Step 1: Requirements

Functional requirements:
- Users upload data (text only) and get unique URL to access it
- Data and links expire after specified time

Non-functional requirements:
- Highly reliable, highly available
- Access to pastes in real-time with minimum latency

Step 2: Back of Envelope Calculations

Traffic
- Limit pastes to 10MB
- Assume 5:1 read:write ratio
- 1M new pastes per day → 5M reads per day → 12 pastes/sec and 58 reads/sec

Storage
- 1M*10KB(average)=10GB/day
- Store for 10 years → 36TB
- 3.6B pastes in 10 years → with bit-64 need 6-letter strings → 64^6=68.7B unique strings
- This means 3.6B*6=22GB storage if each character is 1 byte

Bandwidth
- 12 new pastes/sec → 120KB/s ingress
- 58 req/sec → 0.6MB/s egress

Memory
- 80-20 rule> 0.2*5M*10KB=10GB cache size

Step 3: System Interface Definition

- addPaste(api_key, paste_data, custom_url, user_name, paste_name, exp_date)
- getPaste(api_key, api_paste_key)
- deletePaste(api_key, api_paste_key)

Step 4: Define Data Model

- Need to store billions of records (< 1KB)
- Each paste object is medium size (few MBs)
- No relationship between records
- Read heavy
- Paste table with PK URLHash(varchar(16)) - ContentKey, ExpirationDate, UserID, CreationDate
- User table with PK UserID(int) - Name, Email, CreationDate, LastLogin