TECHNICAL ARCHITECTURE BLUEPRINT

Paid Media Performance Dashboard (V1)

1. System Overview

The Paid Media Performance Dashboard is a full-stack application designed to:

1. **Ingest CSV files daily from an S3 bucket owned by paid media company that Stephen works with**

2. **Transform raw data into structured database tables**

3. **Apply client-specific CPM values to recalculate spend-based metrics**

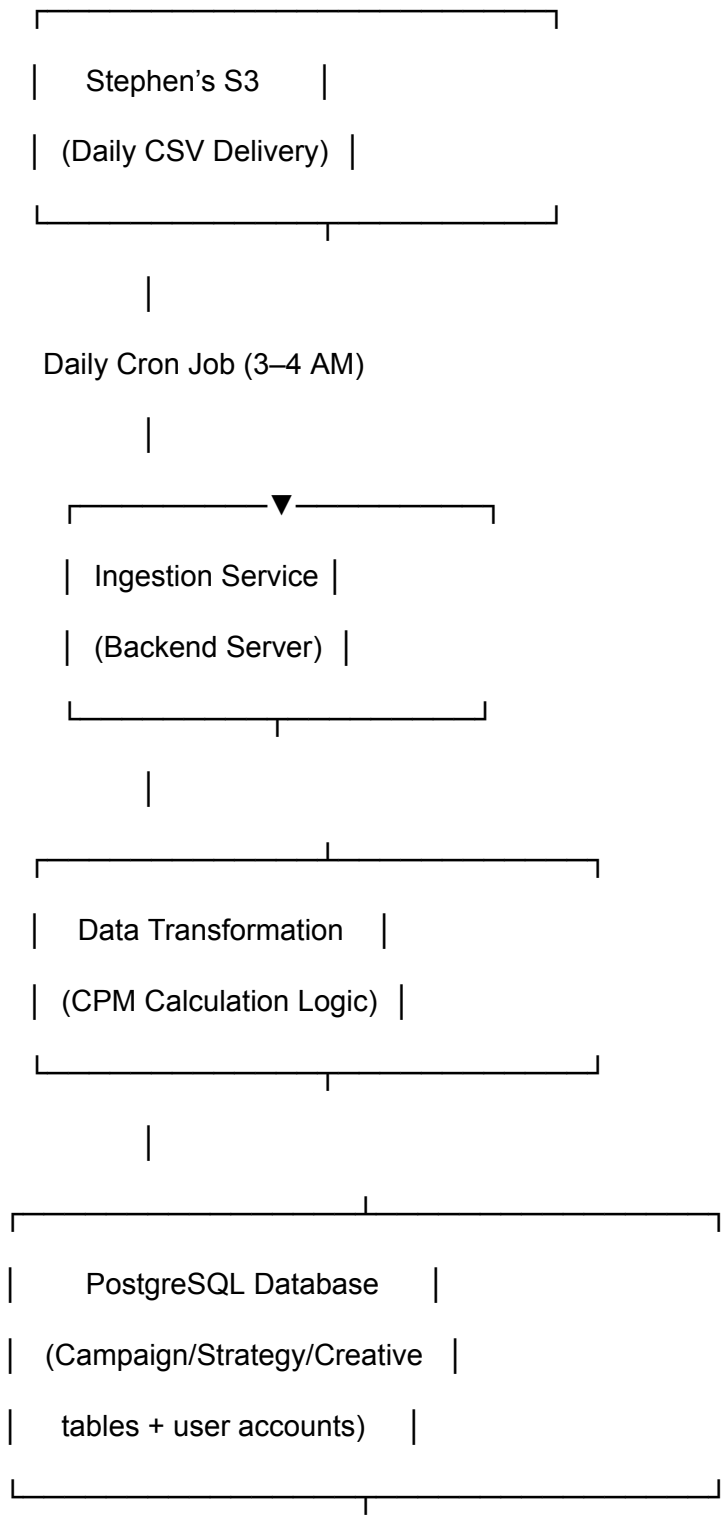
4. **Serve a modern SaaS dashboard interface** that displays:

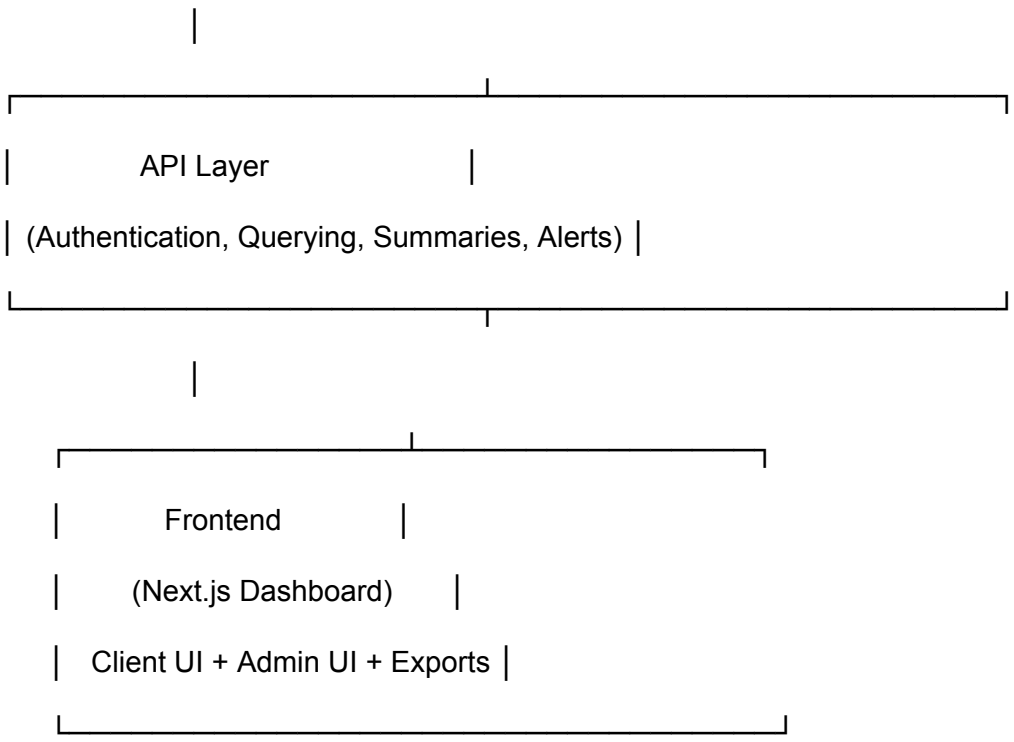
    - Campaign summaries
    - Strategy-level performance
    - Placement performance
    - Creative performance
    - Weekly & monthly summaries

5. **Allow secure client login** with admin-created accounts

6. **Provide Stephen an admin portal** to manage users, CPM values, and ingestion logs

2. High-Level Architecture Diagram





3. Core System Components

Below are the main components and how they work together.

3.1 Frontend Application (Next.js)

Responsibilities:

- Client dashboard interface
- Admin portal interface
- Authentication pages
- Summary pages
- Filters, charts, tables
- Export buttons (PDF/CSV)

Key Libraries:

- **Next.js**
- **TailwindCSS** (or similar modern styling)
- **Recharts / Tremor** for charts

- **NextAuth or custom auth handler**

3.2 Backend Server (Node.js or Python FastAPI)

Responsibilities:

- Fetch data from S3 daily
- Parse CSV files
- Validate records
- Write data to the database
- Apply CPM logic to derive spend-based metrics
- Trigger weekly/monthly summaries
- Generate error alerts when ingestion fails

Key Features:

- Cron scheduler
- Error logging
- Data validation module
- File parser module
- Summaries engine

3.3 Database Layer (PostgreSQL)

Required Tables:

1. **users**
2. **clients**
3. **campaigns**
4. **strategies**
5. **Placements**
6. **creatives**
7. **daily_metrics**
8. **weekly_summaries**
9. **monthly_summaries**
10. **ingestion_logs**
11. **client_settings (contains CPM)**

3.4 Authentication Service

Login Requirements:

- Email + password
- Admin-created credentials
- JWT session tokens
- Basic role-based system:
    - `admin` (Stephen)
    - `client` (client users)

Optional:

- Password reset by admin only

3.5 Ingestion Engine

Runs once daily via cron job:

Process:

1. Connect to S3 bucket
2. Look for new CSV file
3. Download file
4. Validate structure
5. Parse rows
6. Insert raw metrics into database
7. Apply CPM adjustment
8. Update derived metrics (CPC, CPA, ROAS…)
9. Store daily ingestion log
10. If failure → send alert to Stephen

3.6 Alerting System

Triggers alerts for:

- Missing daily file
- Parsing errors
- Validation errors
- S3 authentication issues
- Database write failures

Delivery method:

- Email alert to Stephen

3.7 Summary Engine

Two automated scripts:

Weekly Summary Engine

Runs Mondays at 5 AM
Produces:

- Totals
- Top campaigns
- Underperformers
- Week-over-week deltas
  Stores results in `weekly_summaries`

Monthly Summary Engine

Runs on the 1st of every month

Produces:

- Month totals
- Campaign rankings
- Creative rankings
- Month-over-month trends
  Stores results in `monthly_summaries`

Displayed in dashboard UI

4. Data Flow Diagram

[Daily CSV in S3]

↓

[Ingestion Engine]

↓

[Raw Data Saved to DB]

↓

[Apply CPM → Calculate Spend/CPC/CPA/ROAS]

↓

[Store Finalized Metrics in daily_metrics]

↓

[Frontend API Requests Data by Client]

↓

[Frontend displays results in dashboard]

5. Security Overview

Security controls included in V1:

- HTTPS enforced
- Encrypted S3 credentials
- JWT session expiration
- Role-based access
- Database row-level protection
- Sanitized CSV inputs
- Audit logs for ingestion

6. Deployment Strategy

Frontend

- Vercel (Next.js optimized)

Backend

- AWS ECS or AWS Lambda (depending on preference)
- AWS Cron Scheduler for ingestion

Database

- AWS RDS (PostgreSQL)

File Storage

- Stephen's S3 bucket

7. Scalability Considerations

This system is built so that future phases can easily include:

- SEO data
- GA4 data
- CRM revenue attribution
- Multi-channel dashboards
- Additional data partners
- Auto client onboarding
- White-labeling

The modular architecture keeps all business logic separated by service.

8. Error Handling & Logging

The system keeps a daily ingestion log including:

- File name
- Timestamp
- Records ingested
- Records rejected
- Error messages (if any)
- Resolution status

Admin UI allows Stephen to view logs.

9. Performance Expectations

- Dashboard load time: under **2 seconds**
- Historical dataset queries optimized with indexes
- CSV ingestion: under **20 seconds per file**
- Weekly/monthly summary generation: under **30 seconds**

10. Technical Dependencies (Tentative)

Frontend:

- Next.js
- Tailwind
- Tremor or Recharts
- Axios/React Query for API calls

Backend:

- Node.js / Express or Python FastAPI
- AWS SDK
- Cron

Database:

- PostgreSQL
- Prisma ORM or SQLAlchemy (depending on language)