## ****Day-by-day Breakdown**** (Aug 12–18)

### ****Tue Aug 12 – Project scaffold & tooling (3h)****

**Objective:** Have a working build system & repo structure ready.

* Create a new Git repo (promptpack-extension).
* Initialize **Vite + TypeScript** workspace:
  + /extension → Chrome MV3 code (manifest, service worker, content script).
  + /packages/core → encoder/decoder logic.
  + /packages/ui → React (for later weeks).
* Install dependencies:
  + @types/chrome, vite-plugin-chrome-extension, typescript, eslint, prettier, jest or vitest.
* Write **manifest.json** with:
  + "manifest\_version": 3
  + Permissions: storage, activeTab, scripting, tabs
  + Commands: "toggleSidebar": { "suggested\_key": { "default": "Alt+P" } }
* Create service worker (src/worker.ts) with chrome.runtime.onMessage.addListener stub.

✅ **Deliverable:** pnpm run build (or npm/yarn) produces a loadable unpacked MV3 extension with no errors in Chrome.

### ****Wed Aug 13 – Message bus between SW & content (3h)****

**Objective:** Enable bidirectional messaging so UI & codec can talk.

* Add src/content.ts as a content script that runs on all LLM domains (we’ll refine later).
* In service worker, handle "TOGGLE\_SIDEBAR" command from hotkey → send a message to content script.
* In content script, log “Sidebar toggled” when receiving that message.
* Add utility: sendMessage(tabId, type, payload) for SW → content, and sendMessageToSW(type, payload) for content → SW.

✅ **Deliverable:** Pressing **Alt+P** logs "Sidebar toggled" in the page console.

### ****Thu Aug 14 – Core package setup & tokenizer skeleton (3h)****

**Objective:** Start compression logic in /packages/core.

* Create tokenize(text: string): Token[]:
  + Split on whitespace & punctuation.
  + Output tokens with {text: string, caseFlag: number}.
* Create detokenize(tokens: Token[]): string to reverse exactly.
* Add tests in /packages/core/tests/tokenizer.test.ts with tricky cases (caps, punctuation, apostrophes).
* Hook up simple CLI script pnpm run tokenize "Your text here" → prints token array.

✅ **Deliverable:** Tokenizer passes round-trip tests for simple & edge cases.

### ****Fri Aug 15 – Templating (n-gram hoisting) (3h)****

**Objective:** Reduce redundancy in prompts via reusable spans.

* Implement findTemplates(prompts: string[], nMin=6, nMax=12):
  + Tokenize all prompts.
  + Count n-grams; select top 1–3 repeated spans above threshold.
* Replace those spans in prompts with {{T0}}, {{T1}}, store template table.
* Add tests: confirm replaced spans reconstruct exactly via applyTemplates(tokens, templateTable).

✅ **Deliverable:** Templating reduces token count ≥20% on sample redundant set without breaking round-trip.

### ****Sat Aug 16 – Container format skeleton (3h)****

**Objective:** Define .prmtpck binary structure without compression yet.

* Write encodeContainer(meta, dict, index, data):
  + Magic bytes: "PCK\x02", flags=0, codec=0.
  + Length-prefixed blocks: header (CBOR), dict (empty now), index (empty array), data (raw tokens).
* Write decodeContainer(buffer):
  + Parse magic/version, extract blocks, return object.
* Test: encode + decode returns same meta/index/data.

✅ **Deliverable:** .prmtpck skeleton can be generated & parsed entirely in memory.

### ****Sun Aug 17 – End-of-week integration check (3h)****

**Objective:** Have all week’s parts running together in a dry run.

* Create a demoPrompts array (10–20 prompts).
* Tokenize → template → serialize to container → decode back.
* Log pack size in bytes vs raw UTF-8.
* Hook to content script: pressing Alt+P runs this encode/decode cycle and logs before/after size.

✅ **Deliverable:** Press Alt+P → console shows raw vs container size (still no compression yet).

## Week 1 Output Recap

By end of **Sunday Aug 17**, you’ll have:

* Loadable MV3 extension.
* Hotkey toggling a message between SW & content.
* Tokenizer & templating logic tested.
* Binary .prmtpck skeleton with encode/decode in memory.
* First “size difference” log (should see small win from templating even without compression).