5 hours.  Received 1 hour of help from NSID2/LastName2 in connecting to the database **(this is just an example)** |  |
| Task2 |  |
| Task3 |  |
| Task4 |  |
| **Total for LastName1** | | X hours | Definitely one of the best among the group members |
|  |  |  |  |
| **NSID2/LastName2** | Task7 | Spent 4 hours.  Received 0.5 hour of help from NSID1/LastName1 in connecting to the database **(this is just an example)** |  |
| Task8 |  |
| Task9 |  |
| Task10 |  |
| **Total for LastName2** | | Y hours |  |
|  |  |  |  |
| **NSID3/LastName3** | Task11 | Spent 4 hours.  Received 0.5 hour of help from NSID1/LastName1 in connecting to the database **(this is just an example)** |  |
| Task12 |  |
| Task13 |  |
| Task14 |  |
| **Total for LastName3** | | Z hours |  |
|  |  |  |  |
| **NSID4/LastName4** | Task15 | Spent 4 hours.  Received 0.5 hour of help from NSID3/LastName3 in connecting to the database **(this is just an example)** |  |
| Task16 |  |
| Task17 |  |
| Task18 |  |
| **Total for LastName4** | | Y2 hours |  |
| **Total hours** | |  |  |

Compare these actual data with the rough estimate provided in the *Introduction* and comment on the issues (e.g., whether the initial estimate was okay, over estimated and so on).

List the success or failure experiences that deserve (at least you feel so) the marker’s attention. Note that you might not be successful in all the cases given that you have had limited time and that the primary purpose of this course project is to develop a mid-size project by following all the software development/engineering principles. And thus, even if your implementation is bit weak this is okay. By telling us what you have tried but could not make it because of the challenges involved, you might increase the possibility of getting a higher grade even if you have failed to achieve that particular goal.

Note that there might be significant similarity between the above two tables, especially in terms of the tasks and number of hours spent. However, the purpose of the second table is to give you an opportunity to tell us in details about your contribution to the project, especially those that were challenging and that you did address those. The data provided here should be as accurate as possible and these data **might have an effect in your final grade.**

**7. Conclusion** (1 mark)

In this document, we have provided the details of what to submit for the final milestone of the CMPT 370 course project. We also suggested that the students must use the word version of this document as a Template for the submission. In this section, you must say again the executive summary of your project along with a short text for the future work.

**Acknowledgements** (1 mark)

In case you have taken any help from any third party (i.e., from outside of the group members), you must acknowledge them in this section for their contributions to your project. For example, my acknowledgement for this document is as follows:

The author would like to thank Khalid Billah not only for his hard efforts in conducting the tutorials for this course but also for helping the students in finding code clones from their projects and marking some of the projects. The author also thanks the other markers namely, Saidur Rahman and Farouq Al-Omari for their hard efforts in marking the projects. The author further thanks the guest speakers namely, Gord McCalla, Kevin Schneider, Tom Roberts, Lyle McRae and Nathan Heagy for their excellent talks in this course. Finally, the author greatly appreciates the efforts of the students of this course for their interests in this course and for working so hard for it.

**How to submit?** (REQUIRED + 0 marks)

* 1. Make a directory called FinalMilestone\_groupName.
  2. Make a pdf file for the documentation part, i.e., the completed version of this template in pdf format.
  3. Also copy the project source including the databases and other associated materials inside the directory created in (a) above. Make necessary subfolders if you wish.
  4. Make a tar/zip of the directory and submit to Moodle. Only one member is required to submit.

**You must follow these guidelines for submission, otherwise, a penalty of up to 2 marks will be assigned to your group.**

**References** (1 mark)

You must provide citations for the different things you use or refer to. For example, for this milestone, I have used the following things and I have also provided citations where I have used/referred them:

|  |  |
| --- | --- |
| [1] | The NiCad website: <http://www.txl.ca/forum/viewtopic.php?t=662> |
| [2] | The Clone detection literature website: <http://students.cis.uab.edu/tairasr/clones/literature/> |
| [3] | The JUnit website: <http://www.junit.org/> |
| [4] | The CCFinder website: <http://www.ccfinder.net/> |
| [5] | The Simian Website: <http://www.redhillconsulting.com.au/products/simian/> |
| [6] | Chanchal Roy and Jim Cordy’s Technical Report: <http://research.cs.queensu.ca/TechReports/Reports/2007-541.pdf> |
| [7] | The Eclipse: <http://www.eclipse.org/> |
| [8] | Refactoring in Eclipse: <http://www.cs.umanitoba.ca/~eclipse/13-Refactoring.pdf> |
| [9] | The Design Patterns and CodePro:  <http://www.cs.umanitoba.ca/~eclipse/14-DesignPatterns.pdf> |
| [10] | The Artistic Style webpage: <http://astyle.sourceforge.net/> |