Analysis: 2-Month Timeframe Feature for Telegram Summarizer Bot

Date: October 27, 2025
Task: Assess feasibility of `/summarize 60d` or `/summarize 2mo` commands

Current Implementation Status

1. Timeframe Parsing Logic (timeframe_parser.py)

The `TimeframeParser` class currently supports:

✅ Supported Formats:

- `today` - Today's messages
- `yesterday` - Yesterday's messages
- `last N hours` - Relative hours (e.g., `last 2 hours`)
- `last N days` - Relative days (e.g., `last 3 days`)
- `last N weeks` - Relative weeks (e.g., `last 1 week`)
- `from YYYY-MM-DD to YYYY-MM-DD` - Date ranges
- `on YYYY-MM-DD` - Specific day

❌ NOT Currently Supported:

- `60d` format (shorthand day syntax)
- `2mo` or `2m` format (month syntax)
- Month-based timeframes at all

Key Code Location: Lines 20-22 in `timeframe_parser.py`

```
RELATIVE_PATTERN = re.compile(
    r'^last\s+(\d+)\s+(hour|hours|day|days|week|weeks)$',
    re.IGNORECASE
)
```

This regex only matches `hour`, `day`, and `week` units.

---

2. Message History Limits

In-Memory Storage:

- **Constant:** `MAX_STORED_MESSAGES = 100` (line 352 in bot.py)
- **Limitation:** Only stores the last 100 messages per chat
- **Age Filter:** `MAX_MESSAGE_AGE_HOURS = 24` (line 43 in bot.py) for count-based queries
- **Impact:** Without database, can ONLY summarize the last 100 messages, regardless of timeframe

Database Storage (PostgreSQL):

- **No inherent limits** on message count

- **No age restrictions** on stored messages
- **Can retrieve unlimited messages** within any timeframe
- **Cleanup:** Optional 30-day cleanup function exists but is not automatically called

Key Finding:

The timeframe feature ONLY works properly with PostgreSQL database enabled!

When using timeframes (lines 419-426 in bot.py):

```
if start_time is not None:
    # Try database first
    if db_manager.enabled:
        return await db_manager.get_messages_by_timeframe(...)

    # Fall back to in-memory with timeframe filter
    # ⚠️ This only checks the last 100 messages in memory!
```

---

3. Message Fetching Limitations

Telegram API Reality:

The code at lines 167-177 in bot.py has a critical comment:

```
# Note: Telegram API doesn't provide direct history access
# We'll work with what's available in the bot's context
# This is a limitation - bots can't access full message history
```

Key Constraint: The bot can ONLY see messages that occurred AFTER it was added to the chat. It cannot fetch historical messages from before it joined.

This means:
- Bot must be actively running and storing messages in real-time
- Cannot retroactively fetch 2 months of history
- Must be present in the chat for 2 months to summarize 2 months

---

Feasibility Assessment: `/summarize 60d` or `/summarize 2mo`

✅ Would Work IF:

1. **PostgreSQL Database is Enabled**
   - User has `DATABASE_URL` environment variable configured
   - Database has been storing messages continuously

2. **Bot Has Been Active for 2+ Months**
   - Bot must have been running in the chat for the entire period
   - All messages during that time were stored

3. **Parsing Support is Added**
   - Need to add `60d` or `2mo` syntax support to `TimeframeParser`
   - Current workaround: Use `last 60 days` (already supported!)

❌ Would NOT Work IF:

1. **No Database** (in-memory only)
   - Limited to last 100 messages
   - Can't store 2 months of data

2. **Bot Wasn't Active**
   - Can't fetch historical messages from before bot joined

3. **User Uses Shorthand Syntax** (without code changes)
   - `60d` ❌ Not recognized
   - `2mo` ❌ Not recognized
   - `last 60 days` ✅ Already works!

Practical Limitations Analysis

1. Message Volume Estimation

Typical Chat Activity:
- **Low activity:** 10 messages/day = 600 messages/2 months
- **Medium activity:** 50 messages/day = 3,000 messages/2 months
- **High activity:** 200 messages/day = 12,000 messages/2 months
- **Very active group:** 1,000 messages/day = 60,000 messages/2 months

2. Claude API Token Limits

From line 259 in bot.py:

```
max_tokens=500  # Output limited to 500 tokens
```

Claude 3 Haiku Context Window: 200,000 tokens (~150,000 words)

Estimated Message Sizes:
- Average message: ~50 words (~67 tokens)
- With formatting: ~100 tokens per message

Token Calculations:

Messages	Input Tokens (est.)	Within Context?	Cost Estimate*
600	60,000	✅ Yes	$0.15
3,000	300,000	❌ **EXCEEDS**	N/A
12,000	1,200,000	❌ **EXCEEDS**	N/A
60,000	6,000,000	❌ **EXCEEDS**	N/A

*Based on Claude 3 Haiku pricing: $0.25 per million input tokens

Critical Finding:
- 🚨 **Any medium-to-high activity chat over 2 months will EXCEED Claude's context window!**
- Maximum feasible: ~2,000-3,000 messages (depending on message length)

3. Telegram API Limits

PostgreSQL Query Performance:
- No hard limits in database code (database.py lines 199-249)
- Query uses indexed columns (chat_id, date)
- Should handle 10,000+ messages efficiently

Network Transfer:
- Postgres connection timeout: 60 seconds (line 53, database.py)
- Could timeout with very large result sets

4. Cost Considerations

Claude API Costs (Haiku model):
- Input: $0.25 per million tokens
- Output: $1.25 per million tokens

Cost Examples:
- 600 messages: ~$0.15 per summary
- 1,500 messages: ~$0.38 per summary
- 3,000 messages: Would fail (exceeds context window)

For a moderately active bot:
- 10 summaries/day × 30 days = 300 summaries/month
- If averaging 600 messages each: ~$45/month
- 🚨 **Could get expensive quickly!**

Recommended Safeguards & Warnings

1. Message Count Limits

Recommendation: Add a hard limit to prevent token overflow

```
MAX_MESSAGES_FOR_SUMMARY = 2000  # Conservative limit for Claude context
```

When timeframe exceeds this:

```
⚠️ Warning: The timeframe "last 60 days" contains 15,823 messages.

For optimal results, I can only summarize up to 2,000 messages at once
due to AI processing limits.

Options:
• Summarize the most recent 2,000 messages: /summarize 2000
• Break into smaller periods: /summarize last 30 days
• Get specific date ranges: /summarize from 2024-09-01 to 2024-09-15
```

2. Cost Warning for Large Timeframes

Recommendation: Add warnings for timeframes > 7 days

```
💰 Cost Notice: Summarizing 30+ days of messages uses significant AI processing.

Estimated: ~1,200 messages, approximate cost: $0.30

Proceed? Reply with /confirm_summary to continue.
```

3. Database Requirement Check

Recommendation: Validate database is enabled for timeframe queries

```
if start_time is not None and not db_manager.enabled:
    await update.message.reply_text(
        "⚠️ Timeframe summaries require database storage.\n\n"
        "Currently using in-memory storage (last 100 messages only).\n\n"
        "To enable timeframe summaries:\n"
        "1. Set up PostgreSQL database\n"
        "2. Configure DATABASE_URL environment variable\n"
        "3. Restart the bot"
    )
    return
```

4. Smart Sampling for Large Timeframes

Recommendation: For >2000 messages, use intelligent sampling

Options:
- Sample every Nth message
- Prioritize messages with keywords/questions
- Show daily/weekly summaries instead of message-by-message

Example approach:

```
📊 The timeframe "last 60 days" has 8,234 messages.

I'll create a summarized overview using a representative sample
of 1,500 messages across the entire period...
```

5. Progress Indicators

Recommendation: Show progress for large summaries

```
🤔 Analyzing 1,847 messages from the last 60 days...
   ⏳ Loading messages from database... (15%)
   ⏳ Processing with AI... (60%)
   ⏳ Formatting summary... (90%)
   ✅ Complete!
```

Current Capabilities Summary

✅ What Already Works:

1. `/summarize last 60 days` ← Already supported!
   - Uses existing `RELATIVE_PATTERN` regex
   - Works with database enabled
   - No code changes needed

2. **Database Storage**
   - Unlimited message history (if database configured)
   - Efficient timeframe queries

3. **Timeframe Infrastructure**
   - Solid parsing system
   - Database query methods
   - Age filtering

❌ What Doesn't Work Yet:

1. **Shorthand Syntax**
   - `60d` not recognized
   - `2mo` not recognized
   - Need to extend regex patterns

2. **Safety Limits**
   - No maximum message count
   - No cost warnings
   - No token overflow protection

3. **User Experience**
   - No feedback on database requirement
   - No warnings for large timeframes
   - No progress indicators for slow queries

Conclusion & Recommendations

Answer to Original Questions:

1. Would `/summarize 60d` work?

❌ **Not currently** - syntax not supported
✅ **But `/summarize last 60 days` DOES work!**

2. Would `/summarize 2mo` work?

❌ **No** - month syntax not implemented at all

3. What are the practical limitations?

Limitation	Severity	Mitigation
Telegram API (no history fetch)	🔴 **Critical**	Bot must be active continuously
Database Required	🟡 **Major**	Document requirement clearly
Claude Context Window (~2000 msgs)	🔴 **Critical**	Add hard limits + sampling
Cost ($0.15+ per summary)	🟡 **Major**	Add cost warnings
No shorthand syntax support	🟢 **Minor**	Existing syntax works

Recommended Implementation Plan:

1. ✅ **No changes needed for basic functionality**
   - User can already use `/summarize last 60 days`

2. 🔧 **High Priority Safeguards:**
   - Add `MAX_MESSAGES_FOR_SUMMARY = 2000` limit
   - Add database requirement check
   - Add warning for >1000 messages

3. 📝 **Medium Priority Enhancements:**
   - Add `60d`, `2mo` shorthand syntax
   - Implement smart sampling for large timeframes
   - Add cost estimation warnings

4. 🎨 **Low Priority UX Improvements:**
   - Progress indicators
   - Summary statistics (X messages over Y days)
   - Confirmation prompts for expensive operations

Key Takeaway

The bot CAN already summarize 2 months of messages using `/summarize last 60 days`

BUT:
- ⚠️ Requires PostgreSQL database
- ⚠️ Bot must have been active for 2 months
- ⚠️ Limited to ~2000 messages due to AI constraints
- ⚠️ No current safeguards against excessive costs

Recommendation: Implement message count limits and warnings BEFORE promoting long timeframe features to users.