BDPA
ENABLING DIGITAL THOUGHT LEADERSHIP.

2020

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

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	Only when you login as an administrator, you can ban and unban a user account. [weight 5]			
	A banned user cannot login. An unbanned user can. [weight 5]			
	A logged in user that is banned is forced to logout when they navigate to another page. [weight 5]			
	A banned user cannot book a flight. A banned user's credit card information cannot be used to book a flight even when buying the ticket as a guest (logged out). [weight 5]			
	Only when you login as an administrator, you can ban a user from booking a specific airline. When logged in as that user, you cannot book a flight with that banned airline. [weight 5]			

Change 2	Only as an administrator, you can create and delete attendant-type accounts. [weight 7.5]			
	Attendants have all the required abilities within the system (see requirement text) [weight 5]			
Change 3	When you book a flight, you do not get your ticket. Instead, you get a confirmation number. When not logged in, you can use this confirmation number and your last name to enter a check-in view if your flight departs within 24 hours. Otherwise, if the flight does not depart within 24 hours, you are told to try again later. [weight 7.5]			
	When logged in and looking at your dashboard for upcoming flights and the flight leaves in less than 24 hours, you should be able to check in and not have to enter your confirmation number and last name when you do. [weight 5]			
	You cannot check in for the same flight twice. [weight 5]			
Change 4	It seems like this solution is using the V2 API (all calls to the API are at the v2 endpoint instead of the v1 endpoint). For non-code judges: you don't see any strange API-related errors/app breakages. [weight 8.75]			
Change 5	When you purchase a ticket and you're logged in, you get frequent flier miles. You can go to your account dashboard to see how many frequent flier miles you have. You can purchase a ticket with your frequent flier miles. You can cancel your ticket and get your miles refunded to your account. [weight 8.75]			

Change 6	When booking a flight, the price of the flight changes depending on what type of seat you buy. You can purchase economy, exit row, economy plus, or first class seating now. For code judges: price data should be taken from the API/cached data. [weight 8.75]			
	When booking your flight, you can also purchase in-flight extras (wifi, pillow, etc). For code judges: teams should be pulling the data for this from a call out to the API/cached data. [weight 8.75]			
Change 7	There is a persistent sidebar in the app that shows the required information (check the requirement text). [weight 8.75]			
Change 8	New bookings cannot be made for flights that are scheduled to depart less than 36 hours from now. [weight 8.75]			
Change 9	When purchasing a flight, baggage fees and max number of checked/carry-on bags changes depending on the flight. For code judges: this information is coming from the API/cached data. [weight 8.75]			
Change 10	System logins are email-based. All you need to login is an email address and a password and not your full name or anything like that. [weight 8.75]			

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

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.