Powertown Message

Hi Hamza, before you begin, we want to provide some important context. Reaching this stage reflects our strong belief in your capabilities and your potential to play a meaningful long-term role at Powertown. We are building a company around exceptional individuals who can set the standard for engineering excellence and help shape the culture and technical direction of the team for years to come.

For this role, we are evaluating not only strong full-stack software engineering ability, but also the mindset of someone who could set the bar for future hires and eventually lead and mentor other developers. We maintain high standards because the systems we are building are foundational to the company's future.

At Powertown, we give out hackathon challenges only to highly qualified candidates. It is a genuine honor to be invited to this stage — you are very much on our radar. The challenge you are about to work on is not theoretical; it represents a critical real-world problem that you would directly help solve if you join us. A thoughtful, well-architected response will demonstrate the kind of foundation that builds long-term success within our engineering team.

You are here because we see you as a very strong fit, and we are excited to see how you approach this.

— Elson, Elio, Yanchen, and Minchen

Stage 3 – Hackathon Assignment

Automated BESS CAD Drawing Generation (Learning + UX + Backend)

Problem Statement

Engineering labor does not scale. In BESS interconnection and installation plan sets, many CAD pages are highly repetitive, yet they are redrawn for every project. We want to reduce repetitive drafting time by creating a system that learns common patterns from prior drawing sets, captures site-specific inputs, and produces a complete project-specific plan set that remains editable in standard CAD tools.

Objective Overview

Design and implement a system that can:

1. Ingest a building/site plan CAD file and allow a user to place BESS equipment on it.
2. Provide a single-line diagram (SLD) editor that allows a user to construct an electrical one-line from predefined symbols.
3. Use those inputs to generate a completed CAD plan set, where:

- Specific pages are auto-generated from the inputs
- Remaining pages are fixed/templates
- Output is openable and editable in standard CAD software (e.g., Autodesk)

We care more about architecture, learning logic, and practicality than perfect drafting polish.

Time Expectation

- Designed as a 3–5 day hackathon.
- You will have one full week to complete it, given your schedule.
- In true hackathon fashion, we will send you a $20 DoorDash gift card to cover a

meal while you are working. Email **elio@powertownusa.com** and he will take care of it.

Required Drawing Types (Conceptual Coverage)

Your generation approach should address, at minimum, the following drawing categories:

- Site plan
- One-line electrical diagram (SLD)
- Equipment layout
- Battery installation plan
- Interconnection details

(The actual plan-set pages to automate are specified below.)

Input + UX Requirements (Front End)

Interface A — Site Plan + BESS Placement + Cable Path

Provide a front end where a user can:

- Upload a building/site plan CAD file (DWG/DXF acceptable)
- Render the plan content in a 2D view
- Place ("drop") the battery enclosure / BESS block at any location on the plan
- Draw a cable routing path connecting the BESS to the building electrical point of interconnection
- Support basic editing operations: add / move / delete / redraw

Interface B — Single-Line Diagram Builder

Provide a front end where a user can:

- Start from a blank canvas
- Drag-and-drop predefined electrical symbols (examples: meter, distribution panel, load, switchgear, etc.)
- Connect symbols with lines to form the one-line
- Edit the canvas: add / move / delete / reconnect elements

Symbol requirement: Use the icon set consistent with the "Century Tywood – Single Line Diagram" reference (page 24) as the symbol vocabulary for your SLD palette.

Generation Requirement (Plan Set Output)

Automation vs Fixed Pages

After the two interfaces are completed, your system should generate a completed plan set where:

- Pages to be auto-generated (from inputs):
  1, 2, 4, 5, 6, 7, 12, 13, 16, 17, 24, 25, 42, 43
- All other pages: fixed (template/static) content, but please also generate them in the final output file

Output Format

The generated CAD must be:

- Openable and editable in standard CAD software (e.g., Autodesk)
- Exported as DWG and/or DXF (at least one is required; both is a plus)

Challenge Flow

1. Train or infer patterns from sample BESS CAD drawing sets (or implement a rules-based learning approach that can generalize).
2. Identify:

- Repetitive elements (title blocks, notes, typical callouts, standard blocks)
- Site-specific constraints (equipment placement, cable path, interconnection location, labels)

3. Given a new site plan + user interactions (placement/path + SLD), generate:

- A complete plan set **OR**
- A structured intermediate representation that is ready for CAD export

Key Bottlenecks (Expected)

We expect you to encounter several real-world constraints:

- CAD file formats and libraries (DWG/DXF)
- Geometric constraints (site plan) vs symbolic diagrams (SLD)
- Drafting conventions and standardization (title blocks, annotation, layer standards)
- Maintaining editability while generating programmatically

Again: we care more about system design decisions and learning logic than perfectly formatted drawings.

What to Submit

Please provide three items:

1. A GitHub repository with setup and run instructions
2. A working demo (deployed app, local run, or a 5–10 minute screen recording)
3. A short design write-up in the README addressing:

- How you parse/render input CAD
- How the placement + cable routing data is represented
- How the SLD editor stores symbols, connections, and metadata
- How you generate the automated pages vs fixed pages
- How you export editable CAD
- What you would build next with more time

Evaluation Criteria

We will evaluate the project based on:

- Usability of the two front-end interfaces
- Quality of the intermediate representation (clear, extensible, CAD-ready)
- Editability and correctness of exported CAD output
- Quality and robustness of the automation for the required pages
- Engineering judgment, clarity, and communication
- Practicality: whether this could reduce repetitive drafting effort in real workflows