Project Scoping Submission – FitSense AI

Team Members

- ANJALI PATCHAPPAN
- HARINI HARI
- ACKSHAY NAGAMALLU RAJASEKAR
- BHUMI ASHWINI PANCHAL
- MANOJ HARRIDOSS
- HRISHIKESH PRABHU

1. Introduction

FitSense AI is an AI-powered fitness coaching application designed to deliver personalized, adaptive, and explainable workout guidance. The system uses a large teacher LLM to fine-tune a smaller student LLM that can be efficiently deployed on Google Cloud Platform (GCP). The application supports
goal-based workout planning, dynamic weekly adjustments based on user performance, natural language interaction, exercise execution guidance, and AI-generated workout demonstration videos.

2. Dataset Information

2.1 Dataset Introduction

FitSense AI needs multiple dataset types:

1. Exercise knowledge base (structured)
   Exercise names, muscles, equipment, difficulty, instructions, form cues.
2. Workout plan examples + progression rules
   Templates and rules for periodization, overload, deloading, recovery, constraints.
3. User workout logs (product data)
   Sets/reps/weight/RPE, completion status, fatigue notes, injuries, time availability.
   (This is collected in-app and becomes the main grounding source.)
4. Training science references (text)
   Evidence-backed explanations for "why" behind workouts (volume, intensity, hypertrophy, strength, recovery).

2.2 Data Card

Dataset Name	FitSense AI Multi-Source Training + Product Data
Primary Data Types	Structured exercise metadata, user workout logs, text references, synthetic instruction data
Formats	JSON/CSV (structured), Parquet (analytics), JSONL (LLM training), PDFs/XML (scientific text sources)
Size (initial estimate)	Exercises: 500–2,000; User logs: grows with usage; Synthetic pairs: 50k–300k JSONL
Labels	Plan quality labels (human/heuristic), compliance labels, safety flags, preference pairs (teacher/human)
Update Frequency	Exercises: monthly; User logs: real-time; Synthetic training data: per retraining cycle
Intended Use	Personalized plan generation, grounded Q&A, adaptation engine, safety/technique coaching
Out of Scope	Medical diagnosis, injury treatment advice, nutrition prescription

2.3 Data Sources

- Exercise database (open license)
    - wger project (exercise content + API; exercise/ingredient data licensed CC BY-SA 3.0; app is AGPL)
- Strength performance data (public domain)
    - OpenPowerlifting (competition data contributed to the public domain / CC0 style waiver)
- Scientific text references (license-filtered)
    - PMC Open Access Subset (reusable full-text articles with license groupings; includes commercial-use-allowed subset)
- Optional video datasets (for evaluation only; NOT guaranteed for commercial reuse)
    - Example Kaggle workout video datasets exist, but licensing must be verified per dataset before any training use.
- Product data (in-app)
    - User-entered workout logs, plan edits, feedback, and chat queries (stored with explicit consent).

2.4 Data Rights and Privacy

Licensing

- Use wger exercise text under CC BY-SA → must provide attribution + share-alike for derivative exercise text.
- Use OpenPowerlifting data as public domain (still validate DB-structure caveats per jurisdiction).
- Use PMC OA articles only within license terms (filter to commercial-use-allowed if needed).

User privacy

- Treat workout history as sensitive personal data (health-adjacent).
- Controls:
    - Consent + clear data use policy
    - Data minimization (store only what's needed)
    - Encryption at rest + in transit
    - Per-user access control (no cross-user data leakage)
    - Deletion/export workflow (GDPR-style)
    - No raw user chat logs used for training without opt-in + anonymization

3. Data Planning and Splits

Loading

- Ingest exercise KB from wger API dump (normalized into canonical schema)
- Ingest scientific references (PMC OA) into an embeddings store for retrieval
- Log product events (workouts, edits, feedback) into analytics warehouse

Preprocessing

- Normalize exercises: names, aliases, equipment, muscle groups
- Validate and standardize workout logs: units (kg/lb), timestamps, RPE scale
- Safety tagging: flag risky advice patterns (e.g., pain/injury queries)
- Create training examples:
  - (context = user profile + history + constraints) → (output = plan)
  - (context = exercise) → (output = cues + common mistakes)
  - (context = question + retrieved evidence) → (output = grounded answer + citations)

Splits

- LLM training splits: by user (no user appears in more than one split) to prevent leakage.
- Time-based evaluation: train on weeks 1..N, evaluate on week N+1 adaptation.
- Golden test set: curated prompts (safety + correctness + personalization).

4. GitHub Repository

The GitHub repository contains modular folders for data pipelines, model training, backend services, mobile applications, monitoring, and documentation. A comprehensive README explains setup and usage.

Folder structure (proposed) :-

```
fitsense-ai/
  README.md
  docs/
    scoping.md
    architecture.md
    data_card.md
  data/
    raw/              (DO NOT COMMIT sensitive data)
    processed/
    schemas/
  training/
    teacher_generation/
    finetune_student/
    eval/
  backend/
    api/              (Flask)
    inference/
  mobile/
    app/              (React Native / Flutter)
  infra/
    terraform/
    k8s/
    cloudbuild/
```

monitoring/
  drift/
  metrics/
notebooks/
tests/

README must include

- What FitSense AI does (features)
- Setup (local dev + GCP deploy)
- How to run training + evaluation
- API endpoints + example requests
- Data policy (what is stored, what is not)

5. Project Scope

5.1 Problems

Users don't know how to convert goals → a weekly program (volume/intensity/progression).

Plans don't adapt to real-world execution (missed sessions, fatigue, plateaus).

People perform exercises incorrectly → higher injury risk + poor results.

"Fitness info online" is noisy; users want science-based answers personalized to them.

Generating video demos is expensive; needs caching + reuse.

5.2 Current Solutions

- Static workout apps (fixed programs; limited personalization)
- Human coaching (high quality but expensive + not scalable)
- Generic chatbots (not grounded in user data; hallucinations risk)
- Video libraries (manual creation; cannot generate custom demos)

5.3 Proposed Solution

- Teacher→Student finetuning: large LLM produces synthetic coaching data + preference feedback; smaller LLM serves users on GCP.
- Plan engine + LLM hybrid:
  - deterministic rules for progression safety
  - LLM for personalization, explanation, and natural-language edits
- Grounded RAG chatbot over:
  - user workout logs (personal context)
  - exercise KB
  - filtered scientific references (PMC OA)
- Video generation prompts + caching:
  - prompt builder (LLM)
  - store generated videos in GCS
  - reuse if same exercise + style already exists

6. Current Approach Flow and Bottleneck Detection

6.1 Current Approach (typical user today)

User goal/constraints

- Google/YouTube/Reddit search
  - pick random plan
    - inconsistent execution + no tracking
      - no adaptation / confusion
        - plateau or injury risk

6.2 Bottlenecks

- Information overload + contradictions
- No personalization loop from performance → next plan
- No structured tracking → no grounded coaching
- High cost of generating/hosting new videos without caching

6.3 FitSense AI improvements

- Single interface: plan → log → adapt → explain
- Retrieval-grounded answers (less hallucination)
- Cached generated videos to reduce repeated generation cost

7. Metrics, Objectives, and Business Goals

Key metrics (model + product)

Personalization/Plan Quality

- Plan acceptance rate (% users who keep plan with minimal edits)
- Weekly adherence (sessions completed / planned)
- Progress proxy metrics (volume trend, estimated 1RM trend, reps at fixed load)

Safety

- Unsafe response rate (manual + automated red-team set)
- Injury/pain escalation handling correctness (refusal + safe guidance)

Chat quality

- Grounded answer score (answer supported by retrieved evidence)
- Hallucination rate on “science” queries (human eval)

System

- p95 latency (chat, plan generation)
- Cost per active user (inference + storage + video gen)

Objectives

- Generate a usable plan in <60 seconds
- Improve adherence by measurable margin vs baseline template plan
- Keep unsafe response rate below agreed threshold (acceptance criteria below)

Business goals

- Retention via weekly adaptation loop
- Differentiation via grounded coaching + explainability + generated demos
- Efficient deployment via small model on GCP

8. Failure Analysis

What can go wrong (during project)

- Data licensing mistakes (CC BY-SA handling, dataset misuse)
- Poor eval design → misleading "good" metrics
- Model hallucinations on science/safety
- Cold-start users have little history → weak personalization

What can go wrong (after deployment)

- Drift: user behavior changes (new goals, injuries, seasonal routines)
- Prompt injection: user tries to override safety or access other users' data
- Cost spikes: video generation abuse
- Privacy incidents: misconfigured storage or logs

Mitigations

- License audit checklist + attribution pipeline
- Golden evaluation set + human review gates
- Safety layer:
  - policy prompts + refusal templates
  - allowlist of "safe exercise coaching" behaviors
- Rate limiting + caching for video generation
- Security:
  - per-user isolation
  - secret management
  - logging redaction

9. Deployment Infrastructure

Components

1. Training pipeline (batch)

- Data prep jobs (Cloud Run Jobs / Vertex AI Pipelines)
- Teacher generation job (GPU when needed)
- Student fine-tuning (Vertex AI Training or GKE + GPU)
- Model registry + versioning (Vertex AI Model Registry)

2. Inference (online)

- Flask API on Cloud Run (or GKE for heavier workloads)
- Student LLM hosted on:
  - Vertex AI Endpoint (managed) OR
  - GKE + model server (vLLM/TGI-style)
- Vector DB for RAG:
  - Vertex AI Vector Search or a managed alternative
- GCS bucket for video assets + cache keys

3. Frontend

- Cross-platform mobile app (Android/iOS)
- Auth via Firebase Auth / Identity Platform

Deployment flowchart

Mobile App

- API Gateway / HTTPS
  - Flask Backend (Cloud Run)
    - (a) Student LLM Inference Endpoint
    - (b) RAG Retriever (Vector Search)
    - (c) Workout DB (managed DB)
    - (d) Video Cache (GCS)
      - if cache miss: Video Gen Service -> store in GCS

10. Monitoring Plan

What we will monitor and why:

Model monitoring

- Response quality drift (weekly eval on fixed prompt set)
- Safety regressions (unsafe response detectors + sampled audits)
- RAG grounding health (retrieval hit-rate, citation coverage)

Data monitoring

- Schema changes, missing fields, outliers in logs
- Distribution shifts (workout frequency, goals, equipment availability)

System monitoring

- Latency, error rate, GPU/CPU usage
- Cost monitoring: per-user inference cost, video gen requests

Drift actions

- Trigger retraining if:
    - quality drops below threshold for 2 consecutive eval windows
    - safety flags increase above threshold
    - major distribution shift detected

11.Success and Acceptance Criteria

A release candidate is accepted when:

Functional

- Users can: generate plan → edit plan → log workouts → get next-week adaptation
- Chatbot answers are grounded in user logs + exercise KB + evidence
- Video prompt generation + caching works (no regen on cache hit)

Quality

- Plan acceptance rate ≥ [X%]
- Weekly adherence ≥ [Y%] on pilot users
- Hallucination rate on golden set ≤ [Z%]
- Unsafe response rate ≤ [S%] on safety test suite

Operational

- p95 latency ≤ [target]
- Cost per active user within [target]
- No P0 security/privacy findings in review

12. Timeline Planning

Week 1–2

- Finalize schema + data card + licensing plan
- Build exercise KB ingestion (wger) + initial workout DB
- Define eval sets (plan prompts, safety prompts)

Week 3–4

- Teacher LLM synthetic data generation (plans, cues, edits)
- Student model finetune v1 + baseline evaluation

Week 5–6

- Flask backend + RAG pipeline + user history grounding
- Mobile MVP: onboarding, goals, plan view/edit, logging

Week 7–8

- Adaptation engine (rule + LLM hybrid)
- Video prompt generation + GCS caching
- Metrics instrumentation

Week 9–10

- Monitoring + drift detection prototype
- Safety hardening + red-team testing

Week 11–12

- Pilot test + bug fixes
- Final report + demo + acceptance criteria validation

13. Additional Information

Out-of-scope safety stance

- FitSense AI is not a medical device and will not diagnose/treat injuries.
- For pain/injury/medical conditions: provide safe general guidance + recommend professional consultation.

Coach→Student strategy (why it matters)

- Coach LLM is used for:
  - generating diverse training scenarios (goals, constraints, equipment)
  - creating preference pairs (“good coaching” vs “bad coaching”)
  - grounding templates for RAG and safety responses
- Student LLM is optimized for:
  - low latency
  - predictable cost
  - deployability on GCP