HVE Core challenge

From product intent to reviewed code

What HVE is

Hypervelocity Engineering is a structured engineering framework for GitHub Copilot.

It gives agent work defined roles, standards, human review gates, and saved phase artifacts.

Four component types

Component	Job
Agents	Own roles and multi-step protocols
Prompts	Start repeatable workflows
Instructions	Apply standards by file pattern
Tracking artifacts	Carry state between phases

HVE uses files as interfaces

```
Research artifact
  → Plan + critique
  → Changes log
  → Review findings and decision
```

Reviewers can inspect and diff each handoff, then resume work from it.

Research before code

Research answers:

- Which files own the behavior?
- What constraints already exist?
- Which alternatives were rejected?
- What check could disprove the approach?

Output: `.copilot-tracking/research/`

Plan before implementation

Plan maps criteria to files, checks, dependencies, and human gates.

Output:

```
.copilot-tracking/plans/
```

Implement and Review

Implement follows the approved plan and writes a changes log.

Review checks whether the result meets the request, code quality standards, tests, and instructions.

The human still decides: approve, request changes, or pause.

Direct and combined RPI

Mode	Entry points	Use it when
Direct prompts	`/rpi-research`, `/rpi-plan`, `/rpi-implement`, `/rpi-review`	Learning, demos, explicit approvals
Combined	`/rpi task="..."`	The team understands and trusts the handoffs

Combined RPI uses Follow-up after Review to route defects or missing evidence.

HVE customization is additive

```
.github/copilot-instructions.md
.github/instructions/*.instructions.md
.github/agents/*.agent.md
```

Project context reaches all agents. Coding practices activate by `applyTo` glob.
Custom agents add repository-owned roles and output contracts.

Show that instructions were used

Creating an instruction file does not show that HVE used it.

Show:

1. a matching path;
2. the rule that activated;
3. the plan, code, or review decision it changed.

HVE starts before engineering

```
Business idea
  → BRD Builder / Product Manager Advisor / PRD Builder
  → Agile Coach
  → RPI
  → reviewed code
```

Product agents test value and assumptions. Agile Coach turns intent into an outcome-oriented story and acceptance criteria.

Challenge sequence

1. Run direct Research and Plan prompts.
2. Add repository context, coding practices, and a custom reviewer.
3. Use Product Manager Advisor and Agile Coach.
4. Update the plan, then run Implement and Review.
5. Compare the result with combined `/rpi`.

Build the endpoint

Deliver `POST /api/decisions` within the supplied product intent and API contract.

Scope control is part of the score.

The contract

Input	Rule
`title`	Required non-empty string; duplicate check ignores case
`context`	Required non-empty string
`decision`	Required non-empty string
`status`	`proposed`, `accepted`, or `superseded`

Generate an ID and UTC timestamp. Keep state in memory.

Package tested source in a non-root runtime image with production dependencies and a health check.

Fast checks that find weak work

1. Send a required field as whitespace or a number.
2. Create the same trimmed title with different case.
3. Send malformed JSON.
4. Inspect the exact error envelope.
5. Check which stage supplies source to the runtime image.

These checks test invalid inputs and container build boundaries.

Scoring

Area	Weight	Grade from
HVE RPI artifact chain	40%	Four HVE phase artifacts
HVE customization and activation	25%	Repository files and cited activation
Product and Agile agent handoff	20%	Platform output cited by Plan
Feature and container validation	15%	Output cited by Implement or Review

A manual implementation does not complete an HVE challenge.

Keep the handoff short

`challenge-handoff.md` contains only:

- links or paths to the four HVE artifacts;
- the final human decision;
- unresolved risks and the next action.

Do not copy phase narratives, criteria, command output, or proposed Git text into the handoff.