**Sharing Code Without Sharing Systems**

**The Access Dilemma**

Imagine you've created an exceptional calculator program that everyone wants to use. Currently, the only way someone can use your calculator is by having direct access to your computer.

**Question to consider:** How might you allow others to use your calculator's functionality without ever letting them see or touch your computer? What boundaries could separate users from your actual code while still delivering its value?

**Extension scenario:** A mathematics professor wants to incorporate your advanced statistical functions into her online course for 500 students across the country. She needs students to perform calculations but doesn't want them to install anything on their personal devices. How might you solve this problem?

**The Value Protection Challenge**

Your calculator does things that nobody else's can. Other developers want to incorporate your calculations into their own applications, but you don't want to give away your secret sauce.

**Question to consider:** How might you create a bridge between your code and others' applications that allows them to use your calculator while keeping your implementation details hidden? What models exist for allowing others to "rent" functionality without owning it?

**Extension scenario:** A financial technology startup wants to use your specialized currency conversion algorithm in their mobile app, but they're concerned about reverse engineering. How could you allow them to access your functionality while protecting your intellectual property? What might the communication between their app and your service look like?

**The Tiered Service Puzzle**

Not all customers are equal. Some might need basic calculations while others need advanced features you've developed.

**Question to consider:** How could you design a system that allows different levels of access to different features of your calculator? What patterns exist in other digital products that allow for basic, premium, and enterprise access levels without changing the core product?

**Extension scenario:** You have three distinct customer segments:

1. Students who need basic operations (free tier)
2. Small businesses that need financial functions (paid tier)
3. Research institutions that need advanced statistical capabilities (premium tier)

How would you design your system to recognize these different users and provide appropriate access? How would you handle unauthorized attempts to access premium features?

**The Scale Conundrum**

Success brings challenges. What happens when your calculator becomes so popular that thousands or millions of people try to use it simultaneously?

**Question to consider:** How might computer systems handle massive spikes in usage? What happens when your single program needs to serve many users at once? Think about how popular websites manage to serve millions of users without crashing.

**Extension scenario:** Your calculator service suddenly goes viral after being featured in a popular YouTube video. Usage spikes from 100 requests per hour to 10,000 requests per minute. Your single server is overwhelmed. What architectures could help you scale quickly without rewriting your entire application?

**The Global Reach Challenge**

Your calculator has become popular internationally, with users from multiple countries.

**Question to consider:** How might you optimize service delivery for users who are geographically distant from your original server? What happens to calculation speed when a request has to travel across oceans?

**Extension scenario:** Users in Australia are complaining about slow response times while using your calculator service. Your server is in New York. How might you improve their experience without compromising the integrity of your service or introducing unsustainable complexity?

**The Documentation Dilemma**

For others to use your calculator effectively, they need to understand how to interact with it.

**Question to consider:** How would you communicate the capabilities of your calculator to potential users without revealing its inner workings? What information would developers need to successfully integrate with your calculator?

**Extension scenario:** A team of developers from a non-English speaking country wants to integrate your calculator into their application. They're struggling to understand how to format requests correctly and interpret the responses. What solutions might address this problem?

**The Monitoring Mystery**

As usage of your calculator grows, you need to understand how people are using it and identify potential problems.

**Question to consider:** How might you gain visibility into the usage patterns of your calculator when you can't see users' screens or access their devices? How would you know if something is going wrong?

**Extension scenario:** Several users report receiving incorrect results for complex calculations, but you can't reproduce the issue on your end. How might you design your system to help troubleshoot problems that only appear under specific conditions?

**The Evolution Enigma**

Your calculator needs to evolve with new features and improvements without disrupting existing users.

**Question to consider:** How might you update your calculator's functionality without breaking applications that already depend on it? What strategies exist for evolving software that serves as a dependency for other systems?

**Extension scenario:** You've completely rewritten your calculation engine to be 10x faster, but the input and output formats have changed. How might you roll out this improvement while ensuring current integrations continue to work?

Each of these challenges points toward principles of service-oriented architecture and web APIs. Encourage your students to research how modern applications share functionality across the internet while maintaining security, scalability, and business viability.