3.1 Python Directory Structure

```
backend/
  main.py
  rate_limiter.py
  models.py
  policy_resolver.py (optional)
frontend/
  src/App.jsx
```

3.2 Data Structures

Redis Key Layout

```
rl:user:{userId}:model:{modelId}
rl:tenant:{tenantId}:model:{modelId}
rl:apikey:{apiKey}:model:{modelId}
rl:model:{modelId}:global
rl:qos:{tier}:{modelId}
```

Redis Value Structure

- Sorted Set (ZSET)
- score = timestamp (ms)
- value = timestamp or UUID

TTL

- windowSeconds * 2
- Ensures unused keys die naturally

3.3 Sliding Window Algorithm (LLD)

```
def allow(key, limit, windowSeconds):
    now = current_time_ms()
    windowStart = now - windowSeconds*1000

    # Lua script
    ZREMRANGEBYSCORE key 0 windowStart
    current = ZCARD key
    if current >= limit:
        return false

    ZADD key now now
```

```
    EXPIRE key windowSeconds * 2
    return true
```

3.4 Lua Script (Atomic)

```
redis.call("ZREMRANGEBYSCORE", KEYS[1], 0, ARGV[2])
local current = redis.call("ZCARD", KEYS[1])
if current >= tonumber(ARGV[3]) then
    return current
end
redis.call("ZADD", KEYS[1], ARGV[1], ARGV[1])
redis.call("EXPIRE", KEYS[1], ARGV[4])
return current + 1
```

3.5 Python Implementation (FastAPI)

main.py

```
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
import redis

from rate_limiter import SlidingWindowRateLimiter
from models import RateLimitRequest, RateLimitResponse

app = FastAPI()
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],
    allow_methods=["*"],
    allow_headers=["*"],
)

redis_client = redis.Redis(host="localhost", port=6379)
rate_limiter = SlidingWindowRateLimiter(redis_client)

DEFAULT_LIMIT = 100
DEFAULT_WINDOW_SECONDS = 3600

@app.post("/rate-limit/check", response_model=RateLimitResponse)
def check_rate_limit(body: RateLimitRequest):
    key = f"rl:user:{body.userId}:model:{body.modelId}"
    allowed, count = rate_limiter.check_and_consume(
        key, DEFAULT_WINDOW_SECONDS, DEFAULT_LIMIT
```

```
    )
    return RateLimitResponse(
        allowed=allowed,
        limit=DEFAULT_LIMIT,
        count=count,
        windowSeconds=DEFAULT_WINDOW_SECONDS
    )
```

3.6 React Frontend LLD

App.jsx contains:

- Text inputs
- Button
- fetch POST → shows *ALLOWED / BLOCKED*
- Compact and demo-ready