EVERY REPORT MUST INCLUDE THE FOLLOWING:

- Title page with team members, team leader, phase and date
- Page 2 should be empty except for one line in the middle saying: **[This page is intentionally left blank]**
- Table of contents with page numbers
- Pages should be numbered
- Include meeting minutes (scan if hand-written) approved by all members – rotate minute taking among members
- Include detailed task decomposition: who did what

****

Phase I

Skills & Topics needed: *use case diagrams*, *use case descriptions*, *class diagrams*, *communication diagrams* & *DIA diagramming editor*

****

1. **A complete UML use case diagram** (you will need to drive it in steps until it is complete, but turn in only the complete version) with notes and clarifications (as needed). This must be done using a UML diagramming tool. **PS: Unlike iCoot, CMC is not about automating an existing manual process; thus, we can ignore the "business perspective" and delve directly into the "developer perspective"**

2. A **complete description for every use case** in your diagram which should include the following:
   - Number and title
   - Relationships to other use cases: includes /included by/extends/extended by/specializes/specialized by
   - Whether use case is abstract
   - Actors involved
   - Preconditions
   - Main Scenario (normal path)
   - Post conditions
   - Alternative Scenarios (abnormal paths)
   - Non-functional requirements

3. A **complete UML class diagram** showing
   - Classes: including attributes (with visibilities and data types) as well as methods (with visibilities, parameters (also need data types), and return data types). **PS: use *italics* for abstract classes, `<<interface>>` for interfaces, and <u>underlining</u> for static methods or attributes**
   - Relationships between classes:
     - ○ Inheritance: *Specialization* and *Realization*
     - ○ *Association*: uni- and bi-directional along with multiplicities
     - ○ *Aggregation* and *Composition*
     - ○ *Dependency*

4. A **complete communication diagram for each path in every use case**
   - Your communication diagrams should be used to enhance your class diagram
   - The submitted versions of both must be consistent

****

Phase II

Skills & Topics needed: *Netbeans IDE*, *CVS in Netbeans*, *debugging in Netbeans*, *Java packages* & *javadoc*

****

1. An **updated Use case diagram**
   - Based on feedback from Phase I

2. An **updated UML class diagram**
   - Based on feedback from Phase I
   - Incorporate the provided database library (now provided to you on the N: drive)
   - Recommendations:
     - entity classes: `User`, `Admin`, `Account` (super class for `User` & `Admin`) and `University`
     - controller classes: `UserFuncController`, `AdminFuncController`, `AccountController`, `UniversityController`, `SearchController`, & `DBController`

3. **Java implementation** of your class diagram
   - Must be consistent with class diagram
   - Must use packages
   - Must be fully javadoc documented
     - BEFORE every Class
       - Description
       - *@author(s)*
       - *@version*
     - BEFORE every Class Attribute
       - Description
     - BEFORE every Class Method/Constructor
       - *@param (one for each parameter)*
       - *@return (if applicable)*
       - *@throws(one for each exception)*

4. A **simple driver class** which allows you to try out each of the functionalities (from your updated use case diagram)

5. Your turned in report must include
   - The requirements common to all phases (listed at the very top)
   - The updated use case diagram
   - The updated class diagram

6. DO NOT TURN IN A HARD COPY THIS TIME. Place your report in a folder with your team name. Also include in your folder a copy of the code developed for this phase. In addition, include a READ ME file telling me how to run your driver class and make it do each of the functionalities. Turn in the folder to our handins folder:
   - `/usr/people/handins/CS230` (on Linux)
   - `N:\Handins\CS230` (on Windows)

****

Phase III

Skills & Topics needed: *Basic HTML, HTML form processing, Komposer WYSIWYG editor, JUnit testing in Netbeans, white-box testing, black-box testing & functional testing*

****

1. Create a complete **test harness** for your classes from Phase II: You should create one test class for every Java class created in phase II. You also need to create a single test suite to run all tests. Your test classes must reside in a separate tests folder under your project's root folder. Note that every test class must reside in a package structure similar to the corresponding project class it is testing.

   Alternate your test methods appropriately among
   - *white-box testing* for complicated methods with multiple branches: In your report, include flowcharts and tables describing the path, input and output for each test case
   - *black-box testing* for methods with multiple inputs: In your report, include descriptions of your equivalence classes (per parameter) and the resulting test case to cover them as well as boundary cases
   - *basic testing* for other simple methods

2. Create a **complete functional testing class** that contains one test method for each of the functionalities (from your updated use case diagram) in CMC --- the objective here is to show the customer that all of the functionalities in CMC can be accomplished using your Java classes created so far. If you believe that you've already covered this in your testing of classes, then describe (in your report), for each use case, where you've done the testing and how.

3. A complete fully linked **HTML (static) website** for CMC.com

4. Your turned in report must include
- The name and objective of every Java class in your system
- Under every class
  - The header and objective of every method in the class
  - Type of testing used: *white-box*, *black-box* or *basic*
  - For methods tested using
    - *white-box testing*: include a flowchart and a table describing the path, input and output for each test case
    - *black-box testing*: include a description of the equivalence classes (per parameter) and the resulting test cases to cover them as well as boundary cases (input and output combinations)
    - *basic testing*: the input and output combination(s) used
- A description ofyour functional testing

5. DO NOT TURN IN A HARD COPY THIS TIME.
- Place your report in a folder with your team name.
- include in your folder a copy of your updated code along with test classes & suite developed for this phase (please use proper naming conventions)
- Include a READ ME file telling me how to run your test suite and functional testing class
- Turn in the folder to our handins folder:
  - `/usr/people/handins/CS230` (on Linux)
  - `N:\Handins\CS230` (on Windows)

****

Phase IV

Skills & Topics needed: *Creating Web projects in Netbeans*, *JSP* and *interfacing between HTML and JSP*

****

1. The major requirement for this phase is to produce a working system!

2. Your report should include, in addition to the standard sections common to all phases, the following sections:
- **<u>REFLECTION</u>**: Reflect back on your experience in this project and address the following points

  - Given the set of functionalities that you started with initially,
    - which system functionalities have you completed as required?
    - Which have you modified and completed (and briefly state why you modified them)?
    - Which have you omitted (and briefly state why you dropped them)?
    - Have you added and completed any new system functionalities (and briefly state why you added them)?

  - Describe your experience working on the different phases of this project.
    - Which phase(s) did you enjoy the most? Why?
    - Which phases weren't very successful for your team? Why?
    - What have you liked in particular about the project?
    - If given the opportunity to redo a similar project from scratch, how would your team improve your project experience?

- **ARGUMENT**: Tell me why you deserve a *B* or more. A working project that satisfies ALL systems requirements, and includes a complete report (as described above) will earn your team a grade in the B range. An *A* requires at least TWO of the following:
  - Clear separation between system logic and presentation (i.e. logic should go in java classes and presentation in HTML) resulting in minimal JSP code within HTML
  - A sophisticated GUI
  - Complete error handling (using JSP error pages) & security checks(e.g., a user shouldn't be able to visit some pages without being logged in)

3. DO NOT TURN IN A HARD COPY THIS TIME.

- Place your report in a folder with your team name.
- include in your folder a copy of your final Netbeans project
- Turn in the folder to our handins folder:
  - `/usr/people/handins/CS230` (on Linux)
  - `N:\Handins\CS230` (on Windows)