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:

1. Homebrew (package manager)  
   If not already installed:
2. /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
3. Ruby + Rails
   * Install Ruby (via rbenv or rvm if you prefer):
   * brew install ruby
   * Install Rails:
   * gem install rails
4. Node.js (required for cli.mjs)
   * Install Node with Homebrew:
   * brew install node
   * Check version:
   * node -v
5. Prawn (PDF library for Ruby)  
   Add to your Gemfile (or install directly):
6. gem install prawn prawn-table prawn-svg
7. Git (to pull or back up your code)
8. 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:

1. Rails controller
   * app/controllers/charts\_controller.rb  
     (contains the logic for PDF output)
2. Node renderer
   * lib/hdkit/cli.mjs  
     (draws the SVG bodygraph)
   * lib/hdkit/bodygraph-data.js  
     (static data for channels, gates, centers)
3. Any Rails views/templates you customized for charts (if any).
4. Assets
   * Background image:  
     lib/hdkit/types-5.png
5. Gemfile + Gemfile.lock  
   Ensures your Ruby gems can be restored with bundle install.
6. package.json / yarn.lock (if front-end dependencies are used).
7. 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):

1. **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).
2. **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.
3. **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).
4. **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).
5. **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.
6. **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