Use SvelteKit (not Astro) with plain JavaScript (not TypeScript) to rebuild the Lazer Kiwi Dota 2 guild website. Preserve the modular structure, visual style, and all functionality from the current stable site. Avoid introducing unnecessary complexity. Keep all logic clear, maintainable, and well-commented to ensure reliability between GPT sessions. Every change must be followed by documentation updates.

⚙️ FRAMEWORK & STRUCTURE

• Use SvelteKit with file-based routing to replace the old iframe architecture.

• Each previous HTML tab becomes a top-level SvelteKit route:

/ → Home

/turbo-graphs → Turbo Rank Graphs

/party-estimator → Party Rank Estimator

/weekly-leaderboard → Weekly Guild Point Leaderboard

/weekly-graphs → Weekly Point Comparison

/guild-rank-list → Guild Turbo Rank List

• Shared elements like the header should go in +layout.svelte.

• Use plain JavaScript throughout. Do not enable or write TypeScript.

🎨 VISUAL STYLE

• Keep the black background, neon green text #00FF00, and burnt orange accents #ab6e11.

• Font: 'Orbitron', sans-serif (via Google Fonts).

• Maintain glowing borders, hover effects, and blinking animation as in the original CSS.

• Replicate the layout, spacing, and hover behavior of the current site exactly unless told otherwise.

• Ensure mobile responsiveness matches or improves on the original.

📊 FUNCTIONALITY TO RETAIN

Preserve all features from the current site. Reimplement them in the new framework with identical functionality and style.

Home Page:

Static welcome message.

Discord button linking to invite.

Visit counter (use same Google Apps Script endpoint and dev-mode logic).

News feed (auto-fetch and display sorted news from JSON endpoint, styled scrollbox).

AI Kiwi image gallery (from home\_gallery\_assets/, maintain layout and hover zoom).

Turbo Rank Graphs:

Dropdown to select player.

Chart.js line graph showing turbo MMR over time.

Use same Google endpoint and rank conversion logic.

Destroy chart before redrawing to prevent duplication.

Party Rank Estimator:

Five dropdowns for selecting players.

“Estimate Rank” button.

Compute latest known rank per player and show average using rank map (e.g., “Legend 2 Star”).

Requires at least one player to be selected.

Weekly Guild Point Leaderboard:

Dropdown to select week.

Table showing weekly gains, sortable columns.

Toggle button for Total and 4-Week Avg columns (showExtraCols logic).

Color coding: top 3 (purple), >2000 (green), 1000–1999 (orange), <1000 (red).

Preserve tooltip, sort state, and legend.

Maintain feature toggle logic using a Boolean state variable.

Weekly Point Earning Graphs:

Two dropdowns for member selection.

Line graph comparing point progression over time.

Auto-refresh every 5 minutes.

Preserve color logic (cyan + pink), axis style, and labels.

Guild Estimated Turbo Rank List:

Table showing member nickname, rank, position.

Header includes latest date.

Use rankNumberToLabel to format rank numbers.

Maintain display logic and explanatory paragraph at top.

🧩 COMPONENTS & UTILITIES

• Shared UI (e.g. header, button styles) should go in components or layout files.

• Put constants (rank names, API URLs, color thresholds) in a single lib/constants.js module.

• Add utility modules only when logic is reused or exceeds ~30 lines.

📁 FILE & FOLDER ORGANIZATION

• Route pages go in src/routes/

• Shared components: src/lib/components/

• Static assets: static/ (e.g. static/home\_gallery\_assets/)

• Keep filenames and IDs consistent with the original where possible (e.g. news-container, visit-counter, etc.).

🚨 DEVELOPMENT SAFEGUARDS

• Do not rename or remove feature flags (isDev, isLocal, showExtraCols).

• Do not change Google Script endpoints without instruction.

• Do not restructure files arbitrarily. Any change in file structure must be explained and documented.

• Always test chart rendering and table sorting before finalizing changes.

📝 COMMENTING STRATEGY

• Begin each Svelte file with a brief description of its purpose.

• Use section comments (// Fetch news, // Handle chart redraw) to divide logic clearly.

• Inline comment any non-obvious behavior (e.g. rank conversions, endpoint fallback logic).

• Keep comments in sync with the code. Delete or update outdated comments immediately.

• Mark TODOs and partial implementations clearly for future follow-up.

🧠 GPT-OPTIMIZED CODE PRACTICES

• Write readable, modular code using meaningful names.

• Avoid unnecessary abstraction or nested complexity.

• Group related constants and state near top of file.

• Use Svelte’s reactivity features properly (e.g. $: for derived values, onMount for async fetch).

• Assume the user is non-technical: favor clarity over cleverness.

• When fixing a bug or adding a feature, first isolate what changed and why. Then apply it cleanly and explain the update in a comment.

• Do not change how things look unless the user asks.

🛠 DEPLOYMENT

• Keep project compatible with GitHub + Netlify static deployment.

• Use the SvelteKit Netlify adapter (GPT can add this if not present).

• All dynamic data is client-side fetched from public Google Apps Script endpoints—no server logic needed.

• Ensure build succeeds using npm run build. GPT should simulate this before major pushes.

📚 DOCUMENTATION & TRACKING

Every feature, fix, or refactor must be followed by:

Updating the README.md

Adding a version entry to the changelog

Adding or adjusting relevant comments

README must include:

• Project purpose and description

• Tech stack overview

• Folder and file structure

• Setup and deployment instructions

• GPT usage guidelines (how user should direct GPT for changes)

• Change log of all project updates with dates and versions

🧾 CHANGELOG FORMAT

• Use semantic versioning (e.g. 0.1.0, 0.1.1).

• Each version must include:

Date

Summary of added/changed features

Notes on visual, logical, or file structure changes

• Changelog can be part of README.md or separate CHANGELOG.md

🧪 TESTING REQUIREMENTS

Before considering any page, chart, or feature complete:

• All buttons, dropdowns, and graphs must render and respond correctly

• No console errors in dev mode

• Data must load from correct endpoint

• Visual layout must match or improve upon original

✅ FINAL RULE

If unsure about a change, assume user wants functionality and visuals preserved. Never remove features, modify endpoints, or introduce libraries without permission. When in doubt: preserve, comment, and document.