		Level	Not Passable	Baseline	Milestone	Meets Expectations	Exceeds Expectations
		Explanation	*Does not show evidence of a working understanding of topic*	*Shows a basic understanding and implementation of the topic*	*Shows an understanding of the topic but actual implementation quality or completeness is lacking*	*End-of-course expectation - Shows a full understanding and solid implementation of the topic*	*Excellent understanding of topic and an implementation that goes above and beyond what is covered in class*
		Score	1	2	3	4	5
1	**Project & Development Process**						
1.1	**Complexity of Objective / Project Scope**		The project is poorly scoped, and is either far too simple or far too difficult based on the student's experience and what the class has covered.	The project is a bit too simple (or too complex), given what the class has covered so far.	Project is generally well-scoped, but the student may have under- or over-estimated the difficulty of certain key components.	Project is well-scoped from a production perspective; the student should be able to meet all requirements with the material that they've learned so far in class.	Project is well-scoped from an educational perspective; its requirements push *just* beyond what's been covered in class, giving the student an opportunity to explore and incorporate new material.
1.2	**Requirements Gathering**		Student does not define requirements before starting work.	Student can verbally articulate basic objectives, but produces no written work plan.	Written work plan defines basic objectives, but does not elaborate on specifications or audience requirements.	Written statement of work defines objectives, specifications, and audience requirements.	Written statement of work demonstrates extensive thought and attention to detail.
1.3	**Specification - User Stories**		User stories are vaguely defined or nonexistent.	Student can verbally articulate user stories, but produces no written documentation.	User stories are written down, but are incomplete or lack specificity.	User stories demonstrate a thoughtful understanding of the application's purpose, including roles, goals, and motivations.	User stories demonstrate extensive audience research.
1.4	**Specification - Object Model**		Object model is vaguely defined or nonexistent.	Student defines an object model, but it doesn't completely align with the user stories.	Object model aligns with the user stories, but may not be the best approach to solving the problem.	Object model aligns with the user stories, and is a reasonable approach to solving the problem.	Object model is natural and intuitive based on the user stories, adhering to best practices for solving the problem.
1.5	**Specification - Wireframes**		No wireframes produced.	Wireframes are vague or obviously incomplete.	Wireframes are present for key workflows, but capture few specifics.	Wireframes clearly illustrate the user interface and capture key workflow details.	Wireframes are impressively detailed, defining the interface and workflows in depth.
1.6	***Architecture***		*Planned architecture is poorly chosen, and the student's selection shows a lack of understanding about the nature and purpose of these tools.*	*Although it could do the job, the chosen architecture is not well suited for solving the problems the student is trying to address.*	*Though there may be better alternatives, the chosen architecture is an acceptable way to address the problem(s) at hand*	*Application architecture is chosen in line with standard industry practice.*	*Application architecture is chosen thoughtfully (student can defend their decisions), with an eye to the specific needs of this particular project.*
1.7	**Meeting Requirements**		Software fails to substantially meet any of the student's requirements.	Some of the student's requirements are at least partially implemented.	Software fulfills most of the student's requirements.	Software fulfills all of the student's requirements, and is deployed to a production server for public review.	Software exceeds the student's own requirements by implementing additional relevant and useful functionality.
1.8	**Testing**		No tests.	Feature tests are incomplete. Little to no unit testing is present.	Feature tests cover key workflows, but coverage of edge cases is lacking. Some unit testing is present.	Feature tests cover all key workflows, thoroughly validating the student's requirements. Unit tests cover most public methods/functions.	Feature and unit tests are comprehensive and leverage tools beyond the material covered in class. Code shows evidence of a test-driven workflow.
1.9	**Documentation**		No README content.	README introduces the project, but fails to define the software's installation and/or use.	README defines the software's purpose, installation, and usage.	In addition to (3), README is cleanly formatted and follows documentation conventions found in related software projects.	In addition to (4), the project release is communicated to software community members via blog posts and/or Twitter.
1.10							
2	**Code Review**						

	2.1	**Naming Conventions**	Names are ambiguous, duplicative, or do not accurately describe the concepts they are labeling.	Naming is unambiguous, with minimal abbreviation.	Naming follows language-specific conventions and avoids abbreviation.	Naming follows best practices throughout (including semantic variable naming, a.k.a. "calling it what it is").	Naming is clear enough that very few, if any, comments are necessary.
	2.2	**Indentation & Whitespace**	Code indentation is haphazard or nonexistent. No consistency in use of whitespace.	Most code is properly indented.	Most code is properly indented and whitespace is used consistently around punctuation, names, etc.	All code is properly indented and whitespace is used consistently according to language conventions.	All code is properly indented, whitespace usage is consistent, and code adheres to a documented style guide.
	2.3	**Comments**	No comments in areas where comments would have been desirable.	Comments are sporadic, or have become outdated due to code changes.	Comments are accurate, up-do-date, and address the "what".	Comments are used judiciously to explain and clarify difficult sections of code.	Comments are succinct, precise, and clearly address the "why".
	2.4	**DRY (Don't Repeat Yourself)**	Code has widespread duplication that could have been easily resolved using basic language constructs.	Code avoids only the most obvious points of duplication. Many sections are extremely similar and could be de-duplicated with little effort.	Code uses language and framework constructs to avoid most duplication, but some repetitive sections are present.	Code has almost no significant duplication, using language and framework constructs effectively to keep things DRY.	Code carefully balances conciseness with readability, showing a solid understanding of DRY principles.
	2.5	**SRP (SIngle Responsibilty Principle)**	Individual classes, methods, and templates are overloaded with logic, and there is no clear division of responsibilities.	Some division of responsibilities is evident, but many classes, methods, or templates contain excessive logic that could be factored out into smaller pieces.	The majority of classes, methods, and templates have a single responsibility, but code complexity is unevenly distributed and some areas are difficult to decipher.	Most classes, methods, and templates have a single well-defined responsibility. Complexity is evenly distributed throughout the code.	Virtually all classes, methods, and templates have a single well-defined responsibility. Code uses techniques for isolating functionality that go beyond the conventions provided by the framework.
	2.6						
	2.7						
	2.8						
	2.9						
	2.10						
	3	**Presentation, Team**					
	3.1	**Effectiveness of Presentation**	Presentation is unclear, inaccurate and/or incomplete.	Presentation accurately explains the project's purpose and how it works.	Presentation is clear and accurate, but lacks some detail and specificity.	Presentation is clear and illustrative, effectively outlining the project's goals, challenges, and solutions.	Presentation is compelling, and makes a good case for further development of the project.
	3.2	**Defense of Decisions Made**	Student does not attempt to defend their decisions.	Student addresses some issues but relies on faulty arguments or does not fully defend their decisions.	Student defends some decisions well, but glosses over others.	Student is able to comprehensively defend most major decisions involved in the project.	In defending their decisions, student shows that they have researched the issues thoroughly and considered multiple possible alternatives.
	3.3	*Group Contribution*	*Contributes minimaly to the group, with little to no code to their name.*	*Participates in the project, although contributions may be light/superficial.*	*Participates in the project, but is not a major contributor.*	*Contributes meaningfully to the project*	*Goes above and beyond, making an outsized contribution to the project (either from a literal 'lines written' perspective, or from a knowledge/vision perspective)*
	3.4	*Team Work*	*Student does not work well with others, and negatively impacts the group dynamic*	*Student is not disruptive to the rest of the group*	*Student is able to get work done within the confines of the team, without impeding others*	*Student works effectively with others to accomplish the team's goals.*	*In addition to working effectively with others, student contributes positively to the group dynamic.*
	3.5						
	3.6						
	3.7						
	3.8						
	3.9						
	3.10						