2021

PS2 Code Judgement Rubric

ONLY CODE JUDGES SHOULD BE USING THIS VERSION OF THE RUBRIC!

Judge (not shared with students): _____
Judged Team (name or number): _____
Date: _____

Instructions: for each requirement, mark the box that most closely aligns to your impression of the solution. Your judgments here should reflect how well the solution satisfies the directions in the problem statement and the team's presentation as well as your own review of the solution's source code. **Aesthetic appeal of a solution is <u>not</u> more important than functionality.** If a solution is not pretty but otherwise meets a requirement, that counts as reasonably satisfying that requirement. Keep your expectations high, teams had **a whole month** to work on this. **DO NOT FEEL OBLIGATED TO AWARD HIGH SCORES!** In the case that a solution attempts to "satisfy" a requirement by making it *look* like it works via the UI/demo code but the required feature is actually non-functional, <u>a score of 0 is appropriate.</u>

Solution Functionality Review [total weight: 100]

Requirement(s) to Reference	Score			
	0	1	2	3
	Missing or Completely Non-functional	Attempted But Does Not Satisfy Requirement	Reasonably Satisfies Requirement	Exceeds Requirement Expectations
Change 1 Change 2	You can upload meme images directly from your local computer without entering a URL. **[weight 25]**			
Change 3	Users can change their password, profile picture, email, and/or phone number according to the text of the problem statement (PS2). **[weight 10]**			
	Profile pictures are now optional for users and no longer required during registration. **[weight 2]**			
Change 4	You can easily toggle the app into a "dark mode" visual theme according to the text of the problem statement (PS2). **[weight 10]**			
Change 5	All hashtags in the app appear alongside their trend data as described by the problem statement (PS2). **[weight 10]**			
Change 6	The app supports the new Spotlight view according to the text of the problem statement (PS2). **[weight 10]**			

	The Spotlight view shows an up to date list of the top five most used hashtags in the system along with their trend data. **[weight 3]**			
Change 7	When creating or sharing a new meme in the Chats view, you can send the meme to multiple users instead of just a single user. **[weight 5]**			
Change 8	When a meme sent to your account by another user *via Chats view* is 1) about to expire and 2) has not yet been viewed by your account, you receive a notification about it. Once the meme vanishes, the notification should no longer appear. **[weight 10]**			
Change 9	Users can block and unblock each other. When one user blocks another, they are no longer friends if they were, they cannot become friends, cannot send friend requests to each other, cannot see each other's memes in any view, and cannot send private memes to one another. **[weight 10]**			
Change 10	New logins (and their associated IPs) and updates to account data (i.e. email, phone number, profile picture, password) trigger alerts in the notifications view. **[weight 5]**			

Instructions: for each of the five criteria below, mark the box that most closely aligns to your impression of the solution's source code after your review. Does their source code meet your expectations of quality? **If you notice source code that seems strangely out of place, overly complex, extremely nonsensical, suspiciously unformatted or very badly organized syntactically, or you otherwise infer that students copy-pasted/plagiarized part of their solution's source code from the internet or some illegal source, inform the chief judge immediately.**

Solution Source Code Review [total weight: 100]

Criterion	Score			
	0	1	2	3
	Does Not Meet Expectations	Partially Meets	Meets Expectations	Exceeds Expectations
Naming Conventions	Variables should be well-named (e.g. **accountSum** is good while **acsu** is bad). There should be few "magic numbers" floating around. The chosen variable naming scheme should be consistent and self-documenting. **[weight 5]**			
Organization	The code should be well-organized, with a consistent logical file/directory layout, encapsulation, and proper separation of concerns. Similar units of code should be grouped together in a logical way. If object-oriented language features are used, they should be used properly (i.e. SOLID). **[weight 30]**			
Maintainability and Extensibility	The solution should be implemented such that a future group of programmers would not have to rewrite large portions of the source code to add new functionality. Source code should be DRY where reasonable. Pay special care to inflexible and hardcoded values. **[weight 25]**			
Readability	How easily readable is the source code? Did you have a hard time going through it? Solution code should be as intuitive and self-documenting as possible. Confusing sections should be documented with comments. There should be few "WTF?!" moments. **[weight 25]**			
Efficiency	Any implementations should not be too inefficient. Does some piece of code waste cycles looping or otherwise doing something unnecessary? Is the application wasteful with system resources? Does it make more network requests than necessary? **[weight 15]**			

Positive Comments

Please share any positive comments you have for this team.

Constructive Criticism

Please share any unanswered concerns or comments you have for this team.

Two Questions For This Team

Please share the two most pressing questions you'd like answered about this team's solution.

Shared Thoughts (visible to all judges)

Please add any thoughts or concerns you believe other judges should be made aware of.