**Human Design System Master Reference – 5 Oct 2025**

**🧭 Recent Updates — October 2025 (Summary)**

**1) Node & Mandala Calibration**

* Mandala anchor corrected: Gate 41 aligned at 302° (≈ 2° Aquarius).
* Gate/line mapping uses one aligned order with epsilon-safe boundaries (no line=7 rollover).
* Swiss Ephemeris nodes: compute Mean and True; default set to TRUE for closer alignment near boundaries.
* North/South nodes validated: opposite gates, same line (e.g., 64.5 ↔ 63.5; 6.2 ↔ 36.2).

**2) Ephemeris → Gate.Line Mapping (cli.mjs)**

* Helper `gl(g, l)` ensures safe formatting (prevents stray '.' or missing parts).
* Solar-arc design date: iterative 88° Sun offset with guardrails.
* Sun/Earth harmonization: Earth line forced to Sun line after gate calculation.
* Longitude normalization: robust modulo handling (0–360).

**3) Debugging Aids**

* Lightweight `debugLog(...args)` → writes to `debug.txt` (no timestamps).
* Mandala sanity check: sample gates from the anchor to verify sequence (41 → 19 → 13 → 49 → 30 → 55…).
* Node diagnostics: logs both Mean/True node longitudes and the mapped gate.line for P/D.

**4) Bodygraph SVG Refinements**

* Dual-color overlay for gates present in both personality and design: solid red base + black dashed overlay.
* Integration network (10–57–20–34) drawn with split overlays and junction points.
* Center fill theme updated (`CENTER\_THEME\_ANN`): open / design / personality / both.
* Gate label positioning: nudges consolidated; labels adapt to center status for contrast.

**5) Minimal Web Form (Charts → PDF)**

* Flat, minimal layout: Name (line 1), Date+Time (line 2), Location (line 3), Submit (line 4).
* Date input mask `dd / mm / yyyy` with hidden ISO mirror for backend (`date\_iso`).
* Time field kept simple (`HH:MM`); date auto-advances between segments.
* Google Places Autocomplete for city search; captures place\_id, formatted\_address, lat/lng.
* PDF launcher: preview opens in a new tab; download remains same-tab.
* Submit button centered for better mobile presentation.

**——— Consolidated Reference ———**

**Primary Reference (from: HD\_Backup\_UPDATED.docx)**

HD App Backup & Restore Guide (Updated)

This document contains the updated backup instructions for the Human Design Rails app. It includes the form changes (Google Places Autocomplete), the Incarnation Cross CSV lookup, and file structure so the app can be restored or migrated to a new laptop.

1. Project Structure

Key files and folders:  
- app/controllers/charts\_controller.rb (PDF generation, form handling)  
- app/services/incarnation\_cross\_index.rb (CSV lookup for Incarnation Cross)  
- app/views/charts/new.html.erb (input form with Google Places)  
- lib/hdkit/cli.mjs (Node chart generator)  
- lib/hdkit/data/Incarnation\_Cross\_List.csv (CSV with cross data)  
- config/routes.rb (routes to forms and PDF downloads)  
- Gemfile (prawn, wicked\_pdf, prawn-svg, etc.)  
- app/assets/fonts/DejaVuSans.ttf + DejaVuSans-Bold.ttf

2. Google Places Autocomplete Form

The form for chart input now uses Google Places Autocomplete. This ensures correct latitude/longitude + timezone lookups.  
  
File: app/views/charts/new.html.erb  
- Inputs: name, date, time, location  
- Hidden fields: place\_id, place\_text  
- JS loads Places API and fills hidden fields  
- Controller validates that a suggestion was chosen (place\_id not blank).

3. Charts Controller

File: app/controllers/charts\_controller.rb  
  
Responsibilities:  
- new: renders form  
- create: validates input, requires Google Place suggestion, redirects to download\_prawn  
- download: legacy WickedPDF renderer  
- download\_prawn: main Prawn PDF generator  
  
Changes:  
- Validates that place\_id is present  
- Extracts Incarnation Cross via IncarnationCrossIndex  
- Displays Cross description + signature in PDF

4. Incarnation Cross CSV Lookup

File: app/services/incarnation\_cross\_index.rb  
  
Responsibilities:  
- Loads CSV lib/hdkit/data/Incarnation\_Cross\_List.csv  
- Each row has: angle, g1, g2, g3, g4, description  
- Supports angle codes: R → Right, L → Left, J → Juxtaposition  
- Matching requires exact angle + gate set  
  
Usage in controller:  
row = IncarnationCrossIndex.find(angle: angle\_code, gates: [psun, pearth, dsun, dearth])  
  
If found → show row.description + signature (gates grouped as g1/g2 | g3/g4).

5. Fonts & Assets

Prawn requires Unicode fonts for planet glyphs. DejaVuSans.ttf and DejaVuSans-Bold.ttf are included under app/assets/fonts/ and loaded in charts\_controller.rb before drawing planet tables.

6. CSV Data

File: lib/hdkit/data/Incarnation\_Cross\_List.csv  
- Must contain headers: angle,g1,g2,g3,g4,description  
- angle values: Right, Left, Juxtaposition  
- g1–g4 are integer gate numbers  
- description is free text

7. Restore Instructions

Steps to restore on a new laptop:  
1. Clone repo and bundle install (ensure Ruby + Node installed)  
2. Add Google Places API key in layout/application.html.erb or JS pack  
3. Ensure lib/hdkit/cli.mjs runs with Node (adjust path if needed)  
4. Verify CSV exists in lib/hdkit/data/  
5. Fonts: place DejaVuSans.ttf and DejaVuSans-Bold.ttf in app/assets/fonts/  
6. Run rails server and test /charts/new → input form → generate PDF

**Archived Pre-Update Notes (from: HD Backup.docx)**

Overview

The project is a Rails application that generates Human Design charts in PDF format.  
It works in two parts:

**Rails (charts\_controller.rb)**

Collects accurate Human Design data (gates, lines, profile, cross, etc.)

Passes that data to Node for drawing.

Renders the final PDF with prawn.

**Node (cli.mjs + bodygraph-data.js)**

Receives the data from Rails.

Uses svgBodygraph to draw the bodygraph as SVG.

Returns the SVG string back to Rails.

So: Rails = data + PDF, Node = drawing.

2. Required Software

On a new Mac laptop, install:

Homebrew (package manager)  
If not already installed:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Ruby + Rails

Install Ruby (via rbenv or rvm if you prefer):

brew install ruby

Install Rails:

gem install rails

Node.js (required for cli.mjs)

Install Node with Homebrew:

brew install node

Check version:

node -v

Prawn (PDF library for Ruby)  
Add to your Gemfile (or install directly):

gem install prawn prawn-table prawn-svg

Git (to pull or back up your code)

brew install git

3. Step-by-Step Setup on a New Laptop

Step 1: Clone/restore your Rails project

If backed up with Git:

git clone <your-repo-url>

cd hdkit\_sample\_app

Or copy the project folder from backup.

Step 2: Install Ruby dependencies

Inside the Rails project:

bundle install

Step 3: Install Node dependencies

If your project uses npm/yarn for front-end:

npm install

# or

yarn install

Step 4: Verify cli.mjs

cli.mjs lives in:

lib/hdkit/cli.mjs

It should support the --render-from-json flag for rendering the bodygraph.

You can test it directly:

node lib/hdkit/cli.mjs --render-from-json='{"designPlanets":[["Sun","52.1"]],"personalityPlanets":[["Sun","46.5"]]}'

It should return a JSON string with an "svg" key.

Step 5: Verify Rails PDF endpoint

Start Rails server:

rails server

Visit in your browser:

http://localhost:3000/charts/download\_prawn.pdf?name=Ann&date=1960-06-22&time=08:30&place=Galway

It should generate a PDF.

4. Files to Backup

You need to keep safe copies of the following:

Rails controller

app/controllers/charts\_controller.rb  
(contains the logic for PDF output)

Node renderer

lib/hdkit/cli.mjs  
(draws the SVG bodygraph)

lib/hdkit/bodygraph-data.js  
(static data for channels, gates, centers)

Any Rails views/templates you customized for charts (if any).

Assets

Background image:  
lib/hdkit/types-5.png

Gemfile + Gemfile.lock  
Ensures your Ruby gems can be restored with bundle install.

package.json / yarn.lock (if front-end dependencies are used).

Your database (optional)  
If you’re storing charts in Postgres/SQLite, keep a backup of db/ and migrations.

5. Everyday Workflow

Update charts\_controller.rb when you want to change data or layout in the PDF.

Update cli.mjs or bodygraph-data.js when you want to change drawing logic of the bodygraph.

Run tests by generating a chart PDF in the browser.

Keep changes in Git for easy recovery.

✅ Summary

Install Ruby, Rails, Node, Prawn.

Restore your Rails project + cli.mjs + bodygraph-data.js.

Use Rails for data + PDF, Node for drawing.

Backup the files listed above.

Top of Form

Totally—here’s the logic flow, step-by-step, with **which file does what** (no code):

**User opens the form**

**Route:** GET /charts/new

**Controller:** ChartsController#new

**View rendered:** app/views/charts/new.html.erb

What happens: the page shows your form (with the Google Places autocomplete).

**User submits the form**

**Route:** POST /charts

**Controller:** ChartsController#create

What happens: you validate the inputs (name, date, a chosen place).

If valid: you **redirect** to the PDF endpoint with the cleaned params.

**PDF endpoint is called**

**Route:** GET /charts\_prawn.pdf

**Controller action:** ChartsController#download\_prawn

What happens (high level):

Calls your Node CLI to generate chart data (bodygraph SVG, planets, etc.).

Builds a Ruby chart hash from that output (type, profile, planets, cross text…).

Derives the **angle** and the **four gates** (P Sun/Earth + D Sun/Earth).

**Incarnation Cross lookup**

**Service file:** app/services/incarnation\_cross\_index.rb

**Data file (CSV):** lib/hdkit/data/incarnation\_crosses.csv

What happens:

The controller **calls the service** with angle + 4 gates.

The service **lazy-loads** the CSV the first time it’s asked (reads once, caches in memory).

It builds a fast **index** keyed by "<angle>|<sorted gates>".

It returns the **matching row** (description + the four gates as stored).

**PDF composition**

**Where:** still inside ChartsController#download\_prawn

What happens:

Prawn builds the layout (header, facts table, SVG, etc.).

Inserts the **Incarnation Cross block** (your description + the signature like (g1/g2 | g3/g4)).

Finalizes the PDF bytes.

**Response back to the browser**

**Where:** ChartsController#download\_prawn

What happens: Rails streams the PDF (send\_data) inline (or as a download if you asked).

**One-time setup pieces (just for clarity)**

**CSV lives at:** lib/hdkit/data/incarnation\_crosses.csv (reference data; not served to the browser).

**Service lives at:** app/services/incarnation\_cross\_index.rb (encapsulates reading/indexing).

**Controller owns:** input validation (create), chart generation + PDF assembly (download\_prawn).

**View owns:** the form (new.html.erb) + Google Places field; nothing PDF-related.

That’s the whole sequence: **Form → Create (validate/redirect) → Download Prawn (chart + lookup) → Service (CSV) → PDF → Browser.**

Top of Form

Ok. With 4. c

Bottom of Form

ChatGPT can make mistakes. Check important info. See Cookie Preferences.

Bottom of Form

Prepared for Ann — Human Design System Master Reference (auto-merged)