User → variables concerning user lvl. & its id

- id → C
- name → C
- gender → V (0, 1)
- fitness_lvl → V (0, 1, 2)
- focused_muscle_grp → V (0, ... 5)
- age → V (~~27~~ #no)
- ~~goal~~ → V ...

CAE

Muscles

bicep	0
back	1
chest	2
shoulder	3
tricep	4
legs	5

Exercise → catalogue of all possible exercises & their "importance". Each has a certain "cost" & "import" associated.

- id → C
- name → C
- muscles_targetted → V (0, 5, 1) → mostly we want to ↑ this
- calories_10_min → V (#no) → min-max acc to situation → ↑ will lead to exhaustion
- fun_factor → V (0, 1, ... 4) → user assigned fun factor to an ex
- is_compound → V (0, 1) → compounds are better
- how_critical → V (0, 1 ... 4) → how imp is that exercise

Derived From Exercise

- id
- name
- exercise_imp → V (0, ... 10) → how imp ex is → based on ① ~~focus~~ is it in focus area ② is_compound ③ is_critical ④ how much the user do this exercise ~~fun fact~~
- exercise_cost → V (0, ... 10) → ① duration penalty ② how often user does not do this exercise ③ Calorific need
- -/+ ex_recurrance_factor_per_cycle → V (0...10)
- -/+ ex_rpe → V (0 ... 10)
- \+ ex_overall_fun_factor
- --. ex_exhaustion_factor
- - . ex_missed_~~count~~ eg in_cycle_count
- ~~ex~~

~~Completed~~

Daily Log — Workouts

- id
- date
- exercise List
- workout_fun_factor
- duration
- cals_burnt
- ~~muscles~~

Derived Workout

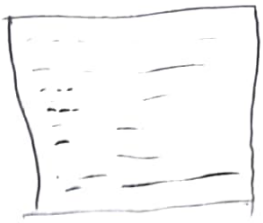
- ~~work~~ suggested_wo_completion_percentage
- wo_imp
- wo_cost
- wo_muscles_targetted
- wo_rpe
- wo_overall_cals

* goal, gender, etc.—

Considerations

→ Last layer is giving out padded result → wts of exercises of exercises all pzbl.

→ Training cycle — or exercise cycle is 30 days.



① Goal?

rec-factor rated-wlo-fn

- Dimension of hidden st?
- No. of neural n/w?

duration, muscles_impa
fun_scale,
is_in_focus_area,
calories,
is_compound,

★ Musc-wt

| Muscle_grp | wt |

→ has improv focus as intensity

★ Exercise_catalogue → has exercise cat psble catalogued

| Ex | Calories | musc_grp | fun_factor | is_compound |

★ Daily log (Todays)

| Date | ExerciseList | Fun Factor | duration | cals | muscle focus |

bicep curl 15 1 3

1, 100, no

cal req for 10 min @ session rated according

★ User Exec Send

Update exercise_user_wt

| Exercise | recurrance | calories | focus | is-comp | avg duration | impactful (perceived) | intensity | ~~duration bal~~ fun_factor |

Name of ex

how recurr the user trains this exercise

calorifics burnt??

the focus area (+ve)

is it compound (+ve)

the duration at which the user trains it in a session (--)

how impactful is this ex. as perceived by the user (+ve)

how much strenous is the exercise perceived (--)

how fun was the exercise (+ve)

-ve/+ve

too_long_dur, exercise_groped

Intl engine ← updates the tbl for next exer

User • console ← takes user IP & cat the exercise

Prediction ← intl get best once

upd wt with how much time was better

multipliers

★ Exercise | Multipliers

will have multipliers for each of the cols of user exer sensitivities

ML where??

ensure overall muscle grp one hit

ensure exercise clubbed

- too_long_duration
- musc_grp_targetted_prev
- cardio_total
- compound_total
- cal_total

Daily Log

- id
- date
- overall_fun_factor
- overall_cals
- duration_of_workout
- muscle_~~focused~~hit
- list_of_exercises
- exhaustion_factor

User

- id
- name
- gender
- fitness_lvl
- focused_muscle_grp
- goal → Future
- age

Exercise

- id →
- name_of_exer
- muscle_grp_hit
- calories_10_min
- fun_factor → user set
- is_compound
- criticality ??

Workout

- List < Exercise >

Derived_User

- consistency_score

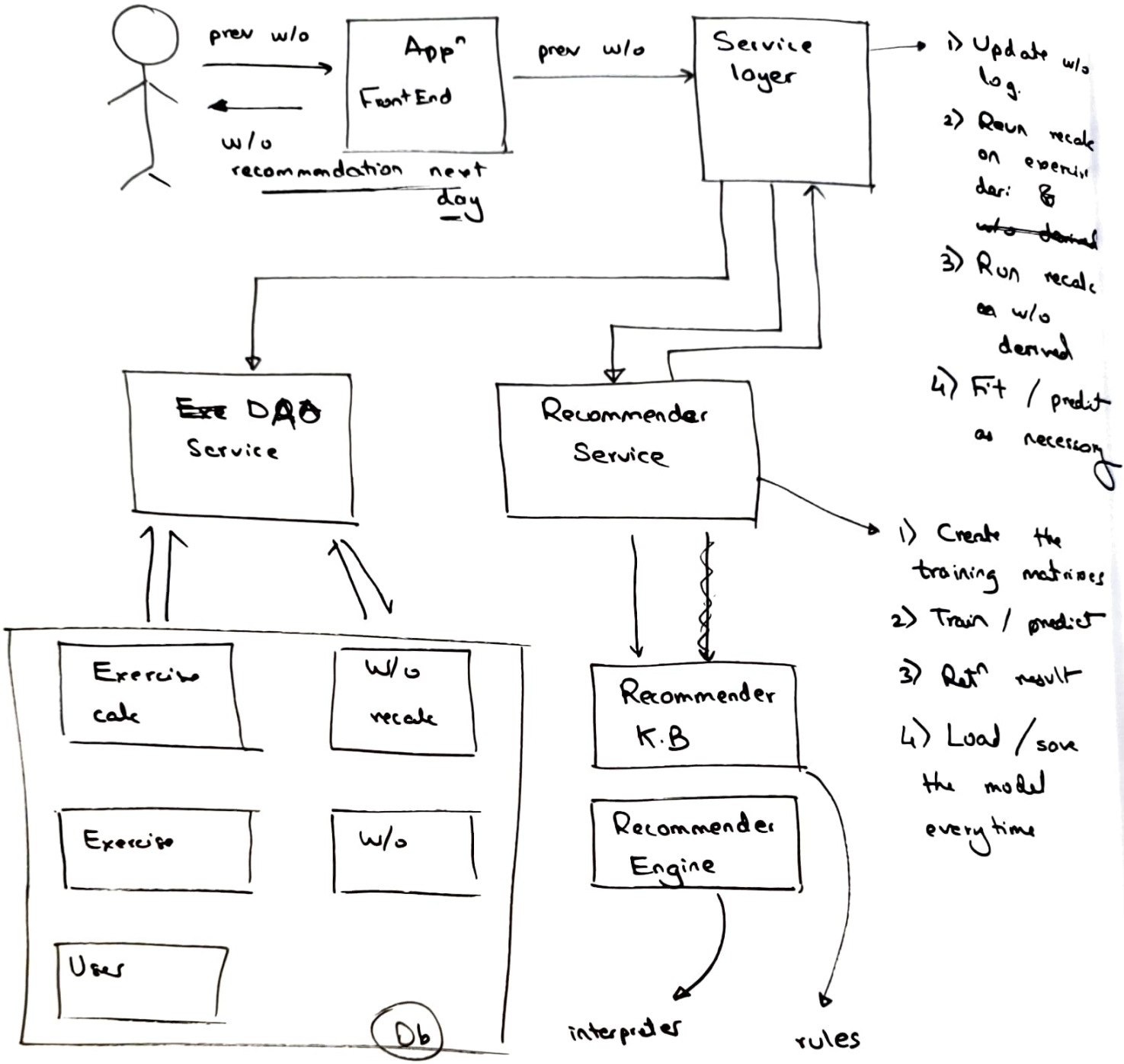
Derived_From_Exercise

- name
- exercise_weightage → calc based on params like criticality, fun_factor, muscle_grp_hit, calories
- imp
- cost
- recuv_fact
- rpe
- fun_factor
- exhaustion_factor

✓ ~~recovered factor~~

✓ ~~perceived_overall~~ 2 fun_factor

- percieved_calories_burnt
- overall_duration
- ~~overall_rpe~~


prev w/o
App^n
Front End
w/o
recommendation next
day
prev w/o
Service
layer
1) Update w/o
log.
2) Rerun recalc
on exercise
dep: &
~~w/o derived~~
3) Run recalc
on w/o
derived
4) Fit / predict
as necessary
~~Exe~~ DAO
Service
Recommender
Service
1) Create the
training matrices
2) Train / predict
3) Ret^n result
4) Load / save
the model
every time
Exercise
calc
W/o
recalc
Recommender
K.B
Exercise
W/o
Recommender
Engine
User
Db
interpreter
rules

App^n FrontEnd

→ get data from user
→ display prompts asking for i/p
→ push to Service layer
→ disp data to user from service layer

CAE

Service Layer

→ get u/o of user
→ interact with Dao Layer to store / retrieve u/o's recommendation
→ trigger recommender service ~~as~~ & ~~when~~ to get u/o
→ get / push to front end service

Dao Layer

→ get u/o data / set
→ get ex / set data
→ etc...

& update the DaO as per Recomm K.B

Recommender Service

→ decide when to train / predict ⟹ Call the engine
→ call the engine / k.b
→ call / read from Database DaO
→ save / load the model.

Recommender Engine

→ Actual update code on tables logic

→ Use K.B to get rules
→ Train / read from the model.

Recommender K.B

→ The rules - aka - the way in which the calc are made & on u/o & ex ⟹ upd Db
→ The weightages with which u need to update the derived tables
→ DaO to update the tables