HW3.5 (team): Requirements Analysis

This homework assignment supersedes HW2 and HW3 (i.e., your grade on this HW will stand as your grade on both HW2 and HW3). In short, your main tasks will be to:

- Rewrite your functional requirements as brief use cases.
- Add a glossary.
- Revise your artifacts to meet the grading criteria (including a new document structure) given next.

Grading Criteria

Your requirements document will be graded based on

- GC1: Qualities of good requirements (with particular emphasis on *ed qualities):
  - Correct*
  - Consistent*
  - Unambiguous*
  - Complete*
  - Relevant
  - Testable
  - Traceable
- GC2: Use of *essential style* that includes concrete UI descriptions
- GC3: Use of *black-box style*
- GC4: Use of *actor* and *actor-goal perspective*
- GC5: Modularity (e.g., no redundant events in use cases)
- GC6: Professional appearance of document
  - Uniform style/formatting
  - Adherence to document structure (see next section)

Document Structure

1. Front Matter and Vision Statement

Include the project name and team members followed by the original vision statement. (Feel free to reformat it to match the style of your document and to fix typos. Make sure to credit the author of the vision statement.)

2. Use-Case Model

Brief UCs. Include a *complete set* of brief use cases (UCs). Each brief UC should describe a success scenario. You do not need to include alternative scenarios (such as those related to data validation and error handling).

Fully dressed UCs. Include 3 fully dressed UCs. Each should have:

- Title
- Actor(s)
- Preconditions
- Postconditions
- Basic flow (main success scenario)
- Alternative flows—a complete set!

For either type of UC, embed cross-references to other UCs (by name) and user interfaces (by name) as appropriate.

3. System Sequence Diagrams

Include 3 system sequence diagrams (SSDs), one for each fully dressed UC. Each SSD should depict 1 scenario (i.e., don't include alternative scenarios).

4. Supplementary Specification

Non-functional requirements. Categorize your non-functional requirements by software quality attribute (reliability, efficiency [includes performance], integrity [includes security], usability, maintainability, testability, flexibility, portability, reusability, and interoperability). If you're having trouble coming up with non-functional requirements, try working top-down from the quality attributes.

Application-specific domain rules. Include, for example, business rules or rules from the application domain. Basically, these "rules" are global constraints that stem from the problem domain. For example, you might include FERPA (Family Educational Rights and Privacy Act) rules if you were building a system for managing student data.

5. Domain Model

Include a class diagram that models the major concepts and relationships in the problem domain. All classes should be *conceptual classes*; do not include *software classes*.

6. Dataflow Model

Include a dataflow diagram that uses the essential style—that is, include only domain entities, no software entities. Hint: there should be at least one human in your diagram.

7. UI Prototypes

List each major interface. Give each interface a name and include a picture. If necessary, include supporting text to explain what the features of the interface are and how a user interacts with the interface. You may use your paper prototype pictures if they are appropriate. Use the interface names to refer to the interfaces in your use cases.

8. Glossary

Include *all terminology* used in the previous sections. Among other things, such terms should include the different user types (e.g., *User*, *Registered User*, *Student*, *Teacher*) and other domain entities (e.g., *Grade Form*, *Profile Information*).

9. Attribution Lists

In a structured presentation, list:

- Each team member's name
- The artifacts that each team member worked on
- What the team member's contribution was to each artifact

Here's an example (although, feel free to be more specific about the contribution):

Team member	**Artifact**	**Contribution**
Jim Foo	Brief UCs: *FindJob*, *ReadLog*	Original author
	SSD: *FindJob*	Made minor revisions
	Dataflow diagram	Made major revisions
	Interface: *Welcome/login*	Original co-author

Additional Notes

- Submit *1 and only 1* PDF file that contains your entire submission.
- Page break rules: Insert the following page breaks *and no others* (normal page wrapping is allowed, though).
    - Start each section on a new page.
    - Start each fully dressed UC on a new page.
- I recommend designating one or more team member as an editor whose job it is to make sure that the document is free of formatting inconsistencies and typos. Editors must have excellent attention to detail.
- You will be discarding your requirements specification. I value creating genuinely useful artifacts over dogmatically generating documents. Some approaches see value in the creation of both definitions and specifications; however, in the context of this class, the specifications seem to me redundant, adding little value. Thus, I'm having you drop them. The UCs that will replace your original requirements definitions should, like the definitions, be written from the user perspective.