<table>
<thead>
<tr>
<th>Area</th>
<th>Subdivision</th>
<th>Item</th>
<th>Criteria</th>
</tr>
</thead>
</table>
| Programming  | Basic coding    | Code                  | Informative names that adhere to language conventions  
Consistent and attractive layout (spacing, indentation, etc.)  
Judicious use of comments, especially for failure paths  
Reasonable function and module length  
Minimal duplication of code  
No magic numbers; constants defined in appropriate scopes  
Idiomatic use of language, including avoidance of bad parts  
Appropriate and skillful use of advanced language features  
Appropriate use of known algorithms and data structures  
Appropriate use of platform and third-party libraries  
Citations for borrowed code and ideas |
|              | Modularity      | Code                  | Code sensibly divided into modules and files  
Namespace, structured and coherent  
Separation of concerns (especially presentation/content)  
Clean and simple module interfaces  
Functions don’t have too many arguments  
Data types immutable when appropriate  
No exposure of representations  
Abstraction barriers not violated  
Inter-module dependences controlled  
Design decisions localized as much as possible  
Specifications  
Sucinct but informative specifications for public interfaces  
Cohesive specification: avoidance of disjunctive behaviors  
Preconditions given, especially on session state |
|              | Verification    | Runtime assertions    | Runtime assertions to check non-trivial expectations  
Representation invariants for abstract types  
Schema invariants declared, maintained (& checked if appropriate)  
Unit tests  
Repeatable suite of tests for key methods of service interfaces |
|              | Security        | Code                  | Appropriate use of security mitigations (eg, sanitization)  
Access control mechanisms implemented, as relevant  
Safe defaults used |
| Design       | Overview        | Purpose and goals     | Brief description of system to be built  
Key goals and purpose (what problem does it solve?)  
Motivation for development (eg, deficiencies of existing solutions)  
Context diagram  
Establishes boundary of system  
Interactions between system and external entities |
|              | Concepts        | Key concepts          | Brief explanation of key enabling concepts |
|              |                 | Data model            | Data model of application state  
Schema representation details excluded  
Syntactically valid diagram with consistent naming & layout  
Generalization used appropriately  
Names of sets and relations well chosen  
Designations in accompanying text of non-obvious elements |
|              | Security concerns|                       | Summary of key security requirements and how addressed  
How standard attacks are mitigated  
Threat model: assumptions about attackers  
User interface  
Wireframes for application  
Flow between pages indicated, with named actions  
Errors accounted for |
|              | Challenges      | Design challenges     | List of problems to resolve in concepts, behaviors or implementation  
For each problem: options available, evaluation, which chosen  
Notes on code design: schema design choices, abstractions |
| Team Work    | Plan            | Stakeholders          | List of stakeholders and their roles |
|              |                 | Resources             | List of computational, cost and time constraints |
|              |                 | Tasks                 | List of tasks, expected effort, allocation to team members  
Calendar of intermediate and final milestones for tasks  
Risks  
Enumeration of expected risks and their mitigations |
|              |                 | Minimum viable product| Identification of minimum viable product for first release  
Subset of features to be included  
Issues postponed (eg, security mitigations, user interface elements)  
Provides real value to users  
Provides opportunity for feedback  
On path to full product |
|              | Team contract   | Team contract         | Expected level of achievement and effort for each team member  
Personal goals for each team member  
Frequency, length and location of team meetings  
How quality of work will be maintained  
How tasks will be assigned, and what to do if deadlines are missed  
How decisions will be made and disagreements resolved |
|              | Meetings        | Agenda                | One agenda for each meeting  
Agenda prepared in advance of meeting  
Progress report  
One report for each meeting, prepared in advance  
Summarizes progress since previous meeting  
Identifies achieved and missed milestones  
Identifies difficulties encountered  
Identifies changes to problem analysis or design |
|              |                 | Meeting minutes       | Summary of discussions and advice from mentor  
Summary of new decisions  
Changes to plans or estimates  
Should be written during meeting |
|              | Reflection      | Peer review           | Constructive but candid evaluations of team mate performance |
|              |                 | Evaluation            | Evaluation of project from team planning perspective |
|              |                 | Lessons learned       | Summary of key lessons learned |