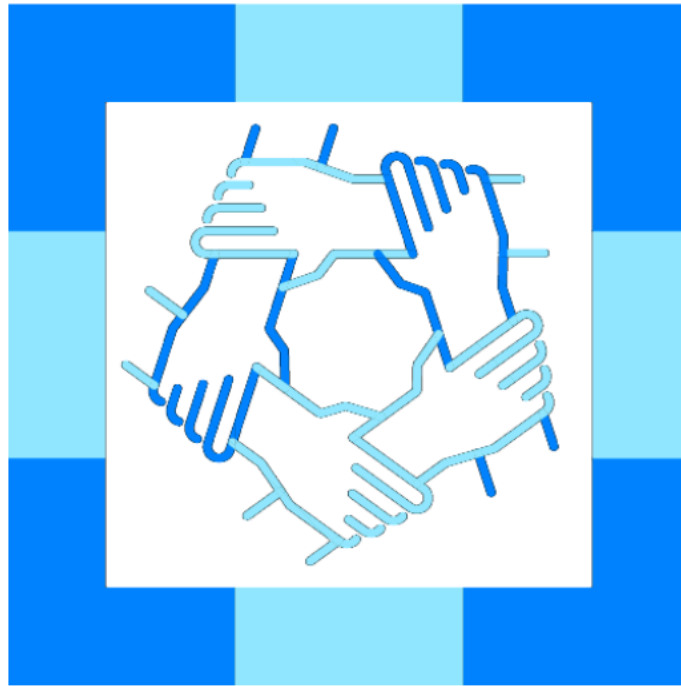


UW-Madison Badger Build Fest



CLOUD NATIVE MADISON

Hackathon Scorecard

Hackathon Scorecard

Team Name: _____

Project Name: _____

1. CI/CD Pipeline (*Automation & pre-deployment scanning*)

| Score | Criteria |
|-------|---------------------------------------------------------------------------------|
| 0 | No automation. Manual build/deploy. |
| 1 | Basic script-based automation (e.g., deploy.sh). |
| 2 | CI pipeline (e.g., GitHub Actions) with static analysis OR dependency scanning. |
| 3 | Robust pipeline that blocks deployment on critical quality/dependency issues. |

Score: ____/3

Notes:-----

2. Configuration & Definition (*Application definition & secret management*)

| Score | Criteria |
|-------|------------------------------------------------------------------------------------------|
| 0 | Critical Failure: Configuration secrets hardcoded in the repository. |
| 1 | Minimal effort: Secrets in a .gitignore'd .env file. |
| 2 | Good: Used standard Kubernetes Configuration/Secrets; app defined in K8s/Helm manifests. |
| 3 | Excellent: Integrated a dedicated secret manager (e.g., Vault) via CSI/injector. |

Score: ____/3

Notes:-----

3. Observability & Audit Trail (Logging, monitoring, and traceability)

| Score | Criteria |
|-------|---------------------------------------------------------------------------|
| 0 | No logging or only basic console.log statements. |
| 1 | Basic, human-readable logs showing application flow. |
| 2 | Implemented structured (JSON) logging (e.g., structlog) for key events. |
| 3 | Logs are shipped to a platform (e.g., Loki) & visualized (e.g., Grafana). |

Score: ____/3

Notes:-----

4. Service Exposure & Policy (Protecting the "front door" - Ingress, API, controls)

| Score | Criteria |
|-------|---------------------------------------------------------------------------------|
| 0 | No API or frontend access controls. |
| 1 | Implemented one basic control (e.g., proper CORS rules or HTTP Headers). |
| 2 | Implemented multiple layers (e.g., Rate Limiting AND a web application filter). |
| 3 | Used a dynamic policy engine (e.g., OPA) to enforce rules at the edge. |

Score: ____/3

Notes:-----

5. Runtime & Identity (Protecting the app while running; user auth)

| Score | Criteria |
|-------|---------------------------------------------------------------------------------------------------|
| 0 | No runtime protection or authentication. |
| 1 | Basic runtime policy (e.g., AppArmor) OR basic user auth (wallet connection). |
| 2 | Implemented both a robust auth mechanism (e.g., JWT) AND a runtime monitoring tool (e.g., Falco). |
| 3 | Demonstrated runtime monitoring actively detecting or preventing a specific unauthorized event. |

Score: ____/3

Notes:-----

General Hackathon Criteria

| Category | Score | Notes |
|-------------------------------|-------|---------------------------------------------------------------------|
| Decentralization & Innovation | / 5 ▾ | (Does this need to be a dApp? Is the idea novel?) |
| Technical Implementation | / 5 ▾ | (Is the code clean? Is the smart contract robust? Does it work?) |
| Presentation & Demo | / 5 ▾ | (Was the demo clear? Did they explain the problem & solution well?) |

Final Score

| Section | Score |
|----------------------------|---------------|
| dApp Stack Technical Total | / 15 ▾ |
| General Criteria Total | / 15 ▾ |
| FINAL TOTAL | / 30 ▾ |