SUM System Design

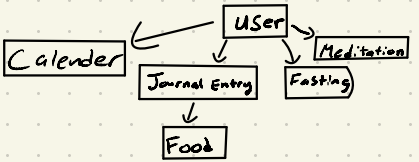
SUM

React-Native Application

expo-sqlite persistant dB across app restarts
metro.config.js root for project assets
[Data from Expo Docs SDK/Sqlite]
Const db= SQLite.openDatabase("db.db") doesn't store db file directy in the project
It's managed by Expo SDK
Expo Sqlite pkg abstracts physical file location and provides connection object.
I likely will be using an ORM here to create my DB tables as that's what I'm used to.
OPTIONS: TypeORM, WatermelonDB, Sequelize, SQLite ORM
Sequelize: Large community and extensive Docs. [Primarily for server side use.
WatermelonDB is ideal for offline data storage and synchronization. Intuitive API, needs to work offline and sync when network is available
For local storage our options then become either ASyncStorage or WatermelonDB but they're very different
Watermelon: useful for complex data structs & large datasets, uses SQLite dB engine.
ASyncStorage: is a simple key-value storage mechanism primarily used for basic local storage doesn't offer advanced querying or relational data modelling. Performance also degrades w/ larger data sets
For our app then we decided to use Watemelon DB.
WatermelonDB goal is real world performance Open Source.
But the main reason were going for WMDB is b/c it's 1. ORM 2. Relational DB.
Redux and WatermelonDB serve different purposes and CAN be used together.
In some cases watermelonDB can replace the need for redux when it comes to managing local data persistance & accessing dB.
WmDB & ReduX can co-exist in RN application. WmDB primarily focuses on local persistant data and management while redux provides a broader framework for managing app state.
In our use case we will be using WmDB w/ Redux.
Our main use case for WmDB will be its own object relational mapping (ORM Layer)
ORMs help define models, perform dB queries & manage relationships b/w entities.
*Cocoapods pkg manager for iOS development & RN App for iOS devices) requires it.
Babel bug "require decorator bfe export option, value must be bool is cured by using babel/plugin-proposal-decorators the dev dependancies Adding Legacy:True adds this dependancy to dev dependencies again without us knowing.

Data Model

What is the relationship b/w Calender to (Fasting, Meditation, Journal)



We should discuss the configuration of journal entries. is it just for food daily log completion or relationship w/ Fasting/Meditation.

PK - Primary Key
FK - Foreign Key

User
id - PK
email - String
name - String
journal_entries - FK (one-to many user-to-journal)

JournalEntry
id - PK
date - Datetime
food_id - FK
user_id - FK

Food
id - PK
name - String
calories - Int
macros - String
meal category - String

With this structure we can establish a many-to one relationship between the Food tabe and JournalEntry table

Fasting
id - PK
user_id - FK
start_date - TIME
End_date - TIME
Duration - (Hours/Mins)
notes - String (optional)

Meditation
id - PK
user_id - FK
date - Timestamp
duration - TIME (mins)
mood - String (optional)

Calender
date - PK
Fasting_id - FK
Meditation_id - FK
Food_journal_id - FK

Date of calender entry was created

By linking respective FKS to Calender Table, We establish relationship between (Fasting, meditation, journal) and corresponding dates in the Calendar

This Struct allows us to query Calendar Table based on dates and retrieve data for each of these activities.

Notes can be user writing how they feel during the fasting period

Videos
id - PK
videourl - String
title - String
description - String
category - String
likes - int
views - int
upload_date - time
thumbnail_url - String

Videos are very resource-intensive utilize CDN

We should also discuss the motivational short videos we want to implement, how we would use a video scrolling feed like (Tiktok) for motivation

SUM App System Design

System Design: Distributed Systems Backend

Clients is a frontend UI Design Problem

Customers → SERVER → Database

Frontend
Mobile (tablet/Phone)
Web (Desktop)

Required Functions
- Fasting Timer
- Food Diary
- Food History
- Meditation Timer
- Calendar
- Offline Mode
- Food Search
- Video feed (for fasting)

Server — Data Management CRUD — CREATE DELETE UPDATE READ

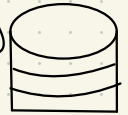
API's

getCalendar(id, timestamp, author)
postCalendar
getUser
getFasting
postHistory
getHistory

Mobile Specific Problem
1. Limited Resources: Internet traffic, battery usage, memory
2. High Level architecture 3. Design Patterns
4. UI Architecture 5. Backend
6. Data storage 7. App Specific

Solution needs to be organized to reduce usage of these resources.

SQL DB Local Storage
WatermelonDB



Author is user Foreign key

Calendar

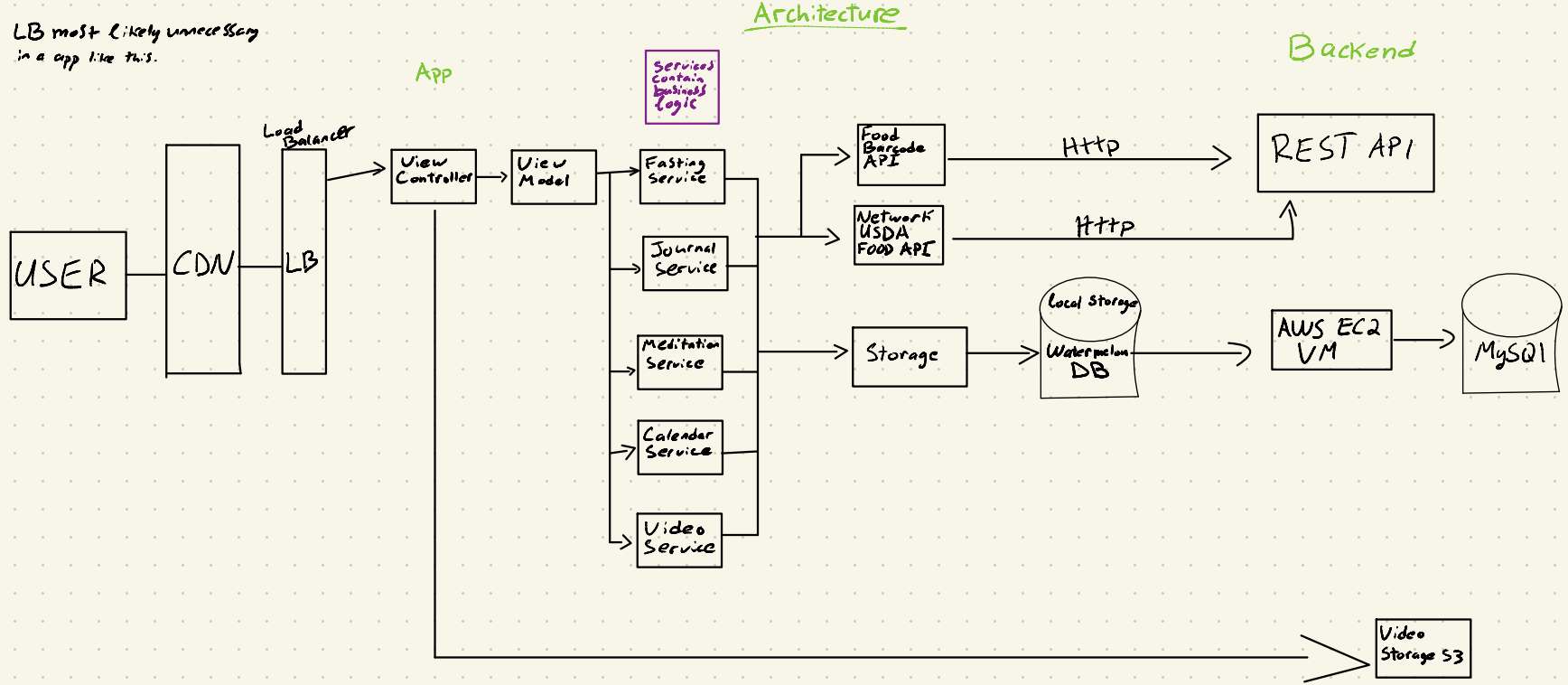
ID	Timestamp	Author
1	Date	User
2		
3		
4		
5		

User

ID	
1	Name

History

ID	Food
1	Food_List


Architecture
LB most likely unnecessary in a app like this.
App
Services contain business logic
Backend
Load Balancer
USER
CDN
LB
View Controller
View Model
Fasting Service
Journal Service
Meditation Service
Calendar Service
Video Service
Food Barcode API
Network USDA FOOD API
Http
Http
REST API
Storage
Local Storage
Watermelon DB
AWS EC2 VM
MySQl
Video Storage S3

UI LAYER