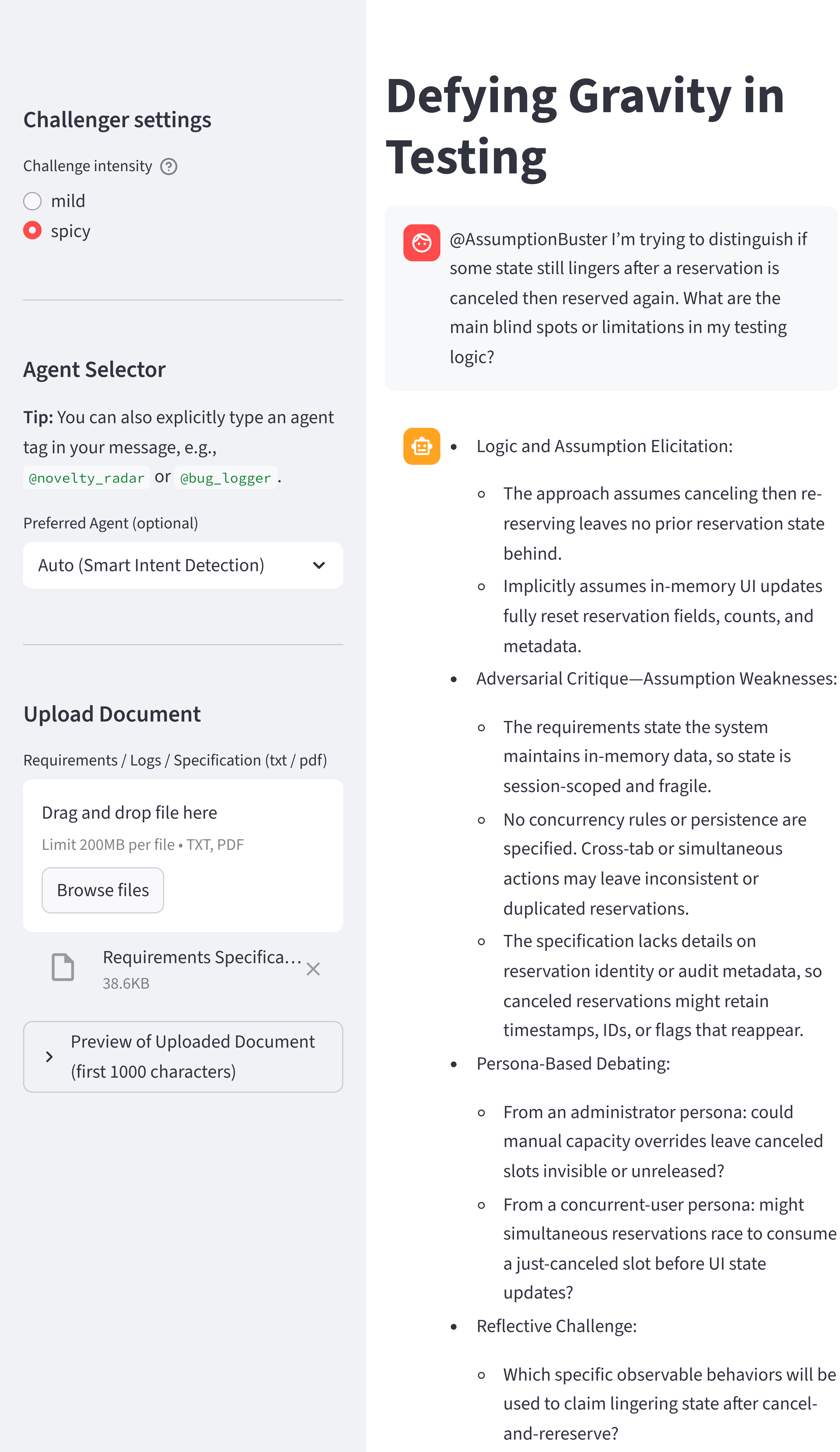

Challenger settings

Challenge intensity

○ mild

● spicy

Agent Selector

Tip: You can also explicitly type an agent tag in your message, e.g., `@novelty_radar` or `@bug_logger`.

Preferred Agent (optional)

Auto (Smart Intent Detection)

Upload Document

Requirements / Logs / Specification (txt / pdf)

Drag and drop file here
Limit 200MB per file • TXT, PDF
Browse files

Requirements Specifica…
38.6KB

Preview of Uploaded Document (first 1000 characters)

Defying Gravity in Testing

@AssumptionBuster I’m trying to distinguish if some state still lingers after a reservation is canceled then reserved again. What are the main blind spots or limitations in my testing logic?

- Logic and Assumption Elicitation:
  - The approach assumes canceling then re-reserving leaves no prior reservation state behind.
  - Implicitly assumes in-memory UI updates fully reset reservation fields, counts, and metadata.
- Adversarial Critique—Assumption Weaknesses:
  - The requirements state the system maintains in-memory data, so state is session-scoped and fragile.
  - No concurrency rules or persistence are specified. Cross-tab or simultaneous actions may leave inconsistent or duplicated reservations.
  - The specification lacks details on reservation identity or audit metadata, so canceled reservations might retain timestamps, IDs, or flags that reappear.
- Persona-Based Debating:
  - From an administrator persona: could manual capacity overrides leave canceled slots invisible or unreleased?
  - From a concurrent-user persona: might simultaneous reservations race to consume a just-canceled slot before UI state updates?
- Reflective Challenge:
  - Which specific observable behaviors will be used to claim lingering state after cancel-and-rereserve?

  - What hidden dependencies or environment constraints is the testing logic implicitly trusting?

Type here… (e.g., '@Brainstormer Find unseen