Rock, Paper, Scissors, Lizard, Spock

The Big Bang Theory

Create a web app game that can be played between two friends on their phones, tablets, and/or computers.
https://en.wikipedia.org/wiki/Rock_paper_scissors#Additional_weapons

Concept

Basic game as shown in the video played in class and described in Wikipedia. Each player makes a choice. The combination of choices results in a win/loss, loss/win, or tie. Unless there's a tie, express a win or loss with the phrase from the show. For example, if Spock and lizard are chosen, "lizard poisons Spock".

Requirements

1. Use **table look up** to determine winner/loser and winning phrase. Do not use branching statements.
2. The GUI design is up to you, but must include
    a. Must be responsive such that it looks good on wide screen, tablet, or phone.
    b. Preferences (all on by default, decisions must persist across devices unless noted)
        i. Dark Mode option (login screen must obey dark pref)
        ii. Vibrate option
        iii. Flashing lights option
        iv. Music option
        v. Sound Effects option
        vi. Tutorial/Tour option
3. You will do the server and the client, both compliant with the RPSLS protocol such that your peer's client can connect to your server and vice versa.
4. The server must handle 1000 simultaneous games.
    a. You must test 1000 simultaneous games to prove you met this requirement.
    b. You'll have to write an HTML bot and a command line bot that play the game.
    c. Then spin up 2050 bots to play the game on your peer's server, and vice versa.
5. Player must register as a new user the first time.
    a. Email (private) is the true user name.
    b. Display name (public) is shown to the world.
    c. Password (don't allow weak passwords)
    d. Must send email to complete registration.
6. Player must authenticate as a registered user.
    a. The first authentication starts the tutorial.
7. Player will always be displayed with a win/loss ratio, e.g. Chicken Head 5/7, Bone 42/170, Babe 0/0
8. Players are first listed in the World lobby.
9. After the first win, player is immediately introduced to guilds and given an option to create a new guild, join an existing guild that's accepting members, or continue playing solo. Guild names always display

the win/loss ratio of the players, e.g UVU Warr10r$ 394/185. Player can leave a guild anytime. Player can join a guild anytime after the first win, even if the player previously chose to continue playing solo.

10. A complete game is best out of 7. If a player disconnects from a game in progress, the game counts as a loss for the disconnecting player and no change for the remaining player.
11. Must use this stack:
    a. **Vue3** (try to do it entirely in Vue; RARE and MINIMAL overrides with JS are allowed)
        i. **Vue routing** (client routing)
        ii. **Pinia** (state management)
        iii. **Vite** (build)
        iv. **Nuxt3**
    b. **Tailwind** (try to do it entirely in Tailwind; RARE and MINIMAL overrides with CSS are allowed)
    c. **Node/Express**
    d. **Mongoose** (mongodb hosted at Atlas)
    e. **Netlify** hosted
    f. TBD
12. List of preapproved (and ~~rejected~~) optional libraries for client and/or server. Feel free to request a library. The list will grow as libraries are approved.
    a. socket.io (both)
    b. axios (both)
    c. lodash (both)
    d. helmet (server)
    e. morgan (server)
    f. mongoose (server)
    ~~g. passport (server)~~
    ~~h. vue-socket.io (client)~~
    ~~i. vue-socket.io-extended (client)~~
    ~~j. pug~~
    k. cors (server) (use it selectively, judiciously, so that the client remains safe)
    l. vuetify or another vue component library
    m. fontawesome
    n. nodemailer
13. Users may join a game in progress between two players as a commentor.
14. Players in a game can choose multiple items to submit in a turn. Only the decision is required.
    a. decision (rock, paper, scissors, lizard, spock)
    b. emoticon(s) 🦆💩
    c. text message (talking trash is optional; the player can be supportive, e.g. "Good move amigo"
    d. speech recording
15. Deliver a zip file from your vsCode (no node_modules folder, or you die from monkey pox with a slice of ebola.)
16. I promise there will be
    a. missing requirements
    b. added requirements
    c. conflicting requirements
    d. ambiguous requirements
    e. bad requirements that must be removed

Don't whine. Don't complain to the department chair. EXPECT IT. Go with the flow. Dodge the punches and keep moving. Do not stop because you have a question on one feature.

~~RPSLS Protocol v0.1~~

see RPSLS Protocol v0.2

FAQ

Q2. Do I have to draw the 5 symbols with Canvas or SVG?
A2. No, you can if you want, but you don't have to. I assigned those spikes to give you a tool for your toolbelt, so you know you could user either one if you ever wanted.

How you display the 5 elements is up to you. Where you get the 5 images is up to you. Whether or not the images are animated is up to you.

Q3. I hate "up to you" requirements. Will I still get penalized for bad decisions?
A3. Yes, "up to you" is not a get-out-of-jail-free card. If I say, "That decision sucks", you can't use *up to you* in the requirements as an excuse for a poor quality decision. How you design an app is up to you. How others evaluate it is not up to you.

Q4. I have an iPhone. iOS doesn't allow [some feature]. What do I do?
A4. I didn't say it had to work on an iPhone. It must work on any phone that allows it. On a phone that doesn't allow it, have a fall back approach.

Q5. I'm feeling overwhelmed. Can you suggest how to tackle this?
A5.

1. Come to EVERY class.
2. Talk to your peers. (Talking about code is allowed. Sharing code is cheating.)
3. Pick the smallest subset of the requirements that gets a basic game going, aka MVP from your software engineering class. Get the MVP working without worrying about getting all the points for all the requirements.
4. Pick the next easiest requirement to add to your working MVP. Get it working.
5. Rinse and repeat until done.

Q6. I don't understand how to use socket.io rooms for simultaneous games between 1000 friends. How do I do that?
A6. You don't have it working between 2 friends. You're putting the cart before the horse. You must learn iterative programming if you want to survive in your career. Do a little at a time, concentrating on one feature until done is the best strategy. **The number one way to an epic fail is coding all the requirements before seeing if it will work.** Can

It's 1000 simultaneous *games;* 2 per game is 2000 players. Plus you need to add 50 commenters.

Q7. Why are the vue-socket libraries banned?
A7. Too many students got burned and blamed me--which is silly because I never recommended them. Even if you swear on a stack of bibles that you will take full responsibility, the answer is no. Even if you talked with an ex-student who said he or she LOVED that library, the answer is no. Don't ask. I've been down that road, and I'll be damned if anyone is taking me down that road again. Experience is learning to avoid getting burned again.

Q8. OK, I can see that vue-socket.io merge libraries are a hot button for you. I won't ask for one. Then how can we use both vue and socket.io?
A8. You googled *vue socket.io*, and the top 10 answers pointed to tutorials that show using one of those libraries. Now you are stuck like a giant elephant held in place by a puny chain. Free your mind. Feeling you must use a merging library is a trap unworthy of a CS grad. I will show you in class who vue can use other libraries without some fancy schmancy merged monstrosity made by a drunk monkey with a martini and a gun. Seriously, the name of some libraries is way better than the quality inside.

Q9. I don't get it. socket.io is for chat between two browsers. How can it be used for binary?
A9. The first day of socket.io in class, I read outloud from socket.io's site that it can be used for binary. That socket.io's Getting Started tutorial didn't use binary doesn't mean anything.

Q10. Will I be graded on code style?
A10. Yes, that's always the case. Even after school, you are judged by your code.

A portion of the grade will be on code organization, and that includes style. I'm not going to dictate a style in this class. I do require you to pick an industry style and be consistent.

Q11. Do you have a code pet peeve?
A11. No, I don't have a single code pet peeve. I have many. Don't worry. I won't hold you responsible because I didn't tell you.

Vertical braces in a JavaScript function is complete crap because nobody does it. You are spitting into industry's face. I said to pick an industry style. I didn't say pick your own style.

Vertical braces in JSON is a different story because everyone does it.

Q12. What are the common *gotchas*?
A12. The most common minor gotcha is leaving in commented out code. Take out your trash before you submit your code.

The most common major gotchas are

- **Procrastinating**. You can't wait until you understand everything before you start. The logs are there to help your anti-procrastinating effort.
- **Not understanding or forgetting the requirements**. Make a checklist for yourself.
- **Skipping spikes**. Why are you adding in a new stack before trying it out in a simple isolated example?
- Not doing **Baby Steps**. Re-read #A6 above.

Q13. You mentioned something about milestones, MVP, and early deliveries. What do I have to deliver when?
A13. I said you should deliver what you did for the project every week. You pick the day and be consistent.

Q14. Can you give a hint of what could be in the pop inspections?
A14.

1. **Break it up.** The big project becomes medium pieces, and then those medium pieces into smaller pieces, and so on.

2. **Estimate the efforts** on all the little pieces.

3. **Total effort** is the sum of the pieces.

4. **Weekly effort** is the total effort divided by the number of weeks.

48 hrs / 6 wks = 8 hours per week

60 hrs / 5 wks = 12 hours per week

5. **Schedule** each week's delivery by assigning every piece to a delivery such that every week's delivery has roughly the weekly effort. Don't overthink it. In a 4 week project, after 1 week, you deliver 1/4 of the total project. After 2 weeks, you deliver 2/4 of the total. After 3, you deliver 3/4. And after 4, you deliver the final delivery, the complete project. Note: you are 1 week into a 5 week project. You do the math.

5. **Publish** your schedule.

6. **Deliver** on your schedule.

Q15. You didn't say all that in lecture. Did you?
A15. Yeah, I did. If it seems new, you blew off what I said as friendly advice that you weren't accountable for.

Q16. OK, early deliveries seem simple the way you explained it. How do *MVP* and *milestones* fit in there?
A16.

Minimum Viable Product

MVP is the smallest subset of pieces that provide value to the customer. As we discussed in class, after one week, if you totally finish the MongoDB installation, schema, and even prepopulate it with fake users, but you don't touch the server or client, you may have done 1/5 of the total work and still have no value for the customer. As far as the customer is concerned, you did blah, blah, blah, WHATEVER. If the customer knew what MongoDB was, the customer might not need you. In this case, the MVP would be 2 players can play a single game of RPSLS

- not best of 7 games
- no MongoDB
- no list of players
- no commentators
- no simultaneous games
- no history
- no chat
- no voice
- no icons
- only the barest of GUI necessary to play a single game

After developing the first MVP, do the next MVP, that is, pick a feature that adds value for the customer. And then the next MVP. On this project you could do a hundred MVP. The hardest and longest MVP is the first one. After that, the next few might be super quick.

Milestones

The Romans were the first to build paved highways and mark them with milestones. We do the same today with mile markers on I-15. Back before GPS, you could see how close you were to your destination by noting the mile marker as you passed it.

Milestones are arbitrary sub-goals towards the project goal of a complete final delivery. Unlike physical milestones on highways and trails, milestones don't have to evenly spaced. It's nice when they are, but it rarely happens.

Relationship to Weekly Deliveries

Both guide you in picking a good schedule. They give you a focus that helps

- give you a sense of progress
- give the customer a sense of progress
- give you and the customer a way to deliver on time by cutting out the least valuable pieces if necessary.
- give you vision for improvement
- give the customer vision for improvement
- keep you ready at any time for a pop inspection request of seeing a demo

Q17. What do you mean by *vision for improvement*?
A17. Two English bricklayers were doing their job in the hot August sun. A passerby asked one what was happening. The first said, "I sweating my a[redacted]off bricking for The Man. What the F[redacted] does it look like I'm doing?"

When asked in turn, the second said, "I'm building a cathedral to God."

Like the bricklayers, you'll do the coding either way. You can choose to stay at the base work level, or you can also participate with the customer in making a great game for players.

You and the customer can't improve what doesn't exist. Improvement ideas really start flowing from experiencing the MVP.

Q18. Why did you say we need the MVP? That's BS. I have tons of improvement ideas now.
A18. I accept that. Does the customer?

Q19. Back to the relationship of MVP and milestones to weekly deliveries. Are you saying an MVP is a milestone? Are you saying the weekly deliveries should be MVPs?
A19. An MVP is certainly a milestone. That doesn't mean a milestone is an MVP. Getting Hello World to work in MongoDB is a milestone, but it's not an MVP. Just as the final delivery was preceded by weekly deliveries, a MVP milestone could be preceded by other milestones.

Look. Let's keep this simple. You determine the milestones that mark your progress towards a weekly delivery. I only have two hard constraints:

1. You declare the milestones at the start of developing the weekly deliverable, not retroactively.
2. No soft fuzzy milestones like
    - *"Learned about Mongo",*
    - *"Designed how I'll handle the RPSLS protocol."*
    - *"Read the specs again."*

- *"Discussed how to test 2050 bots with a peer."*
- *"Researched the best way to write a web bot."*
- *"Chose a Vue component library."*

Q20. Why isn't my question answered here?
A20. Ask it in the 4690 channel in Discord and I'll answer it. I'm going there now to check for questions.

Q21. I have to deliver the first MVP this week?
A21. Although that's a worthy goal, maybe you don't get it done. Deliver something! We're ending week 2 of 5 weeks.

Q22. Holy crap, Batman! I should have delivered 40%?
A22. In a perfect world, yes. Since we don't live in a perfect world, do what you can in the remaining time. The first week is often a wash. The second week shouldn't also be a wash. Deliver something.

Q23. If I include this week, it'll be 4 weeks. I should deliver 25%?
A23. Ah, you have some skill in math. Yes, 1/4 == 25**%.** That would be the goal to be sure you make the final delivery on time. If you can't deliver 25%, deliver something.