Area | Subdivision | Item | Criteria
--- | --- | --- | ---
Grading directions | Highlights | Points to best parts of project | Use links that include line numbers when citing code
 | Justifications | Brief explanation of why each task is good; what it achieves | Use links that include line numbers when citing code
Help wanted (optional) | Points to problematic parts | Requests for advice | Brief explanation of what you'd like advice about
Programming | Basic coding | Code | Informational names that adhere to language conventions
 | | | Consistent and attractive layout (spacing, indentation, etc.)
 | | | Javadoc use of comments, especially for failure paths
 | | | Reasonable function and module length
 | | | Minimal duplication of code
 | | | No magic numbers; constants defined in appropriate scope
 | | | Idiomatic use of language, including avoidance of bad paths
 | | | Appropriate and skillful use of advanced language features
 | | | Appropriate use of这辈子 algorithms and data structures
 | | | Appropriate use of platform and third-party libraries
 | | | Citations for borrowed code and ideas
 | Modularity | Code | Code nicely divided into modules and files
 | | | Name space, structured and coherent
 | | | Separation of concerns (eg, presentation/content)
 | | | Model classes encapsulate representation and functions over data
 | | | 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-modal dependencies controlled
 | | | Design decisions localized as much as possible
 | Specifications | Runtime assertions | Cyclic but informative specifications for public interfaces
 | | | Cohesive specification; avoidance of algorithmic behaviors
 | | | Separation of results returned from side effects
 | | | Declerative: describes what the behavior is, not how it's implemented
 | | | Error and exceptional cases described, separately from normal cases
 | | | Prescriptions given; eg on session state
 | Verification | Unit tests | Repeatable suite of tests for key methods of service interfaces
 | | | Appropriate use of security mitigations (eg, sanitization)
 | | | Access control mechanisms implemented, as relevant
 | | | Safe defaults used
 | Design | Overview | Motivation | Brief description of system to be built
 | | | Key purposes (what problems does it solve? why should it exist?)
 | | | Each purpose summarized in a short sentence and then explained
 | | | Deficiencies of existing solutions (if relevant)
 | Context diagram | Design model | Concepts | List of key concepts with brief definitions
 | | | Concepts have short and memorable names
 | | | Concepts capture central design ideas, not routine notions
 | | | In order of significance, most significant concept first
 | | | For each concept, says which purpose motivates it
 | | | No repetition of information already in data model (eg, multiplicity)
 | | | Longer explanations of novel or subtle concepts
 | | | Data model | Data model of application state
 | | | Schema representation details included
 | | | Syntactically valid diagram with consistent naming & layout
 | | | Generalization used appropriately
 | | | Names of role and relations usually chosen
 | | | Correct use of multiplicity and immediability markings
 | | | Designations in composing test of non-obvious elements
 | | | Avoidance of anti-patterns (eg, boolean flags instead of sublists)
 | | | Data design | Transformed data model corresponding to representation
 | | | Continues indicating allocation of relations to tables or collections
 | | | Behavior | 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
 | | | How between pages indicated, with named actions
 | | | Error handling shown, for validation errors and anticipated failures
 | Challenges | Design challenges | List of problems to resolve in concepts, behaviors or implementation
 | | | For each problem: options available, evaluation, which chosen
 | | | Data design choices and their justifications
 | 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
 | | | Risk | Enumeration of expected risks and their mitigations
 | | | Minimum viable product | Identification of minimum viable product for first release
 | | | Subset of concepts to be included
 | | | Issues proposed (eg, security mitigations, user interface elements)
 | | | Provides real value to users
 | | | Provides opportunity for feedback
 | | | On path to final 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
 | | | Progress report | One report for each meeting, prepared in advance
 | | | Summary | Summarizes progress since previous meeting
 | | | Meeting minutes | Summary of discussions and advice from mentor
 | | | Summary of new decisions
 | | | Changes to plan or milestones
 | | | Reflection | Peer review | Constructive but candid evaluations of teammate performance
 | | | Evaluation | Evaluation of project from team planning perspective
 | | | Lessons learned | Summary of key lessons learned