SUPERCHARGE

Powered by Personal Memory Language (PML)

Product Requirements Document · v2.0

Status Draft – For Review	**Version** 2.0.0
Date February 2026	**Classification** Confidential

Table of Contents

1. Executive Summary

Supercharge is a BYOK (Bring Your Own Key) AI chat web application that gives users a persistent, compressed, LLM-agnostic memory layer — powered by the Personal Memory Language (PML) protocol. Unlike standard chat interfaces where context is lost between sessions, Supercharge continuously builds a structured knowledge graph of the user and injects it as compressed PML into every LLM call, creating the experience of a truly personal AI that remembers everything.

Key Differentiators	
Memory-First	PML context is injected into every LLM call, making any model feel personal.
BYOK	Users supply their own API key. No model lock-in. Works with GPT-4, Claude, Gemini.
Zero Lock-in	Memory exported as portable PML text. User owns their data.
Dark Minimal UI	Glassmorphic dark interface optimised for focus and extended usage.

2. Product Vision & Philosophy

2.1 The Core Problem

Every LLM conversation starts from zero. The model has no knowledge of who you are, what you care about, your ongoing projects, your preferences, or your history. Users must re-explain context every single session, creating friction that makes AI assistants feel generic and disposable.

2.2 The Solution

PML is a compressed, structured memory syntax designed to be injected into an LLM system prompt. It encodes facts, preferences, plans, relationships, and emotional context in a loss-efficient format that maximises information density within token budgets. The LLM never speaks PML to the user — it uses PML internally to hydrate its responses with contextual awareness and empathy.

2.3 Why LLM + PML works

Critically, because the consumer is a frontier LLM (GPT-4, Claude 3.5, Gemini 1.5 Pro), many gaps that would require rule engines in traditional software are filled naturally by the model. The LLM handles inference (deriving brother-in-law from rel:sister + rel:husband), contradiction detection, emotional nuance, and episodic continuity — all without additional backend logic. PML's job is to efficiently represent what the LLM needs to reason from.

3. PML Specification — Final v2.0

Personal Memory Language — the complete canonical reference for the Supercharge backend and LLM kernel.

3.1 Node Structure

Every memory is a node following this canonical syntax:

```
COMMAND #CAT:Root.Sub [Item|key:val|key:val] <GlobalKey:val> @LINK ^INHERIT

COMMAND    → STORE | UPDATE | PATCH | DELETE | RECALL | LINK | MERGE | ON | CTX
#CAT       → 2-letter category hash (see 3.2)
:Root.Sub  → Dot-notation path (unlimited depth)
[Item]     → The memory payload
|key:val   → Inline metadata (pipe-separated)
<key:val>  → Global context metadata
@Node      → Relationship link to another node
^Template  → Inherit schema from parent node
```

3.2 Category Hash Reference

Hash	Category	Use Cases
`#pf`	Preference	Likes, dislikes, tastes, content styles, UI preferences
`#ac`	Action	Past events, completed tasks, habits, logged behaviours
`#fc`	Fact	Hard data, IDs, numbers, account details, specs
`#en`	Entity	People, pets, objects — linked relationship nodes
`#lc`	Location	Places, addresses, frequently visited spots
`#pl`	Plan	Future intent, goals, scheduled reminders
`#wk`	Work	Projects, clients, deadlines, professional context
`#hl`	Health	Medical history, diet, fitness tracking, medication

`#st`	State	Current mood, energy, focus, situational status
`#ep`	Episode	Conversational events — timestamped interaction log

3.3 Operator Reference

Op	Name	Meaning
`|`	Pipe	Metadata attached to specific item
`!`	Bang	Strict negation / ban
`~`	Tilde	Uncertainty — draft or unconfirmed
`>`	Arrow	Alias or redirect to another node
`*`	Wildcard	Matches all children of a node
`@`	Link	Relationship pointer to another node (new)
`^`	Inherit	Copy schema from parent template node (new)

3.4 Metadata Keys

Key	Meaning	Type / Values	Example
`t:`	Timestamp	ISO date YYYY-MM-DD	`<t:2026-03-01>`
`c:`	Condition	Free-text trigger	`|c:only on weekends`
`f:`	Frequency	Free-text cadence	`|f:daily`
`s:`	Sentiment	++ / + / 0 / – / —	`<s:++>`
`p:`	Priority	H / M / L	`<p:H>`
`src:`	Source	user | web | inferred	`<src:user>`
`ttl:`	Expiry	ISO date	`<ttl:2026-12-01>`
`vis:`	Visibility	priv | shared | global	`<vis:priv>`
`conf:`	Confidence	Float 0.0–1.0	`<conf:0.85>`
`val:`	Typed value	n: number, d: date, a: array, b: bool	`|val:n:85`
`ctx:`	Context scope	work | personal | global	`<ctx:work>`
`rel:`	Relationship	sister / friend / client / etc	`|rel:sister`
`salience:`	Emotional weight	Float 0.0–1.0	`<salience:0.9>`

3.5 Command Set

Command	Purpose & Behaviour

STORE	Create a new memory node. Fails silently if node already exists — use UPDATE to modify.
UPDATE	Destructively overwrite an existing node. Replaces the full value. Use PATCH for time-series.
PATCH	Non-destructive append — adds a timestamped version alongside existing values. For evolving data like weight, mood, opinion.
DELETE	Permanently remove a node. The backend marks it STALE rather than hard-deleting, preserving history.
RECALL	Query the memory store. Supports WHERE, AND, OR, SINCE, UNTIL, LIMIT, SORT operators.
LINK	Create an explicit bidirectional relationship between two existing nodes.
MERGE	Combine two nodes that describe the same entity (e.g. after deduplication).
ON	Register a conditional trigger. Fired by date, entity mention, context switch, or state change.
CTX	Open or close a context scope block. Scopes isolate work / personal / global memories.

3.6 RECALL Query Syntax

```
RECALL #CAT:Path WHERE key:val
RECALL #pf:food WHERE s:--                      // all strong dislikes
RECALL #pl:* WHERE status:pending                // all pending plans
RECALL #en:person WHERE rel:sister               // family members
RECALL #wk:* WHERE p:H AND ctx:work              // urgent work tasks
RECALL #hl:fitness.weight SINCE 2026-01-01 LIMIT 5 SORT t:desc

Operators: WHERE  AND  OR  SINCE  UNTIL  LIMIT  SORT
```

3.7 Trigger Syntax (ON)

```
ON t:2026-03-01 => RECALL #wk:task.book_hotel       // date trigger
ON #st:mood [val:b:low] => RECALL #pf:comfort.*     // state trigger
ON CTX:work => RECALL #wk:project WHERE p:H         // context switch
ON #en:person [riya] MENTIONED => RECALL @en:person.riya  // entity mention
```

3.8 Full Worked Example

A multi-turn conversation showing the complete PML output format.

Turn 1 — User: "I'm Arjun, based in Mumbai. I'm a UX designer working on an app called Pulse for client NovaTech."

```
STORE #en:person [arjun|rel:self] <src:user>
STORE #lc:home [mumbai] @en:person.arjun
STORE #fc:identity [profession|val:ux_designer] @en:person.arjun
STORE #wk:project [pulse] @en:client.novatech <p:H|ctx:work>
STORE #en:client [novatech] @wk:project.pulse
```

Turn 2 — User: "I hate Comic Sans but I actually love it ironically on memes."

```
STORE #pf:design [!comic_sans|s:--]
STORE #pf:design.meme [comic_sans|c:ironic|s:++]
// Conflict resolution: .meme specific overrides general ban
```

Turn 3 — User: "My weight was 87kg last month, now I'm down to 82kg."

```
PATCH #hl:fitness [weight|val:n:87|unit:kg] <t:2026-01-22>
PATCH #hl:fitness [weight|val:n:82|unit:kg] <t:2026-02-22>
// PATCH preserves history — backend stores both as time-series
```

4. System Architecture

4.1 High-Level Stack

Technology Stack	
Frontend	React + TypeScript, Tailwind CSS, Framer Motion
Auth	Firebase Authentication (Email/Password + Google OAuth)
Database	Firestore — per-user memory.pml document + chat history
Storage	Firebase Storage — profile photos
LLM Interface	User-supplied API key (OpenAI / Anthropic / Google). Calls made client-side — keys never touch our servers.
Memory Parser	TypeScript PML parser — runs in-browser. Parses MEMORY_OP blocks, executes STORE/UPDATE/PATCH/DELETE against Firestore.
Hosting	Firebase Hosting or Vercel

4.2 Request Lifecycle

1	**User sends message** Input captured in chat UI
2	**Memory Injection** Backend fetches user's PML from Firestore and prepends to system prompt
3	**LLM API Call** Request sent to user's chosen provider with their BYOK key
4	**Response Parsing** RESPONSE block displayed to user. MEMORY_OP block silently extracted.
5	**PML Execution** Parser runs STORE/UPDATE/PATCH/DELETE commands against Firestore
6	**State Update** Memory store updated. Next message will include new context.

4.3 Memory Injection Format

The system prompt sent to the LLM follows this template:

```
[SYSTEM PROMPT]
You are a personal AI assistant with a persistent memory layer.
Below is the user's compressed PML memory. Use it to personalise every response.
Never output PML syntax to the user. At the end of every response, output a
MEMORY_OP block with any new or updated memories from this conversation.

[PML MEMORY CONTEXT]
#en:person [arjun|rel:self] @lc:home.mumbai
#fc:identity [profession|val:ux_designer]
#wk:project [pulse] @en:client.novatech <p:H>
... (full PML store)

[CONVERSATION HISTORY]
... (last N messages)
```

5. Application Overview

App Identity	
App Name	Supercharge
Tagline	The AI that actually remembers you.
Target Users	Power users, knowledge workers, individuals who rely on AI daily
Platform	Web (desktop-first, responsive mobile)
Auth Modes	Email/Password + Google Sign-In (Firebase)
LLM Support	OpenAI (GPT-4, GPT-4o), Anthropic (Claude 3.5 Sonnet), Google (Gemini 1.5 Pro)
Memory Storage	Firestore — per-user PML document, versioned
Data Ownership	Full export at any time. PML is plain text.

6. UI/UX Specification

6.1 Design Language

The visual identity is built on three core principles: dark, minimal, and spatial. The interface should feel like a focused thinking environment — not a consumer chat toy.

Design Tokens	
Background Primary	#0F0F1A — near-black with blue undertone
Background Surface	#1A1A2E — card and panel backgrounds
Glass Layer	rgba(255,255,255,0.05) + backdrop-filter: blur(20px)
Glass Border	rgba(255,255,255,0.08) — 1px
Accent Primary	#6C63FF — electric violet
Accent Secondary	#A78BFA — soft purple
Text Primary	#E2E8F0 — off-white
Text Muted	#94A3B8 — slate
Success	#10B981 — emerald
Error	#EF4444 — red
Font	Inter — 400, 500, 600 weights
Border Radius	12px cards, 8px inputs, 24px buttons, 50% avatars

Transition	all 0.2s cubic-bezier(0.4, 0, 0.2, 1)
Shadow	0 8px 32px rgba(0,0,0,0.4) for glass cards

6.2 Glassmorphism Rules

- All modals, sidebars, and floating panels use glass: rgba(26,26,46,0.7) + blur(20px) + 1px rgba(255,255,255,0.08) border
- No hard opaque backgrounds on overlays — the depth effect is part of the hierarchy
- Hover states: increase glass opacity to 0.85 and accent border to rgba(108,99,255,0.4)
- Focus rings: 2px solid #6C63FF with 0.3 opacity glow

7. Screen-by-Screen Flow

7.1 Authentication Screen

Route: /auth — First screen all unauthenticated users see.

Layout

- Full-viewport dark background with a subtle animated radial gradient (violet → transparent) behind the auth card
- Centered glass card: max-width 440px, padding 48px, blur background, 1px violet border
- Top: Supercharge wordmark in Inter 600, accent violet, with a small memory-node icon
- Subheading: 'The AI that actually remembers you.' in muted text

Components

- Google Sign-In button: full-width, white background, Google logo left, 'Continue with Google' text, 12px radius
- Divider: '— or —' in muted text
- Email input field: dark glass background, 1px border, placeholder 'your@email.com'
- Password input field: with show/hide toggle icon
- Primary CTA button: full-width, gradient from #6C63FF to #A78BFA, 'Sign In' or 'Create Account'
- Toggle link: 'Don't have an account? Sign up' — switches form between sign-in and register
- Forgot password link: small muted text below password field

States

- Loading: button shows spinner, inputs disabled
- Error: red border on affected field, error message below in red
- Success: brief success flash then redirect to /onboarding or /chat

7.2 Onboarding Screen

Route: /onboarding — Shown once to new users immediately after first auth.

Step 1 — Welcome Carousel (3 slides)

Full-screen dark layout with a centered card. Progress dots at the bottom. Left/right chevron arrows. Auto-advances every 6 seconds or on user click.

Slide	Title	Body	Illustration
1	**Supercharge remembers.**	Every preference, plan, and fact you share is stored as compressed PML — injected into every conversation.	*Animated nodes forming a knowledge graph*
2	**Your key. Your data.**	Bring your own API key from OpenAI, Anthropic, or Google. We never store it on our servers.	*Key icon → shield → lock animation*
3	**Start talking.**	Just chat naturally. Supercharge extracts memories automatically — you never write PML yourself.	*Chat bubbles with a memory bar filling up*

Step 2 — API Key Setup (below carousel, same page)

- Section heading: 'Connect your AI model'
- Provider selector: 3 pill buttons — OpenAI | Anthropic | Google. Active pill: filled accent background.
- Below pills: small text showing the model that will be used (e.g. 'GPT-4o' for OpenAI)
- API key input: password-type field with eye toggle, placeholder 'sk-...' or 'sk-ant-...' depending on provider
- Helper text: 'Your key is stored locally and never sent to our servers. Used only to make direct calls to the provider.'
- CTA button: 'Start Chatting →' — validates key format (regex), saves encrypted to Firestore, redirects to /chat
- Skip link: 'I'll add this later' — allows entry without a key, but chat will be disabled until key is added

7.3 Chat Screen

Route: /chat — Primary interface. All subsequent interaction happens here.

Top Navigation Bar

Left Zone	Center Zone	Right Zone
Hamburger icon (≡) — opens sidebar drawer	Supercharge wordmark (click to new chat)	User avatar (32px) → opens profile dropdown

Left Sidebar Drawer (Hamburger Menu)

Opens as an overlay drawer from the left. Glass background with blur. Width: 280px. Closes on outside click or pressing Escape.

Sections from top to bottom:

- New Chat — button with + icon. Creates a new conversation, clears active messages, preserves memory.
- Chat History — collapsible section. Lists past conversations by date group: Today, Yesterday, Last 7 days, Older. Each item shows conversation title (auto-generated from first message, truncated to 36 chars). Click to load. Long-press or right-click shows Rename / Delete context menu.
- Manage Chats — opens a modal with a searchable, selectable list of all conversations. Bulk delete and export as JSON options.
- API Settings — navigates to /settings/api. Allows changing provider and key. Shows current active provider badge.
- Divider
- User info at bottom: avatar + display name + email in small muted text. Sign Out button.

Profile Dropdown (Top Right Avatar)

Appears below avatar on click. Glass card. Contains: Display name, email, Edit Profile link, Memory Explorer link, Export Memory link, Sign Out.

Chat Message Area

- Scrollable container: flex-column, newest message at bottom
- Empty state: centered 'What's on your mind?' prompt with 3 suggested starter chips
- User messages: right-aligned bubble, accent gradient background, white text, 8px radius on left corners only
- AI messages: left-aligned, glass background, primary text color, 8px radius on right corners only
- Each AI message shows a faint 'memory updated' indicator (small violet dot + 'n memories saved') when a MEMORY_OP occurred
- Code blocks inside messages: dark background, monospace font, copy button top-right
- Markdown rendering: bold, italic, lists, headers supported

Input Area

- Sticky bottom of viewport: glass background, 1px top border
- Multi-line textarea: auto-expands up to 8 lines, then scrolls. Placeholder: 'Message Supercharge...'
- Right side of input: send button (arrow icon, accent color, disabled when empty or loading)
- Left side: attach icon (future — for image support) and memory icon (opens memory viewer panel)
- Below input: small muted text — 'Supercharge uses [Provider] · [n] memories active'

7.4 API Settings Screen

Route: /settings/api — Accessible from sidebar.

- Current provider displayed prominently with model name
- Provider switch: same 3-pill selector from onboarding
- API key field: shows last 4 characters masked (sk-...XXXX), click to reveal/edit
- Save button: re-validates key format, updates Firestore, shows success toast
- Danger zone section: 'Remove API key' button with confirmation dialog

7.5 Memory Explorer

Accessible from profile dropdown or memory icon in chat input area.

- Side panel (not full-page): 400px wide, glass background, slides in from right
- Header: 'Your Memory' with node count badge and Export button
- Category tabs: All | Preferences | People | Plans | Work | Health | ...
- Each memory displayed as a compact card: category badge, node path, value, timestamp
- Edit icon on each card: opens inline edit for manual corrections
- Delete icon: removes node with undo toast
- Search bar at top: filters nodes by keyword

8. Firebase & Authentication

Firebase Services	
Auth	Email/Password + Google provider. Custom claims for plan tier (future).
Firestore	users/{uid}/memory — PML store as structured document. users/{uid}/chats/{chatId} — message history.
Storage	users/{uid}/avatar — profile photo upload.
Hosting	Static frontend deployment with Firebase Hosting or Vercel.

8.1 Firestore Data Model

```
users/
  {uid}/
    profile: { displayName, email, photoURL, createdAt }
    settings: { provider, apiKeyEncrypted, model, timezone }
    memory: { pml: string, nodeCount: number, lastUpdated: timestamp }
    chats/
      {chatId}/
        title: string
        createdAt: timestamp
        updatedAt: timestamp
        messages: [{ role, content, timestamp, memoryOp? }]
```

8.2 API Key Security

- API keys are AES-256 encrypted client-side before being written to Firestore
- Decryption key is derived from the user's auth token — Anthropic/Supercharge servers cannot read keys
- Keys are never logged, never transmitted to backend functions
- LLM calls are made directly from the browser to the provider API

9. BYOK API Management

Supported Providers	
OpenAI	Models: gpt-4o, gpt-4-turbo, gpt-3.5-turbo. Key format: sk-...
Anthropic	Models: claude-3-5-sonnet-20241022, claude-3-opus-20240229. Key format: sk-ant-...
Google	Models: gemini-1.5-pro, gemini-1.5-flash. Key format: AIza...

9.1 Provider Abstraction Layer

A unified LLMClient interface normalises the different APIs so the PML kernel works identically regardless of the provider. The interface handles message formatting, system prompt injection, response streaming, and MEMORY_OP block extraction.

9.2 Model Selection

Users can select their preferred model within the provider's available options. The recommended defaults are GPT-4o for OpenAI, Claude 3.5 Sonnet for Anthropic, and Gemini 1.5 Pro for Google — all of which have sufficient context window size to hold a full PML memory store.

10. LLM Integration & PML Kernel

10.1 System Prompt Construction

On every message, the frontend constructs the full system prompt by: (1) fetching the user's PML from Firestore, (2) fetching the last N messages from the current chat, (3) concatenating the PML kernel instructions + PML memory + conversation history into the system prompt, and (4) appending the user's new message.

10.2 MEMORY_OP Parsing

After each LLM response, the parser scans for the MEMORY_OP block delimited by triple backticks. It extracts each command line, routes to the appropriate handler (STORE, UPDATE, PATCH, DELETE), validates PML syntax, and writes changes to Firestore. Invalid commands are logged but do not throw to the user.

10.3 Context Window Management

PML is designed to be token-efficient. A full user memory store typically occupies 800–2000 tokens. For models with limited context, a priority-ranked subset is injected: high-salience nodes and recently-accessed nodes are always included; low-priority old nodes are summarised or omitted with a count indicator.

11. Security & Privacy

- API keys: encrypted client-side, never readable by Supercharge infrastructure
- Memory data: stored in user's own Firestore document, accessible only via their auth token
- No telemetry: user conversation content is not logged by Supercharge
- Export: full PML export as .txt available at any time from profile dropdown
- Deletion: account deletion cascade-deletes all Firestore documents including memory and chat history
- vis:priv tagged nodes: never displayed in the Memory Explorer UI, only injected into system prompt
- HTTPS only: all Firestore reads/writes and API calls over TLS

12. Non-Functional Requirements

Requirement	Specification
Performance	First Contentful Paint < 1.5s. Chat input → first token display < 2s (network-dependent). PML parser < 50ms per MEMORY_OP block.
Responsiveness	Fully functional on 375px+ viewport. Sidebar collapses to drawer on mobile. Touch targets 44px+.
Reliability	Firestore offline persistence enabled. Failed MEMORY_OP commands queued and retried on reconnect.
Accessibility	WCAG 2.1 AA. Keyboard-navigable sidebar and chat. ARIA labels on all interactive elements. Focus-visible rings.
Browser Support	Chrome 100+, Safari 15+, Firefox 110+, Edge 100+.
Data Limits	Max PML store size: 50,000 tokens (approx. 200KB). Warn at 80%. Chat history: unlimited, paginated in batches of 50.

13. Versioning & Roadmap

PML Version Roadmap

Version	Status	Features Included
v1.0	Shipped (MVP)	Categories, operators (|!~>*), metadata keys (t/c/f/s/p/src/ttl), STORE/UPDATE/DELETE commands, basic sentiment, two output blocks
v2.0	**This Document**	Typed values (n/d/a/b), @ link operator, ^ inherit, PATCH command, RECALL query syntax with WHERE/SINCE/LIMIT/SORT, ON triggers, CTX scoping, conf: key, salience: key, vis:priv, #ep episode category, rel: key
v2.5	Next — Q2 2026	MERGE command, multi-agent shared CTX, bulk import from plain text / resume / notes, memory health score, conflict surface to user
v3.0	Future — Q4 2026	Inference rules engine (optional, for non-LLM backends), memory graph visualisation, semantic search over PML store, inter-user memory sharing with permission model

Product Roadmap

Phase	Deliverables
Phase 1 — MVP	Auth + onboarding, BYOK key setup, chat interface, PML v2.0 kernel, basic Firestore memory read/write
Phase 2 — Polish	Memory Explorer panel, chat history management, streaming responses, mobile optimisation, onboarding carousel
Phase 3 — Power	Memory health metrics, export/import, provider auto-detect from key format, keyboard shortcuts, PWA offline mode
Phase 4 — Scale	Team/shared contexts, memory analytics dashboard, plugin system for custom #CAT extensions, API for third-party integrations