This is the third of five assignments that you will complete over the course of the semester:

1: Requirements Draft (10% of homework grade)
2: Final Requirements and Requirement-Based Tests (25%)
3: Design Draft (15%)
4: Final Design and Implementation (25%)
5: Testing (25%)

Each assignment is graded over a series of categories. You will be judged on a scale of 1-4 for each criterion, where a 1 corresponds to a 60%, a 2 corresponds to 75%, a 3 corresponds to 90%, and a 4 corresponds to 100%. If there is no work for a criterion or it is clear that even a minimal amount of effort was not put in, you will receive a 0% for that section of the assignment.

The following is a tentative idea of what we are looking for in Assignment 3. This may change before final grading, but gives criteria to aim for with your submission. A "4" in a category requires all requested elements to be present. Missing elements will result in a lower grade.

Organization (10%):

- Have a good organization including a logical layout.
- All sections present.
- Design formatted to be easily understood.
- Uses good grammar, and has a single voice.
- No irrelevant data.

Architecture (20%):

- Introduction, architecture, interfaces, and data stores present.
- Material provides proper context and background on the group's version of GRADS.
- Proper differentiation between internal data stores (any persistent storage used internally by GRADS) and external databases. □
- Proper use of interfaces when discussing architecture (i.e., "interface" between SABE and databases, not "Java interfaces").

Structural Design (40%):

- Overall design
    - Extensible OO design for building Progress Summary and calculating the result of the graduation rules □
    - High cohesion and low coupling.□
    - No driver is included (main() method).
    - All interfacing with GRADS is through the interface. Access is controlled.
    - Top-level implementation of GRADSIntf present. □
    - Customized Exceptions
- □Class Diagram
    - Properly formed UML □

    - Databases should not be present in class diagram □
- Justification and Explanation
    - VERY IMPORTANT to justify and explain your design. Must show that different options were considered and why/how group arrived at final design. Must demonstrate understanding of OO principles. □
    - Automatic maximum of 2 on this section if no justification present. □
- Class Descriptions
    - Level of detail is sufficient. Is this implementable by another team?

Dynamic Design (30%):

- Sequence Diagrams□
    - “Generate Progress Summary” scenario must be present.
    - Properly formed UML □
    - Instances, not static classes. □
    - Life lines and activation boxes present □
    - Actor present □
    - Calls labeled □
    - Database calls handled correctly. □
- Diagram description present and understandable.