Hassan Naeem Tester Project

Duration 1 Week

1-Week Pre-Trial Project Specification

Objective: Build a HIPAA-compliant proof-of-concept in GCP, covering infrastructure, security, CI/CD, stubbed application, backups, monitoring,and teardown. This spec ensures every requirement is clearly documented for seamless implementation.

---

1. Project Setup & Security Foundations (Day 1)

1.1. Create GCP Project

- **Name:** `dev-therapist`
- **Organization:** Under your GCP Org with an existing BAA
- **Billing:** Link to an account, then create a **Budget** with threshold `$0` and email alert

1.2. Enable Required APIs via Terraform

- Cloud Run
- Secret Manager
- Cloud KMS
- Cloud Build
- Cloud Logging & Monitoring
- Firestore API

1.3. Service Account & IAM Roles

- **Service Account:** `svc-devops@<org>.iam.gserviceaccount.com`
- **Terraform:** Grant exactly:
- `roles/run.admin`
- `roles/secretmanager.admin`
- `roles/cloudkms.cryptoKeyEncrypterDecrypter`
- `roles/logging.logWriter`
- `roles/monitoring.metricWriter`

Deliverable: Terraform files in `terraform/` that spin up project, enable APIs, create service account, bind roles.

---

2. Secrets & Customer-Managed Encryption Key (Day 2)

2.1. KMS Key Ring & Symmetric Key

- Create a **Key Ring** `therapist-keys`

- Create a **Symmetric Key** `db-key`

2.2. Secret Manager

- Define a secret `DB_URI`, encrypted with `db-key`

2.3. IAM for KMS Key

- Grant `svc-devops` and Cloud Build SA `roles/cloudkms.cryptoKeyEncrypterDecrypter` on `db-key`

2.4. Encryption/Decryption Script

- **Path:** `scripts/encrypt.js`
- **Function:** Read plaintext from stdin, call KMS encrypt & decrypt, output both
- **Usage:**

```
echo -n "patient-data" | node scripts/encrypt.js
```

Deliverable: Terraform for KMS & Secret Manager + `encrypt.js` with instructions in README.

---

3. Stub Application & CI/CD Pipeline (Day 4)

3.1. Repository Structure

```
/terraform
/api
  └── index.js (Express + Firestore emulator)
/web
  └── pages/index.js (Next.js stub)
/mobile
  └── App.js (React Native or PWA stub)
/cloudbuild.yaml
.github/workflows/ci.yml
/scripts/encrypt.js
/scripts/backup.sh
/scripts/destroy.sh
README.md
```

3.2. API Endpoints (Firestore Emulator)

- `POST /auth/login` → `{ token: "<dummy-jwt>" }`
- `GET /messages` → `[]`
- `POST /messages` → `{ ciphertext: "<encrypted>" }`

- `GET /analytics/themes` → `{ themes: { stress: 5, sleep: 3 } }`

3.3. Web & Mobile Stubs

- **Web:** Fetch and display stub data
- **Mobile:** Mirror web behavior

3.4. CI/CD (cloudbuild.yaml)

1. Install & lint (`npm ci && npm run lint && npm test`)
2. Build Docker images for `/api`, `/web`, `/mobile`
3. Deploy to Cloud Run:
4. `dev-api`
5. `dev-web`
6. `dev-mobile`
7. **Autoscale:** min=0, max=1
8. Inject secrets from Secret Manager

3.5. GitHub Actions

- Trigger Cloud Build on PR to `main`

Deliverable: Working CI/CD pipeline, stub services live on Cloud Run.

---

4. Backup & Teardown (Day 6)

4.1. Firestore Emulator Backup

- **Script:** `scripts/backup.sh`
- **Command:**

  ```
  npx firestore emulators:export ./backup
  ```

4.2. Clean Teardown Script

- **Script:** `scripts/destroy.sh`
- **Function:** Delete GCP project or run `terraform destroy -auto-approve`

Deliverable: Verified backup export and full teardown to zero resources.

---

5. Monitoring & Alerts (Day 7)

5.1. Terraform Metrics & Alerts

- **Uptime Check:** `dev-web` URL, interval 1 minute

- **Log-based Metric:** Count 5xx responses from `dev-api`
- **Alert Policy:** Email notification when:
- Uptime fails twice in 5 minutes
- 5xx count > 5 in a 10-minute window

5.2. Validation

- Simulate a 5xx error in `dev-api`
- Confirm alert email received

Deliverable: Monitoring and alert rules live, tested.

---

6. Final Deliverables & Demo

1. **GitHub Repo** complete with all code, scripts, Terraform, CI/CD, and README.
2. **README.md** covering:
3. Prerequisites (GCP CLI, Service Account keys)
4. `terraform apply` steps
5. `encrypt.js` demonstration
6. CI/CD deployment process
7. `backup.sh` & `destroy.sh` usage
8. Monitoring & alert test instructions
9. **Demo Video** (2–3 min Loom/Zoom) showing:
10. Project setup & IAM
11. KMS encrypt/decrypt demo
12. CI/CD build & deploy
13. Backup export
14. Teardown
15. Alert trigger

[redacted]

---

7. Expected Outputs & Verification

[redacted]

1. **Terraform Plan & Apply**
2. A successful `terraform plan` listing all resources (VPC, subnets, service accounts, KMS, Cloud SQL, Secret Manager).
3. A clean `terraform apply` log indicating zero errors and creating resources under `dev-therapist`.

4. **Service URLs & Endpoints**

5. **Cloud Run API URL:** e.g. `https://bridgecare-api-<hash>-uc.a.run.app`
   - Verified by `curl https://.../api/auth/login` → `{ token: "<dummy-jwt>" }`.
6. **Web Frontend URL:** e.g. `https://bridgecare-web-<hash>-uc.a.run.app`
   - Should render stub login page.

7. **Mobile App Demo:** A test QR code or link for TestFlight/Play Internal, or PWA URL.

8. **Encryption Demo**

9. Output of `echo -n "sensitive-test" | node scripts/encrypt.js` showing both ciphertext and recovered plaintext.

10. **Backup & Restore Artifacts**

11. A local `./backup/` directory or GCS bucket containing Firestore export files or Cloud SQL export.

12. Successful restore via `scripts/restore.sh`, creating `restore-sql` instance.

13. **Monitoring & Alerts**

14. Uptime check metric visible in Cloud Monitoring dashboard.
15. Log-based metric showing HTTP 5xx counts.

16. An alert email (or Slack message) triggered by simulated downtime or forced 5xx.

17. **CI/CD Pipeline Logs**

18. Cloud Build history showing successful builds and deployments for `/api`, `/web`, and `/mobile`.

19. GitHub Actions checks passing on PR merge.

20. **Demo Video & Documentation**

21. A 3–5 minute video highlighting:
    1. Terraform apply flow.
    2. KMS encrypt/decrypt demo.
    3. CI/CD deploy steps and live service interaction (login → journal stub → analytics stub).
    4. Backup export and restore demo.
    5. Teardown execution.
    6. Alert trigger and email receipt.
22. A comprehensive `README.md` in the repo covering all commands and instructions.
