AESTHETIC RECORD

Architectural Review & Proposal

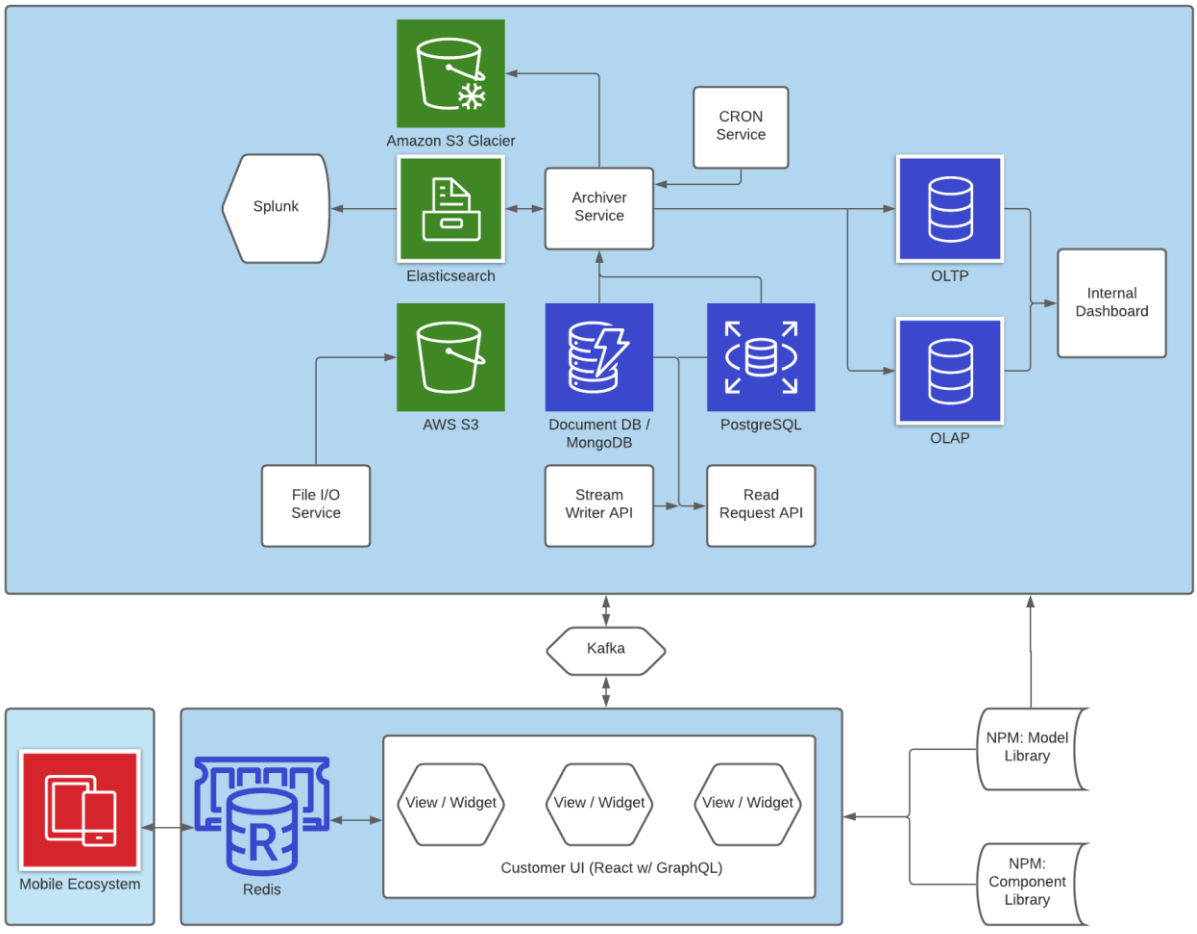
The initial review of the existing application, and its structure, drew me to the conclusion that the problem to fix is not just scalability on its own accord, but rather design for healthy scalability. The review results in two reasonable pathway forwards.

- Wrap the entirety of the existing application in a containerization service, such as Docker, and tie auto-scaling to Kubernetes configurations that are determined by the business.
  - Pro: Quickest pathway forward to issue remedy.
  - Major Con: Hosting budget will eventually become exorbitant with growth and concurrent user increase. Eventually hosting expenditure would exceed the investment required for producing a healthy, intrinsically scalable application.

It is not my recommendation for the aforementioned to be a permanent solution. It is best suited as a short-term band aid for avoiding poor user experience and potential brand damage. Only Batman can throw money at problems until they're fixed.

- The other solution is to develop and implement a healthy, scalable architecture.
  - Pro: Resolves underlying issues of the application; creates solid foundation for new features development.
  - Con: Time investment; development investment.

Proposed Architecture Map v1.0.0



Services & Technology Overview

AWSS3: AWS S3 is the proposed service to be leveraged for non-stale directory-based storage.

AWSS3G: Amazon S3 Glacier is the proposed service to be leveraged for stale directory-based storage. Stale storage is best suited for data which the business does not anticipate any use cases, but as a matter of retention policy, does not wish to purge or apply a TTL.

AS: Archiver Service will be the microservice which manages data assigned a TTL within its current storage context; it will also be the microservice which writes business defined pertinent data for transactions and analytics into the corresponding OLTP and OLAP database; service to also be utilized as a Kafka topic subscriber for *business* case (not technical) errors experienced within the application during run time.

CS: CRON Service will be the microservice which triggers all timed operations pertinent to application functionality (i.e. Billing, Archiving, migrating deltas to OLTP and OLAP data stores).

CUI: Customer UI represents the front-end UI/UX; major technologies proposed are the React library, Apollo Client and GraphQL design.

Containerization: Docker & Kubernetes (blue containers) are the technologies proposed to be used for automation of application scalability.

Document DB / MongoDB: Proposed data store source for fluid and dynamic data that is not definitively structured nor permanently scoped. Complements GraphQL design well.

Elasticsearch: Proposed technology for aggregation of time series data (logs).

FIOS: File I/O Service microservice which will manage any fetch requests of directory storage, uploads to directory storage and user provided files for data to be imported from.

ID: Internal Dashboard would be the single source of truth for data visualization of logs, transactions and analytics for pertinent business partners.

Kafka: Asynchronous message bus which allows data processing and transactions without held connections. Will improve performance and latency.

ME: Mobile Ecosystem is the containerized mobile platform for the application.

NPMCL: NPM Component Library is the storage method through which UI components should be housed; a private, importable repository from NPM (or an alternative service / private repository). This ensures Component models have a single source of truth and are not changed within downstream services.

NPMML: NPM Model Library is the storage method through which application Models should be housed; a private, importable repository from NPM (or an alternative service / private repository). This ensures application Models have a single source of truth and are not changed within downstream services.

OLAP: Online Analytical Processing DB is a data store service that houses the best report generating structure, as determined by the business, and pertinent partners, to perform analytics on data stored.

OLTP: Online Transaction Processing DB is a data store service that houses the best report generating structure, as determined by the business, and pertinent partners, to perform analytics on transactions and deltas within the application.

PostgreSQL: Relational data store service for definitively structured data (i.e. Billing, Permanently Scoped Entity).

RRAPI: Read Request API will be the service which fulfills read-only fetch request; providing separation from Create, Update and Delete operations.

Redis: In memory data store that is built around efficient session and cache management for an application.

Splunk: Out of the box solution for a log visualization service; would be deprecated after log visualization properly integrated into ID.

SWAPI: Stream Writer API will be the service which fulfills Create, Update and Delete requests within the application.

Generator (not pictured, development environment only): A service written with the intent to fulfill any mock data / response requests during development for all intake services, response services and data aggregation services.

Proposed Development Approach

- Existing Application
  - Suspend new feature development on existing application.
  - Prioritize *only* mission critical bug resolutions to avoid brand damage and continued poor user experience.
- Proposed Rewrite
  - *Minimum* estimated time frame for completion; 7 – 8 months from start of first sprint.
  - Establish additional Development and UAT environments for new development.
  - Confirm business requirements for new features requested to be developed during rewrite (three-week window from proposal acceptance date).
  - Freeze new feature request acceptance at the end of the aforementioned third week.
  - Iron out parameters and protocols for acceptance of DCR's (design change requests) after freeze date.
  - Confirm business requirements for TTL (time to live) on data storage and archival needs of the business.
  - Establish definitive UAT process with business for confirmation of development completion.
  - Plan database migration.
  - Plan integration case testing.
- Post Rewrite
  - Execute database migration.
  - Integrate.