Introduction × +

You can type something here...

Browser Tab Manager

You can type something here...

Project By:

Rishita Merchant
Sanskriti Dhar
Sneha Shinde

Problem Statement:

Most of the time, while working or studying, we end up with multiple browser tabs open—many of which are no longer useful. Having too many tabs open slows down the system and makes it difficult to stay organized and focused.

Solution:

To solve this problem, we implemented the Least Recently Used (LRU) algorithm using Python’s heapq. This automatically detects and closes tabs that have been inactive for a while, helping users maintain a cleaner and faster browsing experience.

Tech Stack

Backend (FastAPI)

- Python 3
- FastAPI
- `heapq` (for priority-based LRU logic)

Frontend

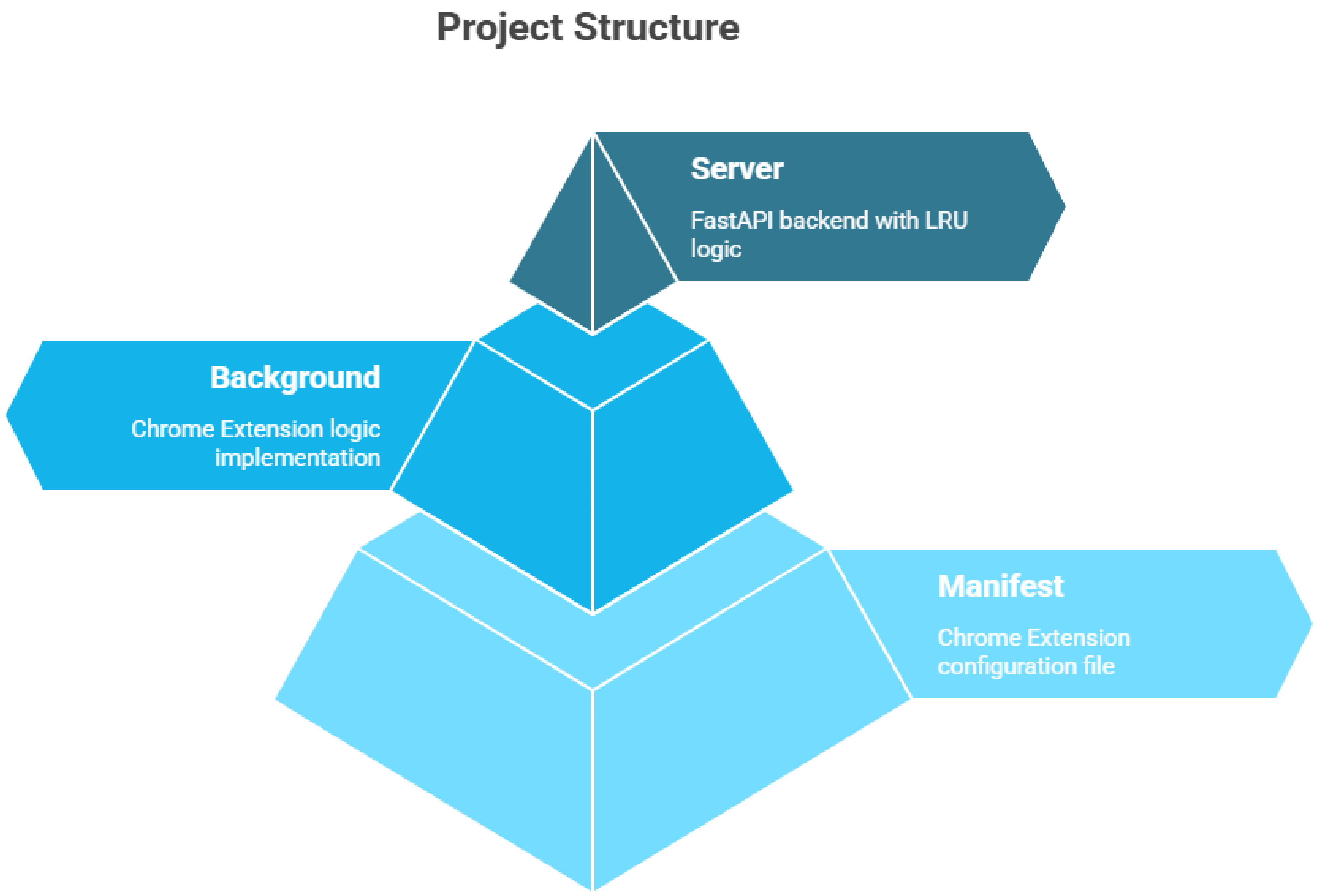
- JavaScript (background script)
- Chrome Extension Manifest V3

Communication

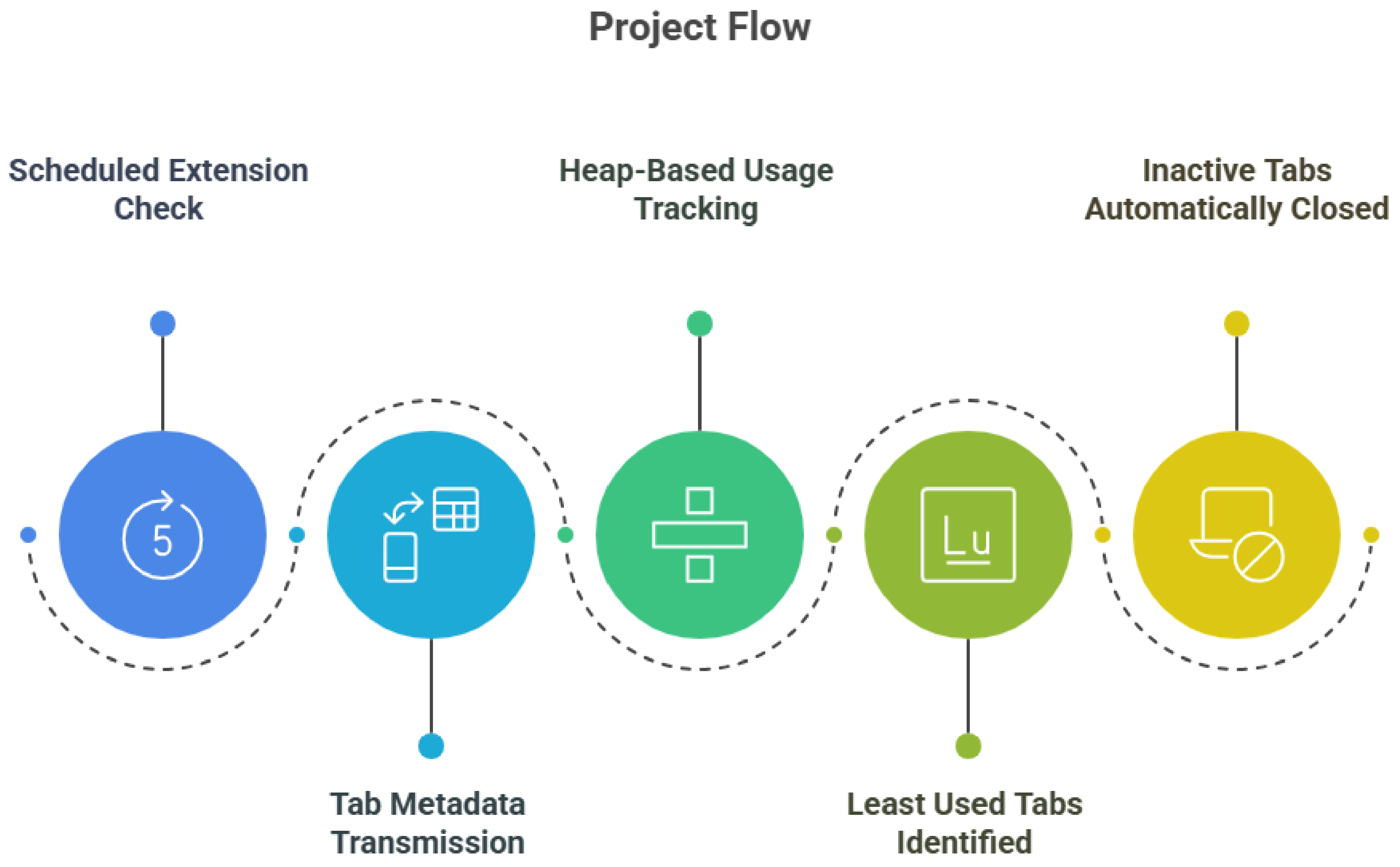
- REST API
- JSON for data transfer

Project Structure

- **Server:** A FastAPI backend that maintains a min-heap to identify least recently used tabs and responds with tabs to be closed
- **Background Script:** Executes tab tracking, API communication, and auto-close logi
- **Manifest:** Defines extension permissions and behavior


Project Structure
Server
FastAPI backend with LRU logic
Background
Chrome Extension logic implementation
Manifest
Chrome Extension configuration file

Project Flow

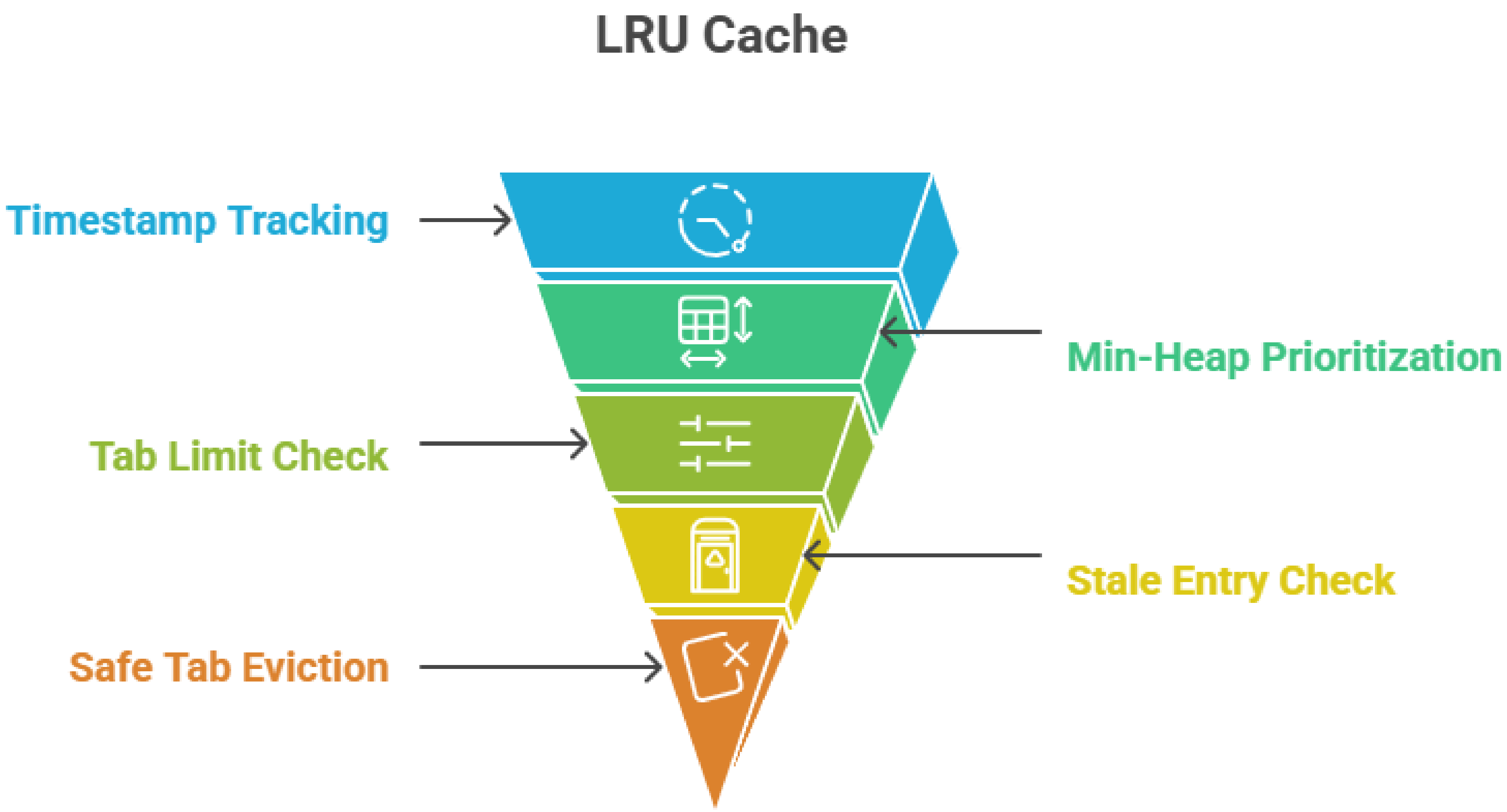


Project Flow
Scheduled Extension Check
Heap-Based Usage Tracking
Inactive Tabs Automatically Closed
5
Lu
Tab Metadata Transmission
Least Used Tabs Identified

- **Scheduled Extension Check:** Extension activates every 5 seconds to collect data
- **Tab Metadata Transmission:** Sends tab details to the FastAPI backend
- **Heap-Based Usage Tracking:** Backend stores tab info and updates a min-heap
- **Least Used Tabs Identified:** Identifies LRU tabs when the limit is exceeded
- **Inactive Tabs Automatically Closed:** Closes LRU tabs using chrome.tabs.remove

LRU Cache Logic

- **Timestamp Tracking:** Each tab update includes a timestamp (last_seen) which is stored in tab_cache and also pushed into a heap as (last_seen, tab_id)
- **Min-Heap Prioritization:** The heapq min-heap ensures the tab with the oldest usage time (i.e., least recently used) is always at the top — ready for efficient eviction
- **Tab Limit Check & Trigger:** When the number of tabs exceeds MAX_CACHE_SIZE, the backend starts popping from the heap to identify candidate tabs for eviction
- **Stale Entry Check:** Before evicting, it checks if the popped tab's last_seen matches the current value in tab_cache — this avoids removing recently updated tabs due to stale heap entries
- **Safe Tab Eviction:** If matched, the tab is deleted from tab_cache and its ID is sent to the frontend to be automatically closed via chrome.tabs.remove()



Time Complexity of LRU Algorithm

Operation	Data Structure	Time Complexity	Notes
Add/update tab entry	Dictionary (dict)	O(1)	Fast key-based update
Push to heap	Min-Heap (heapq)	O(log n)	Maintains LRU order
Pop LRU tab	Min-Heap (heapq)	O(log n)	Removes oldest tab
Validate timestamp	Dictionary (dict)	O(1)	Avoids stale eviction
Remove from cache	Dictionary (dict)	O(1)	Constant-time deletion

Conclusion

You can type something here...

Thank You

And here your subtitle.

Any Questions?