***~~Company Management System~~***

|  |  |
| --- | --- |
| **Customer**  LT Igawa, USN  25 Company Officer  410-293-7525  [igawa@usna.edu](mailto:igawa@usna.edu) | **Technical Advisor**  Lori Delooze  Assistant Professor  410-293-6820  [delooze@usna.edu](mailto:delooze@usna.edu) |
| **Team Lead**  **1/C Michael Laws**  **650-336-5297**  [**m113666@usna.edu**](mailto:m113666@usna.edu) | **Graphical Design**  **1/C Ryan Rabe**  **347-770-7222**  [**m115412@usna.edu**](javascript:__top__.add_selection('Midn Ryan Dean Pierce Rabe <m115412@usna.edu>', true);) |
| **Site Functionality**  **1/C Dimitri Hatley**  **1281-825-7187**  [**m112826@usna.edu**](javascript:__top__.add_selection('Dmitri Hatley <m112826@usna.edu>', true);) | **Database**  **1/C Michael Harrison**  **1850-525-1134**  [**m112802@usna.edu**](javascript:__top__.add_selection('Midn Michael Lee Harrison <m112802@usna.edu>', true);) |

***~~CMS is a web interfaced, database back ended, modularly designed, system to manage the daily paperwork that a Company of Midshipmen deals with on a daily basis.~~***

c.      **Paper Copies of Slides**. Include a paper copy of all the slides used in your presentation, and screen shots of your GUI in action.  These are to be ready at the start of the period, and turned in to your instructor *prior* to beginning your oral presentation.

1.     To save paper, slides MAY be printed out as handouts with up to 6 slides to a page, so long as the paper copy result is readable to someone who is borderline for needing bi-focals.  If in doubt, print out just 2 slides (or even 1) per page and/or use a larger font on your slides to begin with.

2.     It is recommended that the milestone lead physically check off each of the required items to ensure that all are present and in good order well before the scheduled milestone delivery date**.**

d.     **Electronic Copy:** By the start of your capstone presentation, email your instructor and your customer a single zip file named “TeamName\_ProjectName\_course#\_sec#\_SemesterAYxx.piz” For example, if your team name is the CodeMaulers, your project was a Flight Scheduler Assistant, you are in IC400 section 2001 and it is the spring of academic year 1982-1983, you would name your zipfile: CodeMaulers\_FlightScheduler\_IC400\_sec2001\_SprAY83.   Note that you will need to alter the file extension from .zip to .piz so that the USNA email system will not strip the .zip file out as part of its malicious code routines. Your final delivery is not considered complete until your instructor has received the zip file. Make sure there is nothing else other than the below in the folder that you zip up, and that you have retained your own electronic copy (if desired). Also, name the folder that you zip up “TeamName\_ProjectName\_course#\_sec#\_SemesterAYxx” using the same conventions discussed above.  The folder you zip up and send must include:

1.     **Source Code:** Turn in an electronic copy of all source code you developed for the project. Note that you do **NOT** need to turn in a paper print out of all of your source code (see the After Action Review section discussed below). All of your final source code is to be well commented, and use descriptive naming conventions. For classes, capitalize the beginning letter of each word in the class name (ie., ClassName). For constants, capitalize all letters, and put underscores between interior words (ie., NUM\_AVIATORS). For objects/variables, capitalize only the first letter of interior words (ie., operationsOfficer). Note that opOff would NOT be considered a descriptive variable name.  All source code and text/data files are to be placed into a folder entitled “sourceCode” within your zipped folder.

**3.     Project Binders** (2 identical ones are required).

Prepare two professional looking binders complete with labeled and tabbed sections (labeled as described below) containing paper copies of the following items updated and complete for the final software system you are delivering. Deliver one of these binders to your instructor and the other to your customer prior to the start of your milestone delivery.  Note that, in the interest of saving a few trees, you do **NOT** need to turn in a paper print out of any of your source code except as indicated in the “After Action Review” section below.  Include electronic copies of all of the below in an appropriately named folder as described in Section 1.d above.

**d.      Workload Matrices** (there are just 2 required). Give your final, Cumulative Workload Graphics, presented two ways:

a.      Team-member-oriented.  Cumulative for this and all previous milestones.

b.     Task-oriented.  Cumulative for this and all previous milestones.

**e.      Gantt Chart.** Give your project planning Gantt chart clearly annotated to show the original plan as contrasted with what actually occurred to include which milestones have been completed or partially completed to date, and which milestones remain.

**a.     Project Planning Percentage.** Give an unreduced fraction showing the number of whole weeks during which you were exactly on schedule with your ORIGINAL Gantt Chart divided by the total number of whole weeks the project entailed.

**i.** Use this fraction as the basis for showing what percentage of your project went completely according to your Gantt Chart plan.

**ii.** Note that your grade will not be impacted by whether your project went exactly according to plan (a rare event in the world outside the Yard), but rather by whether you correctly identify how close your project came to the original plan.

**b.     Project Planning Analysis.**  For each milestone that did not go exactly according to your original Gantt Chart plan, discuss why things did not go as you had planned.

**f.       Functional Requirements Trace Table**.  Give your final, updated, Functional Requirements Trace Table. Include your customer’s initials next to all functional requirements that the customer certifies have been met to their satisfaction. At the top of your trace table, include an *unreduced* fraction showing the total number to date of Functional Requirements whose set of test cases all have the status of ‘Completed’ divided by the total number of Functional Requirements in your system.

**g.      Analysis/Design Artifacts.** Present the UML Use Case diagrams, UML detailed Class diagrams, and UML Sequence Diagrams (or other modeling artifacts in the case of hardware or non-object oriented software development) that show the union of all the modeling that has occurred with your system.

**j.       Publicity Artifact.**  Your publicity artifact.

**k.      Colleagues:** In the spirit of the Colleagues principle of the Software Engineering Code of Ethics, section 7.02, (Assist colleagues in their professional development) select and present one aspect of their software system that clearly goes above and beyond what was learned in this or prior classes.  This ‘aspect’ should focus on one of the things you found most challenging to implement in your system.  At a minimum, you must include paper printouts of each of the following for EACH Colleagues aspect you previously presented.  The information contained here must be sufficiently detailed that someone can fully understand how to implement the functionality on their system.  At a minimum, include paper copies of

a.      slides that identify what your functionality is and how it is used, with screen shots as appropriate,

b.     a slide with a UML sequence diagram of both the pre-defined and user-defined system components involved in the functionality,

c.      a slide with the source code showing exactly how the functionality is implemented

d.     include screen shots of a demo run of your system showcasing your functionality.

**l.       User’s Manual**: Give the final, updated, user’s manual for your system.

**m.    After Action Review**: An after action review is a structured review process for analyzing *what* happened, *why* it happened, and *how* it can be done better by you or by others in the future. Your grade will not be impacted by whether a poor situation (as described below) exists in your system, but rather by whether you correctly identify and analyze the impact of the poor situation.  For your After Action Review, turn in **and present** your results for the following:

1.     **Self-Analysis of your software development process**.

a.      A discussion of what went wrong during the development of your system and why.

***~~Mistakes Made:~~***

***~~Did not fully solidify requirements before coding~~***

***~~Did not fully learn the potential or quirks of Django before coding~~***

***~~Suggestions for the Future:~~***

***~~Spend time fully learning the languages and frameworks and their internal systems before using them~~***

***~~Complexity Management~~***

***~~Many MIDN data points to keep track of and reference~~***

***~~SOLUTION: Use the Django framework to aid in design and development~~***

***~~2. Team coordination and cohesion~~***

***~~Dynamic projects require active communication~~***

***~~SOLUTION: External server based solutions for code storage and execution~~***

***~~3. Learning a new framework and language~~***

***~~We had to learn the language python and the framework Django~~***

***~~SOLUTION: Time and effort.~~***

b.     What you would do differently if you had to start this project over again from scratch, and how you would accomplish these differences.

***~~Simple problems may have complicated solutions.~~***

***~~Good team communication is as important as good code.~~***

***~~A framework to aid development may or may not be a good idea depending on the application.~~***

2.     **Ethics.** Consider the entire course-long development of your project within the context of Principle 1: Public as given in the article "[Software Engineering Code of Ethics is Approved](http://usna.edu/Users/cs/needham/courses/ic480/sprAY11/SE_Code_Of_Ethics.pdf)" *Communications of the ACM*, October 1999/Vol 42. No. 10. pp 102-107.

a.      **Title & Description**: For YOUR CHOICE of 2 sub-principles (1.01 ... 1.08) under Principle 1: Public, give a one sentence title & description of the respective sub-principles and why you selected each sub-principle.

b.     **Sub-Principal Analysis:** For each sub-principle chosen in part a, give a one page analysis (single-spaced, 1 inch margins all around, 10 pt Times-New Roman font) of what went right, what went wrong, and what parts of your system and/or design/development experience could/should be improved from the perspective of each particular sub-principle.  These must be thoughtful analyses of your system from the various ethical perspectives, not just a light “everything is pretty much fine” collection of spell-checked sentences.

3.     **Known Bugs.** Include a discussion of any known bugs, places where you know (or suspect) the system is unstable, and address any major mistakes or wrong turns you made during the system’s development.

1. CDO options: Biggest bug in our project is the fact that CDO options are not restricted to the current CDO, but rather are accessible by all 1/C MIDN. Reason for the bug is that the CDO check references an non-existing sub-system of the project. In the original design we were intending to implement an internal watch bill structure that would restrict certain options to current watch standers; however, the system was never actually put in place.  
  
2. Data validation: not a bug per say, but more of a missing feature. As of now we have a very limited data validation at the server level, therefore an incorrectly submitted entry has a chance to corrupt database records and possibly propagate through other queries, especially overarching ones like 0800 report.

4.     **Lowest Cohesion.** Identify the module that exhibits the worst level of cohesion in your system. Give:

There is no single example of high or low cohesion or coupling in our project. All modules follow exactly the same hierarchical relation to each other, and share both coupling and cohesion characteristics amongst each other.

5.     **Tightest Coupling.** Identify the modules that exhibit the worst level of coupling in your system. Give:

Our project is very "flat" by design. We have an overarching "mid" module that references all other modules; in terms of reliance, all other modules depend on "mid" module for proper functionality.

6.     **Highest Complexity**. Identify the module that exhibits the highest McCabe’s Metric M-value in your system. Give:

a.      the source code for the module (using a font size large enough to be read),

b.     the graph, resulting from your statement to graph conversions, that clearly shows the number of edges and number of nodes involved in the source code,

c.      the M value computation expression (and result),

d.     the steps (if any) you took to reduce the complexity of the module (include the before and after graphs and M value expressions and results if you were able to reduce the complexity of the module), and,

e.      your rationale for allowing module to remain in your system at its current complexity level.