You will apply the following outline for each component in a project. Some components may not have all design document sections. An application component, for example, would not have an Interface Functions section. Explanations are sometimes included parenthetically. All descriptions should be in clear, very concise English (do not resort to pseudo code).

Please use the fields in bold to organize your documents. Component Name will obviously be the name of the component you are designing, but the other bold fields should be used verbatim.

While design documents should be complete, brevity is very important.

**Component Name**

**Data Structures**
For each data structure:

- **Fields** List the fields that make up the state of the data structure using descriptive names. Describe the purpose and type of each field (parent - system pointer back to object parent) or (color - enum which tells what color the object is).

- **Invariants** List conditional invariants (if any of the following are true, all must be true: head pointer is NULL, tail pointer is NULL, length is 0). Also list unconditional invariants (parent pointer should never be NULL).

**Interface Functions**
For each interface function:

- **Preconditions** List the conditions which must be true of the system and of the input for this function to run successfully.

- **Data** List shared data objects used, noting the data owner.

- **Algorithm** Description of the algorithm to be used, noting any used primitive or utility functions. (high level terse prose, no pseudo code!)

- **Postconditions** List the conditions which will be true of the system and of the output after this function runs.

**Utility Functions**
For each utility function:

- **Data** List shared data objects used, noting the data owner.

- **Behavior** Description of what the function does, as well as how it is accomplished (high level terse prose, no pseudo code!)

**Testing Strategy**
For the component as a whole:

- **High Level Plan** Describe the general categories of bugs this component will suffer from. Describe the most efficient plan for testing each of the categories.

- **Corner Cases** List the specific cases that are very likely to cause problems in the implementation as you are designing it.