Fullstack Web App - Assignment1

Student name:

Location: Lusaka

Cohort: AI Coding Bootcamp Cohort 1

Total Grade:

Double Weighted Categories:

1. Frontend
2. Backend
3. Quality and Testing
4. Security
5. Architecture and Code Organization

Single Weighted Categories:

1. Design
2. Dev Ex, CI/CD
3. IT Ops
4. Product Management

# Comments

# Rubric (1 = Unacceptable → 5 = Exceptional)

| **Category** | **1 — Unacceptable** | **2 — Needs Work** | **3 — Meets Expectations** | **4 — Exceeds Expectations** | **5 — Exceptional** |
| --- | --- | --- | --- | --- | --- |
| **Design (UI/UX)** | Inconsistent layout; illegible or inaccessible; no mobile support. | Basic responsiveness; noticeable visual bugs; limited attention to a11y. | Clean, consistent UI; mobile‑friendly; passes basic [a11y checks](https://www.a11yproject.com/checklist/) (contrast, keyboard nav). | Thoughtful visual hierarchy; custom theming; comprehensive [a11y](https://www.a11yproject.com/checklist/) (screen‑reader flows). | Pixel‑perfect, branded design; motion/interaction polish; formal a11y audit with fixes. |
| **Frontend Implementation** | Frequent runtime errors; spaghetti code; no state mgmt conventions. | Works but brittle; large components; ad‑hoc state handling. | Modular components; state handled via TanStack Query/RHF; minimal warnings. | Well‑typed hooks; code‑splitting; performance optimizations (lazy‑load, memo). | Production‑level quality: SSR/SEO, exhaustive error states, lighthouse ~90+. |
| **Backend / API** | Endpoints fail or are missing; insecure rules; no validation. | CRUD works but poor error handling; some validation gaps. | tRPC routes typed; Zod validation; Firestore rules enforce auth. | Thoughtful data modelling; composite indexes; graceful failures. | Multi‑env config; seeding scripts; zero‑downtime migrations or blue‑green deploys. |
| **Dev Experience & CI/CD** | Manual builds; no linter/tests in pipeline; flaky deploys. | Basic GitHub Action to deploy; tests run locally only. | Turbo‑cached pipeline: lint, type‑check, tests, Storybook build, preview deploy. | Parallel jobs, test reports, codecov; deploy promotes on tag/Changeset. | Cache‑aware, <5 min runtime; canary deploys; Slack/Discord notifications & rollbacks. |
| **Cloud / IT Ops** | Hard‑coded secrets; no monitoring; unclear infra costs. | Env vars in repo secrets; basic Firebase console logs. | Secrets via T3 Env + functions:config; Cloud Logging dashboards. | Alerting rules, Crashlytics/Sentry integration; cost budgets. | IaC or scripts for full recreate; autoscaling tuned; custom metrics & alerts. |
| **Product Management** | No clear goals; scope creep; missing acceptance criteria. | Trello/Issues exist but vague; ad‑hoc prioritization. | Defined MVP; backlog groomed; demo accepts against criteria. | Road‑map with milestones; burn‑down chart; stakeholder demos. | Data‑driven decisions (analytics); retro action items implemented; public changelog. |
| **Quality & Testing** | No automated tests; manual QA only. | <30 % unit coverage; flaky E2E; lint disabled. | ≥60 % unit+component coverage; Playwright happy path; lint & Prettier pass CI. | Visual regression via Storybook; a11y checks; seed data resets; ≥80 % coverage. | Mutation or property‑based tests; contract/fuzz tests; zero‑regression policy. |
| **Security** | Public DB; default rules; secrets in code. | Auth enforced but rules overly broad; no dependency scanning. | Principle‑of‑least‑privilege rules; OWASP top‑10 reviewed; secrets managed. | Automated security tests (ZAP/GH Dependabot); 2FA enforced on repo. | Threat model documented; security ADRs; periodic penetration test results. |
| **Architecture & Code Organization** | Single huge file; unclear boundaries; no ADRs. | Ad‑hoc folders; circular deps; inconsistent naming. | Follows monorepo layout; shared package for types; ADR in /docs. | Decoupled modules; clear domain boundaries; tree‑shakeable libraries. | Hexagonal/CQRS or similar advanced patterns; plug‑in architecture; exemplary ADR trail. |

# Grading Prompt:

Please evaluate the current project according to the rubric below. If there is anything you can’t evaluate because you don’t have the context to run it then let me know. Otherwise give a score for each category of the rubric, and explain your reasoning for your score, including references to specific files and sections of those files to justify your reasoning. If the score is greater than 1 please give some positive feedback, and if it's less than 5 please give some negative feedback. That way the student knows why they scored what they did. Then calculate the final score for the student.

{

"metadata": {

"assignment": "Fullstack Web App - Assignment1",

"student\_name": "",

"location": "Lusaka",

"cohort": "AI Coding Bootcamp Cohort 1",

"total\_grade": null

},

"double\_weighted\_categories": [

"Frontend Implementation",

"Backend / API",

"Quality & Testing",

"Security",

"Architecture & Code Organization",

"Comments"

],

"rubric": {

"Design (UI/UX)": {

"1": "Inconsistent layout; illegible or inaccessible; no mobile support.",

"2": "Basic responsiveness; noticeable visual bugs; limited attention to accessibility (a11y).",

"3": "Clean, consistent UI; mobile-friendly; passes basic a11y checks (contrast, keyboard navigation).",

"4": "Thoughtful visual hierarchy; custom theming; comprehensive a11y (screen-reader flows).",

"5": "Pixel-perfect, branded design; motion/interaction polish; formal a11y audit with fixes."

},

"Frontend Implementation": {

"1": "Frequent runtime errors; spaghetti code; no state-management conventions.",

"2": "Works but brittle; large components; ad-hoc state handling.",

"3": "Modular components; state handled via TanStack Query/React-Hook-Form; minimal warnings.",

"4": "Well-typed hooks; code-splitting; performance optimizations (lazy-load, memo).",

"5": "Production-level quality: SSR/SEO, exhaustive error states, Lighthouse ~90+."

},

"Backend / API": {

"1": "Endpoints fail or are missing; insecure rules; no validation.",

"2": "CRUD works but poor error handling; some validation gaps.",

"3": "tRPC routes typed; Zod validation; Firestore rules enforce auth.",

"4": "Thoughtful data modelling; composite indexes; graceful failures.",

"5": "Multi-env config; seeding scripts; zero-downtime migrations or blue-green deploys."

},

"Dev Experience & CI/CD": {

"1": "Manual builds; no linter/tests in pipeline; flaky deploys.",

"2": "Basic GitHub Action to deploy; tests run locally only.",

"3": "Turbo-cached pipeline: lint, type-check, tests, Storybook build, preview deploy.",

"4": "Parallel jobs, test reports, code-coverage; deploy promotes on tag/Changeset.",

"5": "Cache-aware, <5 min runtime; canary deploys; Slack/Discord notifications & rollbacks."

},

"Cloud / IT Ops": {

"1": "Hard-coded secrets; no monitoring; unclear infra costs.",

"2": "Env vars in repo secrets; basic Firebase console logs.",

"3": "Secrets via T3 Env + functions:config; Cloud Logging dashboards.",

"4": "Alerting rules, Crashlytics/Sentry integration; cost budgets.",

"5": "Infrastructure-as-Code or scripts for full recreate; autoscaling tuned; custom metrics & alerts."

},

"Product Management": {

"1": "No clear goals; scope creep; missing acceptance criteria.",

"2": "Trello/Issues exist but vague; ad-hoc prioritization.",

"3": "Defined MVP; backlog groomed; demo accepts against criteria.",

"4": "Road-map with milestones; burn-down chart; stakeholder demos.",

"5": "Data-driven decisions (analytics); retrospectives actioned; public changelog."

},

"Quality & Testing": {

"1": "No automated tests; manual QA only.",

"2": "<30 % unit coverage; flaky E2E; lint disabled.",

"3": "≥60 % unit+component coverage; Playwright happy path; lint & Prettier pass CI.",

"4": "Visual regression via Storybook; a11y checks; seed data resets; ≥80 % coverage.",

"5": "Mutation or property-based tests; contract/fuzz tests; zero-regression policy."

},

"Security": {

"1": "Public DB; default rules; secrets in code.",

"2": "Auth enforced but rules overly broad; no dependency scanning.",

"3": "Principle-of-least-privilege rules; OWASP top-10 reviewed; secrets managed.",

"4": "Automated security tests (OWASP ZAP/GitHub Dependabot); 2FA enforced on repo.",

"5": "Threat model documented; security ADRs; periodic penetration-test results."

},

"Architecture & Code Organization": {

"1": "Single huge file; unclear boundaries; no Architecture Decision Records (ADRs).",

"2": "Ad-hoc folders; circular dependencies; inconsistent naming.",

"3": "Follows monorepo layout; shared package for types; ADR in /docs.",

"4": "Decoupled modules; clear domain boundaries; tree-shakeable libraries.",

"5": "Hexagonal/CQRS or similar advanced patterns; plug-in architecture; exemplary ADR trail."

}

}

}