



SOFTWARE DEVELOPMENT LIFE CYCLE

1) PLANNING AND REQUIREMENT ENGINEERING

PROBLEM:

Current hackathon evaluations rely heavily on superficial metrics like "lines of code", "number of commits", or "no. of issues opened", which often rewards the loudest voices and bulk library imports over genuine technical depth and actual improvement of the candidate. This lack of visibility into individual contributions leads to unfair credit distributions,

team disperses, and difficulty for judges to distinguish between custom built logic and pre-existing code.

By failing to bridge the gap between raw Git data and high level impact, the current hackathon systems overlooks the core architectural effects and complete problem solving that define a winning project.

PRODUCT DESCRIPTION

"Hacker Introspector" is a platform where hackers and judges of MLH hackathons can go to to learn more about a project that is not visible to the naked eye.

Hackathons are won and lost on the surface - but the teeth is in the code.

Hacker introspector is an automated audit engine to give judges, organizers, and teams an instant x-ray view of project's technical integrity. Instead of manually digging through commit histories and file structures, Introspector parses the repository to generate a comprehensive

"health check" of a submission.

In the heat of the hackathon, hackers often miss important insights about their projects like who architected the critical backend system, who wrote unit tests, how the project was deployed. And judges often rely on demo video and UI missing critical insights like did the team actually use the tech stack they said in the video, what problems they had and how they resolved it, or is the UI/UX developed or imported from third parties.

Hacker Introspector solves the problem at stake

STAKEHOLDERS



Development team
(ME - solo)



Hackathon participants

(Hackers of MLH)



Judges
(MLH JUDGES)

FUNCTIONAL REQUIREMENTS

SCOPE FOR NVP :

- ⇒ Repo URL input : input public github repository URL for analysis.
- ⇒ Tech stack detection : System shall scan repository to identify and list all programming languages, frameworks, and libraries used.
- ⇒ API rules scanner : Verify that a Hackathon required API is used or not.
- ⇒ Timestamp verification : System flags commits made outside hackathon start and end times.

⇒ Commit author & auditors: check if all members contributed and what they contributed or only one person did all the work.

⇒ Commit velocity chart: display graph showing commit activity over duration of hackathon.

⇒ File breakdown chart: Pie chart of repository file type.

OUT OF SCOPE (AFTER MVP)

- ⇒ Batch imports : import multiple projects to audit at once.
- ⇒ Private repo access : support authentication (OAuth or PAT) to analyse private repositories for judges.
- ⇒ Live webhook : automatically reanalyze a project whenever a new commit is pushed during the hackathon.

⇒ complexity metrics : analyze code complexity scores to identify "spaghetti code" vs clean architecture.

⇒ Boilerplate detection, differentiate between autogenerated (create-react-app) and actual hacker written logic.

⇒ project health score: generate health score out of 100 based on commit velocity, documentation, build success etc.

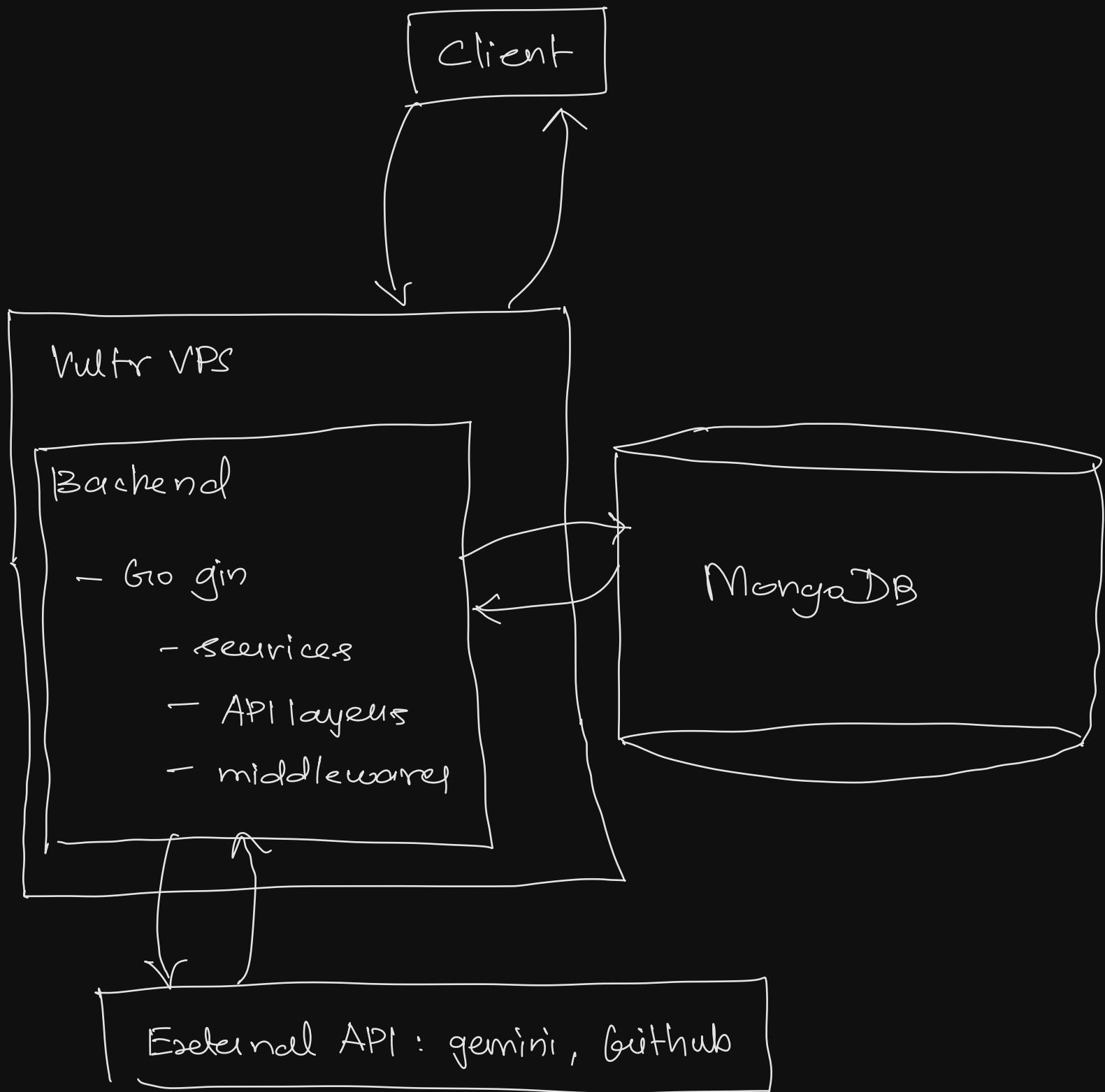
- ⇒ side by side comparison : selected two repositories and compare their metrics side by side.
- ⇒ comment and annotate : allow judges to comment (low commits but high code quality).
- ⇒ export to PDF : Downloadable PDF audit.

NON FUNCTIONAL REQUIREMENTS

For MVP, focus is on completing a working demo, if working demo is completed within hackathon time frame, focus will be shifted to response time, security, failure recovery, and scaling.

DESIGN

high level architecture (not finalized).



Audit workflow (not finalized)

