Deliverable	*Grade*		*Description*	*Grading questions*
Models	15%	Object models	Object models describing the problem domain, the abstract state of the application, and any significant implementation structures (such as database schemas).	Were object models constructed for the right aspects of the project?
				Is each object model legible and syntactically valid?
				Does each object model have appropriate labels for all nodes and arcs?
				Is the choice of names tasteful and helpful?
				Are any non-obvious relation or set names defined in a glossary?
				Are the models at an appropriate level of abstraction, and free of clutter from irrelevant details?
				Are the abstract models free of implementation details?
				Are any fundamental aspects of the problem or design missing?
				Are the modeled aspects represented correctly?
				Do the models make good use of generalization and specialization?
				Do the models correspond to the design notes, specification and code?
				For related models (such as abstract state and database schemas), is the relationship or transformation explained?
		State machines	State machines describing the problem domain, the behavior of the application, and any significant implementation machines (such as protocols for initialization or communication).	Were state machines constructed for the right aspects of the project?
				Is each state machine legible and syntactically valid?
				Does each state machine have appropriate labels for all nodes and arcs?
				Is the choice of names tasteful and helpful?
				Are any non-obvious events defined in a glossary?
				Do arc labels correspond to events and nodes to states?
				Are the models at an appropriate level of abstraction, and free of clutter from irrelevant details?
				Are any fundamental aspects of the problem or design missing?
				Are the modeled aspects represented correctly?
				Do the models make good use of hierarchical features of Statechart notation?
				Do the models correspond to the design notes, specification and code?
Design notes	20%	Key challenges	Overview of the main design challenges presented by the problem to be solved.	Is the overview simple, clear and well explained?
				Are all key challenges included?
				Are lower level and less important issues appropriately omitted?
		Issues arising	A succinct record of issues that arose during the design process, and how each was resolved. When a tradeoff is made, a brief rationale for the decision.	Is the list clear and well organized?
				Are the issues explained in a simple, precise and intelligible way?
				Are references to the models made when appropriate?
				Are both sides of a tradeoff articulated?
				Is the rationale for a tradeoff plausible and adequately explained?
		Critique	A brief evaluation, having completed the entire project, of what worked well in the design and what didn't, and what you learned from the experience.	Is the evaluation substantive and evidence of careful analysis, or pro forma?
				Are the lessons learned compelling and grounded in the project?
				Does the evaluation recognize major flaws of the project, if any?
Specification	15%	Overview	A brief overview of what the application does.	Is the overview clear and to the point?
		Key features	A list of key features, as might appear in an advertisement.	Are the features explained in a simple, direct and precise manner?
				Is the set of features appropriate for the size of the project?
				Do the features form a coherent set?
		User manual	A succinct user manual describing how the application is used. Should include any required setup, command line arguments, platform requirements, and instructions for issuing particular commands or interacting with particular web pages. If the user interface is sufficiently clear, detailed instructions are not needed.	Is the manual well organized?
				Are all relevant areas addressed?
				Are the directions clear and easy to follow?
				Does the manual use the conceptual terminology established in the models?
Implementation	40%	Code	The code itself, liberally but judiciously commented. Should include brief specifications for each method or function, and representation invariants for abstract data types.	Are the specified features implemented?
				Does the application run without crashes?
				Do features behave as expected?
				Is the user interface clear and easy to navigate?
				Are appropriate mitigations used to guard against security attacks?
				Is the code well organized into directories and files?
				Is the code appropriately indented and spaced?
				Are classes and methods/functions of appropriate length?
				Are names of files, classes, functions, variables, etc well chosen?
				Is the level of commenting appropriate?
				Are specifications and invariants present and sufficient?
				Is the programming language used appropriately?
				Is the code written in a clear and well structured way?
				Does the code make good use of runtime assertions?
				Are abstract datatypes used when appropriate?
				Are immutable types used when appropriate?
				Are magic numbers and hard coded constants avoided?
				Are program elements scoped appropriately?
				Does the code maintain separation of concerns?
				Are idioms taught in the course exploited when appropriate?
		Module dependency diagram	A diagram showing the modules in the implementation and how they depend on one another. An indication of which design patterns were used, if any.	Does the MDD correspond to the actual code?
				Does the MDD show decouplings and separation of concerns when appropriate?
				Are concerns separated appropriately at the module level?
		Code notes	A brief summary of any issues arising in the implementation and how they were resolved.	Are the issues explained in a simple, precise and intelligible way?
				Were the resolutions reasonable?
Testing	10%	Test plan	An explanation of how the application was tested (that is, how the cases were selected, executed and evaluated).	Is the explanation of the test plan sufficient, but not too verbose?
		Test cases	An annotated list of test cases; these do not need to be repeated if they are given in the code itself.	Are the test cases well structured?
				Are the test cases annotated appropriately?
		Rationale and conclusions	A brief rationale for the level of testing and choice of cases, and a conclusion based on the results of the testing of how much confidence you have in the dependability of the final product.	Is the rationale clearly and simply explained?
				Is the rationale compelling?
				Are there sufficient test cases, and adequate coverage?
				Are the conclusions warranted given the evidence?