so.. I have completed the question generation part. Now moving forward for probability logic. My server returns qns through json format. I want you to code out the probability part in javascript accordingly

Nice, good progress.

Below is a minimal, self-contained JS implementation for the probability-based adaptive logic that fits your JSON format from `server.py` (`difficulty` is already there on each question).

Assumptions (you can tweak constants easily):

- Each question object from backend looks like:
  `{ question, options, correctoptionletter, difficulty, ... }`
- Ability θ starts at 0.
- Discrimination $a = 1$.
- Learning rate `ALPHA = 0.4`.
- Target probability zone: 0.4–0.6 (around 0.5).

1. Core helpers (probability + updates)

```
// --- CONFIG ---
const ALPHA = 0.4;      // how fast ability changes
const A_PARAM = 1.0;    // discrimination
const THETA_MIN = -3;
const THETA_MAX = 3;

// Logistic 2PL probability: P(theta) = 1 / (1 + e^(-a (theta - b)))
function calcProbability(theta, difficulty, a = A_PARAM) {
  const exponent = -a * (theta - difficulty);
  return 1 / (1 + Math.exp(exponent));
}

// Update ability after an answer
// isCorrect: boolean, difficulty = question.difficulty (b)
function updateTheta(theta, difficulty, isCorrect) {
  const p = calcProbability(theta, difficulty);

  let newTheta;
  if (isCorrect) {
    // correct: bigger jump if question was unlikely to be correct
    const surprise = 1 - p;
    newTheta = theta + ALPHA * surprise;
  } else {
    // incorrect: bigger drop if question was expected to be easy
    const disappointment = p;
    newTheta = theta - ALPHA * disappointment;
  }

  // clamp
  if (newTheta > THETA_MAX) newTheta = THETA_MAX;
  if (newTheta < THETA_MIN) newTheta = THETA_MIN;

  return newTheta;
}
```

2. Selecting the next question using P(θ)

Given:

- `questions`: array of question objects from your server
- `askedIds`: Set of already used question IDs (or indexes)
- `theta`: current ability

```
// Choose the question whose P(theta) is closest to 0.5
function selectNextQuestion(questions, askedIds, theta) {
  const TARGET_P = 0.5;

  let bestQuestion = null;
  let bestIndex = -1;
  let smallestDiff = Infinity;

  questions.forEach((q, index) => {
    if (askedIds.has(index)) return; // skip already asked

    const b = q.difficulty ?? 0;  // default 0 if missing
    const p = calcProbability(theta, b);
    const diff = Math.abs(p - TARGET_P);

    if (diff < smallestDiff) {
      smallestDiff = diff;
      bestQuestion = q;
      bestIndex = index;
    }
  });

  return { question: bestQuestion, index: bestIndex };
}
```

3. How to integrate on the frontend (React-style)

Example flow in JS:

```
// state-like variables
let theta = 0;                  // current ability
const askedIds = new Set();     // indexes of asked questions
let questions = [];             // fill from your backend

// 1) After fetching quiz from /generate-quiz
//    Suppose backend returns: { quizname, questions: [...] }
async function startQuiz() {
  const res = await fetch("http://localhost:8000/generate-quiz", {
    method: "POST",
    // formData etc based on your existing code
  });
  const data = await res.json();
  questions = data.questions;

  askNextQuestion();
}

function askNextQuestion() {
  const { question, index } = selectNextQuestion(questions, askedIds, theta);
  if (!question) {
    console.log("No more questions available");
    return;
  }
  askedIds.add(index);

  // Now render 'question' on UI
  renderQuestionOnUI(question);
}

// 2) When user submits an answer
function handleAnswerSubmit(selectedOptionLetter, currentQuestion) {
  const isCorrect = selectedOptionLetter === currentQuestion.correctoptionletter;

  // update theta using this question's difficulty
  theta = updateTheta(theta, currentQuestion.difficulty, isCorrect);

  console.log("Answer:", isCorrect ? "correct" : "wrong");
  console.log("Updated theta:", theta);

  // then move to next question
  askNextQuestion();
}
```

4. If you want a simpler “+0.4 / -0.4” version

If you just want the clean version that always does `θ = θ ± 0.4` (as in your flow), but still uses P(θ) only for selection:

```
function updateThetaSimple(theta, isCorrect) {
  const STEP = 0.4;
  let newTheta = isCorrect ? theta + STEP : theta - STEP;
  if (newTheta > THETA_MAX) newTheta = THETA_MAX;
  if (newTheta < THETA_MIN) newTheta = THETA_MIN;
  return newTheta;
}
```

Then swap `updateTheta` with `updateThetaSimple` in `handleAnswerSubmit`.

If you paste your current frontend snippet (where you handle the quiz JSON and user answers), code can be wired directly into that.

⁂

1. server.py
2. speakerLLM.py