Evaluation Rubric

Category	Novice (1 point)	Intermediate (2 points)	Proficient (3 points)	Mastered (4 points)
Pitch Content	Shares basic information, such as purpose and target audience	Gives clear explanation of the app's purpose, design and how it addresses user needs	Presents clear and compelling explanation of the problem they're trying to solve, market demand, audience, and how the app was designed to meet user needs	Makes a persuasive pitch backed by evidence that shows how the app meets, exceeds, or redefines user needs
Pitch Delivery	Informational; one team member presents	Confident, enthusiastic; more than one team member presents	Engaging, good use of visuals to support story; team highlights contributions of each member	Creative, memorable storytelling; engaging visual support; smooth transitions between team members
User Interface	Consistent screens that support app's purpose	Clear, functional design with familiar elements; prototype supports basic user tasks	Elegant, concise, pleasing design with thoughtful use of color, layout, and readability; prototype gives user a sense of place within navigation	Design empowers the user to interact with content; prototype uses animation, color, and layout to create a seamless, engaging experience
User Experience	Clear intent; users can accomplish one or more goals	Consistent and standard navigation; intuitive path through app content	Adaptable to user needs; addresses accessibility, privacy, and security.	Innovative, surprising and delightful; gives users a new kind of experience that sets it apart from competitors
For the technical sections of this scoring rubric, one judge -- with familiarity with Swift and iOS development practices, will be designated to review and score.*				
Coding Concepts*	Some connection between app functionality and underlying code	Explanation of how general coding concepts like data types, conditional logic, or touch events relate to the app	Description of specific coding tasks necessary to build their app; demonstration of how that code powers the app's functionality	Explanation of the app's architecture, data structure, algorithms, and features; discussion of decision-making in developing this approach
Technical Review*	Swift code runs in specific examples; code is basic with no abstraction	Code runs without error in all cases; code is basic with some evidence of abstraction	Code is organized with clear Swift naming conventions; high evidence of abstraction; follows iOS guidelines	Code is well documented with comments; effective use of Swift features; employs organization, such as Model-View-Controller