Quality Tier Classifier Platform

Comprehensive Technical & Functional Documentation

Version: 2026.1
Last Updated: January 22, 2026
Deployed URL: equalitybeta.abacusai.app (https://equalitybeta.abacusai.app)

Table of Contents

---

1. Executive Summary

The **Quality Tier Classifier** is an enterprise-grade traffic source quality management platform designed to automate the classification and optimization of advertising sub-IDs (traffic sources) based on call quality and lead transfer performance metrics. The platform processes CSV data exports from BigQuery, applies sophisticated rule-based classification logic with ML-powered analytics, and provides actionable recommendations for tier promotions, demotions, and pause decisions.

Core Value Proposition

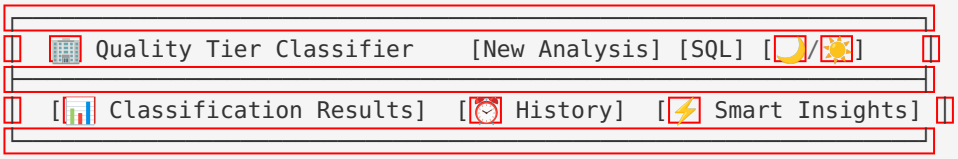
- **Automated Classification**: Processes 900+ records in seconds with consistent rule application
- **Multi-Dimensional Analysis**: Supports 5 aggregation dimensions (Sub ID, Source, Placement, Media Type, Overall)
- **Intelligent Insights**: ML-powered clustering, anomaly detection, and priority scoring
- **Action Audit Trail**: Complete history logging with user attribution and revenue tracking
- **14-Day Warning System**: Prevents abrupt pauses with grace periods for remediation

---

2. UI/UX Elements & Key Features

2.1 Navigation Architecture

The platform employs a **hierarchical tab-based navigation** system:



Primary Tabs:
1. **Classification Results** - Main data table with filtering and actions
2. **History** - Action audit log with comprehensive filtering
3. **Smart Insights** - AI/ML-powered analytics dashboard

Top Bar Actions:
- **New Analysis** (🚀) - Start fresh CSV upload
- **BigQuery SQL** - Generate data extraction query
- **Theme Toggle** - Dark/Light mode switcher

2.2 Workflow Stages

The application follows a **3-step wizard workflow**:

```
[1. Upload CSV] → [2. Map Columns] → [3. View Results]
```

Stage 1: Upload Step

- Drag-and-drop CSV file upload
- File validation and preview
- BigQuery export format guidance
- Example data download link

Stage 2: Mapping Step

- Automatic column detection and suggestion
- Required vs optional field indication
- Field grouping by category (Core, Call, Lead, Click, Redirect, Meta)
- Validation before proceeding

Stage 3: Results Dashboard

- Full classification results table
- Expandable row details
- Bulk selection and actions
- Export functionality

2.3 Main Classification Table

The centerpiece of the platform with **enhanced readability** (2026 update):

Element	Specification
Header Font	13px, weight 600, sticky positioning
Data Font	13-15px depending on column importance
Row Padding	12px vertical, 10-12px horizontal
Sub ID Display	15px bold, purple highlight (#BEA0FE)
Revenue Column	14px bold, green accent (#D7FF32)
Max Height	78vh with smooth scrolling

Columns Displayed:
- ☐ Selection checkbox
- ± Expand/collapse toggle
- Sub ID (primary identifier)
- Vertical (Medicare, Health, Home, Life, Auto)
- Traffic Type (Full O&O, Partial O&O, Non O&O)
- Current Tier → Recommended Tier
- Action Badge (color-coded)
- Quality Metrics (Call %, Lead %)
- Volume Metrics (Call Vol, Lead Vol, Click Vol, Redir Vol)
- RP Metrics (RPQCall, RPLead, RPClick, RPRedir)
- Total Revenue

2.4 Expanded Row Detail View

When a row is expanded, a **4-tab detail panel** appears:

```
┌──────────────────────────────────────────────────────────────┐
│ [📊 Analysis] [💰 Revenue & Volume] [📝 Notes] [✏️ Log Action] │
├──────────────────────────────────────────────────────────────┤
│                    Tab Content Here                          │
└──────────────────────────────────────────────────────────────┘
```

1. **Analysis Tab**: Quality breakdown, threshold comparisons, peer benchmarking
2. **Revenue & Volume Tab**: Revenue composition chart, volume breakdown by channel
3. **Notes Tab**: Contextual AI recommendations, historical performance notes
4. **Log Action Tab**: Action recording form with name input and notes

2.5 Smart Insights Panel (AI/ML)

The ML-powered analytics dashboard provides:

Cluster Overview

- **5 Behavioral Clusters**: Star Performers, Solid Contributors, Growth Potential, Watch List, Critical Attention
- **Interactive Cards**: Click to drill down into cluster members
- **Cluster Metrics**: Count, avg revenue, quality metrics

Drill-Down View

- Back navigation to cluster overview
- Summary statistics bar
- Scrollable list of Sub IDs with:
- Composite Score
- Call Quality Rate
- Lead Quality Rate
- Click-to-navigate to main table

Priority Matrix

- Impact × Urgency × Confidence scoring
- Timeframe categorization (Immediate, Short-term, Medium-term)
- Potential revenue projections

2.6 Theming System

eMAX Brand Color Palette:

Color Name	Hex Code	Usage
Excel Green	`#D7FF32`	Success, Premium tier, positive metrics
Excel Purple	`#BEA0FE`	Standard tier, Sub ID highlights
Excel Orange	`#FF7863`	Warnings, Pause actions, negative
Excel Black	`#141414`	Dark mode background
Excel White	`#F5F5F5`	Light mode background

Theme Implementation:
- `ThemeProvider` context wrapping entire app
- `useTheme()` hook for component access
- LocalStorage persistence for preference
- Dynamic CSS variable application

2.7 State Persistence

The application maintains state across navigation using:

```
// LocalStorage keys used:
localStorage.setItem('classifier_csv_data', JSON.stringify(csvData));
localStorage.setItem('classifier_column_mapping', JSON.stringify(mapping));
localStorage.setItem('classifier_results', JSON.stringify(results));
localStorage.setItem('classifier_step', currentStep);
```

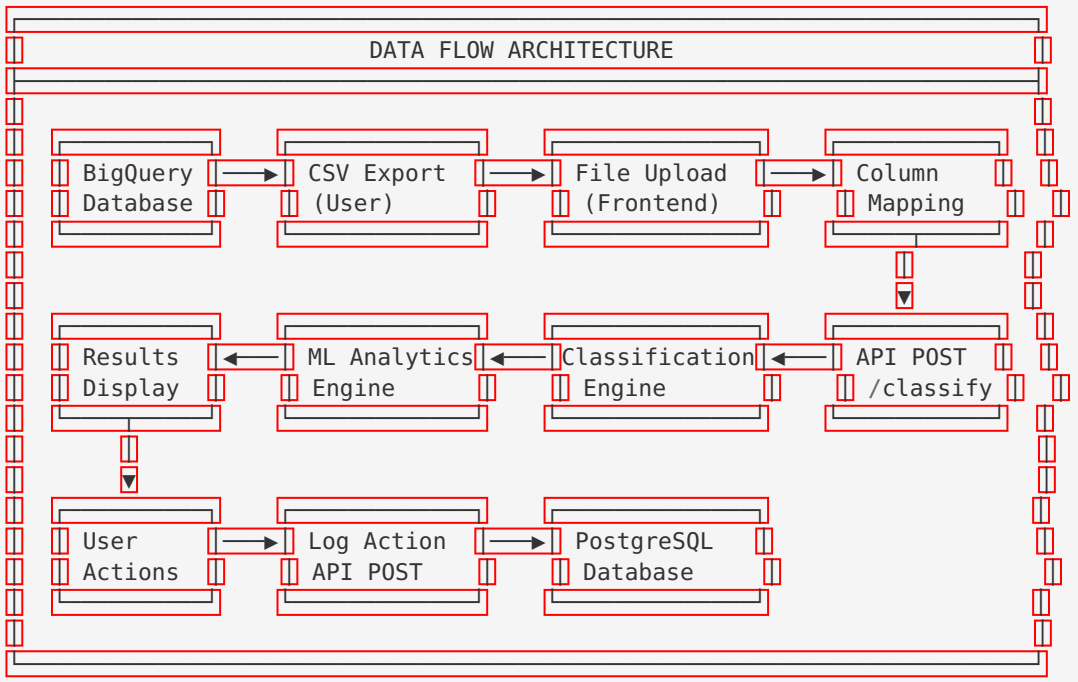
Hydration Handling:
- Server-side rendering with client-side state restoration

- `suppressHydrationWarning` on dynamic content
- Deterministic initial renders

---

3. Data Flow & Processes

3.1 End-to-End Data Pipeline



3.2 CSV Processing Pipeline

Step 1: File Parsing

```
// Browser-side CSV parsing using native APIs
const text = await file.text();
const lines = text.split('\n');
const headers = lines[0].split(',').map(h => h.trim());
const rows = lines.slice(1).map(line => parseCSVLine(line, headers));
```

Step 2: Column Mapping

```
// Required fields (must be mapped)
const REQUIRED_FIELDS = ['subid', 'vertical', 'traffic_type', 'internal_channel'];

// Optional fields by category
const FIELD_GROUPS = {
  core: ['subid', 'vertical', 'traffic_type', 'internal_channel', 'cur-
rent_classification'],
  call: ['total_calls', 'paid_calls', 'calls_over_threshold', 'call_quality_rate', 'ca
ll_revenue'],
  lead: ['total_leads_dialed', 'leads_transferred', 'lead_transfer_rate', 'lead_revenu
e'],
  click: ['click_volume', 'click_revenue'],
  redirect: ['redirect_volume', 'redirect_revenue'],
  meta: ['channel', 'placement', 'description', 'source_name', 'media_type', 'cam-
paign_type']
};
```

Step 3: Aggregation (Multi-Dimensional)

```
// Aggregation key generation based on selected dimension
function getAggregationKey(row: ParsedRow, dimension: AggregationDimension): string {
  switch (dimension) {
    case 'sub_id': return row.subId;
    case 'source_name': return `${row.sourceName}|${row.vertical}|${row.trafficType}|$
{row.internalChannel}`;
    case 'placement': return `${row.placement}|${row.vertical}|${row.trafficType}|${ro
w.internalChannel}`;
    case 'media_type': return `${row.mediaType}|${row.vertical}|${row.trafficType}|${r
ow.internalChannel}`;
    case 'overall': return `${row.vertical}|${row.trafficType}|${row.internalChannel}
`;
  }
}
```

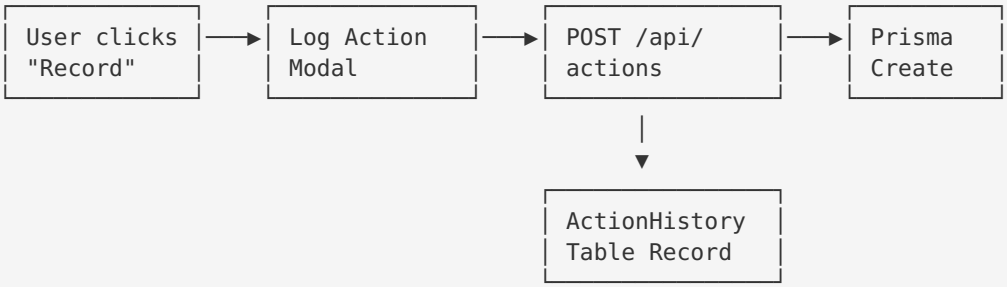
Step 4: Metric Calculation

```
// Quality rate calculations
callQualityRate = callsOverThreshold / totalCalls;  // Calls meeting duration
threshold
leadTransferRate = leadsTransferred / totalLeadsDialed;  // Successful transfers

// Revenue per unit calculations
rpQCall = callRevenue / paidCalls;      // Revenue per qualified call
rpLead = leadRevenue / leadsTransferred; // Revenue per transferred lead
rpClick = clickRevenue / clickVolume;    // Revenue per click
rpRedirect = redirectRevenue / redirectVolume; // Revenue per redirect
```

3.3 Classification Pipeline

```
// Classification flow for each record
for (const row of aggregatedRows) {
  // 1. Derive current classification from existing tier
  const currentClassification = deriveCurrentClassification(row.currentTier);

  // 2. Get applicable thresholds for vertical + traffic type
  const thresholds = getThresholds(row.vertical, row.trafficType);

  // 3. Classify individual metrics
  const callClassification = classifyMetric('Call', row.callQualityRate, row.totalCall
s, thresholds);
  const leadClassification = classifyMetric('Lead', row.leadTransferRate, row.totalLea
dsDialed, thresholds);

  // 4. Apply combined classification rules
  const result = classifyRecord({
    ...row,
    callClassification,
    leadClassification,
    currentClassification
  });

  // 5. Add to results array
  results.push(result);
}
```

3.4 Action Logging Flow



---

4. Algorithms & Data Processing Methods

4.1 Classification Algorithm (2026 Rules)

The core classification logic follows a **dual-metric evaluation system**:

```
┌─────────────────────────────────────────────────────────────────┐
│                  CLASSIFICATION DECISION TREE                   │
├─────────────────────────────────────────────────────────────────┤
│                                                                 │
│  IF CURRENTLY PREMIUM:                                          │
│  ├─ Both metrics Premium range → KEEP PREMIUM ✓                 │
│  ├─ One metric dropped to Standard → KEEP PREMIUM (watching)    │
│  ├─ Both metrics dropped to Standard → DEMOTE TO STANDARD       │
│  └─ Any metric in Pause range → DEMOTE + 14-DAY WARNING ⚠️      │
│                                                                 │
│  IF CURRENTLY STANDARD:                                         │
│  ├─ Both metrics Premium range (30+ days) → UPGRADE TO PREMIUM ↑ │
│  ├─ One metric Premium range → KEEP STANDARD (almost there)     │
│  ├─ Both metrics Standard range → KEEP STANDARD                 │
│  ├─ One metric Pause range → 14-DAY WARNING ⚠️                  │
│  └─ Both metrics Pause range → PAUSE IMMEDIATELY 🛑             │
│                                                                 │
│  CRITICAL RULE: Premium sources NEVER get paused immediately.   │
│  They are demoted to Standard first with 14 days to remediate.  │
│                                                                 │
└─────────────────────────────────────────────────────────────────┘
```

4.2 Metric Relevance Algorithm (10% Threshold)

The platform implements **smart metric handling** to avoid penalizing sources on non-primary metrics:

```
// Determine if a metric is relevant for this source
function isMetricRelevant(metricRevenue: number, totalRevenue: number): boolean {
  const revenueShare = metricRevenue / totalRevenue;
  return revenueShare >= 0.10; // 10% revenue share threshold
}

// If metric contributes <10% of revenue, classify as 'not_primary'
// instead of penalizing for poor performance on secondary metrics
```

4.3 Quality Thresholds by Vertical/Traffic Type

Medicare Thresholds (45+ min calls):

Traffic Type	Metric	Premium Min	Standard Min	Pause Max	Target
Full O&O	Call	9%	6%	5%	10%
Full O&O	Lead	1.5%	0.8%	0.7%	2%
Partial O&O	Call	N/A	7%	6%	8%
Partial O&O	Lead	N/A	0.8%	0.7%	1%
Non O&O	Call	N/A	4%	3%	7%
Non O&O	Lead	N/A	0.5%	0.4%	1%

Health Thresholds (20+ min calls):

Traffic Type	Metric	Premium Min	Standard Min	Pause Max	Target
Full O&O	Call	14%	7%	6%	15%
Full O&O	Lead	9%	5%	4%	9%
Partial O&O	Call	12%	5%	4%	14%
Partial O&O	Lead	7%	3%	2%	7%
Non O&O	Call	N/A	4%	3%	6%
Non O&O	Lead	N/A	2%	1%	3%

Life Thresholds (35+ min calls):

Traffic Type	Metric	Premium Min	Standard Min	Pause Max	Target
Full O&O	Call	10%	6%	5%	12%
Full O&O	Lead	1.5%	0.75%	0.7%	2%
Partial O&O	Call	9%	5%	4%	10%
Partial O&O	Lead	1.5%	0.75%	0.7%	2%
Non O&O	Call	N/A	5%	3%	8%
Non O&O	Lead	N/A	0.5%	0.4%	1%

Auto Thresholds (20+ min calls):

Traffic Type	Metric	Premium Min	Standard Min	Pause Max	Target
Full O&O	Call	25%	20%	19%	26%
Full O&O	Lead	2.5%	1.5%	1.4%	3%
Partial O&O	Call	N/A	15%	14%	16%
Partial O&O	Lead	N/A	1%	0.9%	2%
Non O&O	Call	N/A	10%	9%	11%
Non O&O	Lead	N/A	0.8%	0.7%	1%

Home Thresholds (20+ min calls):

Traffic Type	Metric	Premium Min	Standard Min	Pause Max	Target
Full O&O	Call	25%	20%	19%	26%
Full O&O	Lead	2.5%	1.5%	1.4%	3%
Partial O&O	Call	N/A	10%	9%	11%
Partial O&O	Lead	N/A	1%	0.9%	1%
Non O&O	Call	N/A	10%	9%	10%
Non O&O	Lead	N/A	0.8%	0.7%	1%

Volume Thresholds for Actionability:
- **Call Metrics**: Minimum 50 calls required
- **Lead Metrics**: Minimum 100 leads required

4.4 ML Analytics Algorithms

4.4.1 Anomaly Detection (Z-Score Based)

```
function detectAnomalies(records: ClassificationRecord[]): AnomalyResult[] {
  // Calculate cohort statistics
  const cohortStats = calculateCohortStats(records);

  return records.map(record => {
    const cohort = `${record.vertical}|${record.trafficType}`;
    const stats = cohortStats[cohort];

    // Calculate Z-scores
    const zScores = {
      callQuality: calculateZScore(record.callQualityRate, stats.callQuality),
      leadQuality: calculateZScore(record.leadTransferRate, stats.leadQuality),
      revenue: calculateZScore(record.totalRevenue, stats.revenue)
    };

    // Flag as anomaly if |Z| > 2.0 for any metric
    const isAnomaly = Object.values(zScores).some(z => Math.abs(z) > 2.0);

    return { subId: record.subId, isAnomaly, zScores, ... };
  });
}
```

4.4.2 Performance Clustering (K-Means Inspired)

```
// 5 behavioral clusters based on composite performance score
const CLUSTER_DEFINITIONS = [
  { id: 0, label: 'Star Performers', description: 'Top 10% across all metrics' },
  { id: 1, label: 'Solid Contributors', description: 'Above average, consistent' },
  { id: 2, label: 'Growth Potential', description: 'Average with upward indicators' },
  { id: 3, label: 'Watch List', description: 'Below average, needs attention' },
  { id: 4, label: 'Critical Attention', description: 'At risk of pause' }
];

function assignCluster(record: ClassificationRecord): number {
  const compositeScore = calculateCompositeScore(record);
  // Score thresholds: [0-20, 20-40, 40-60, 60-80, 80-100]
  return Math.min(4, Math.floor(compositeScore / 20));
}
```

4.4.3 Priority Scoring (Impact × Urgency × Confidence)

```
function calculatePriorityScore(item: OpportunityItem): number {
  // Impact: Revenue potential (0-100)
  const impact = (item.potentialRevenue / maxPotentialRevenue) * 100;

  // Urgency: Time sensitivity multiplier
  const urgencyMultiplier = {
    'immediate': 1.5,
    'short-term': 1.2,
    'medium-term': 1.0
  }[item.timeframe];

  // Confidence: Data reliability (0-1)
  const confidence = item.confidenceLevel;

  return (impact * urgencyMultiplier * confidence) / 100;
}
```

4.4.4 Portfolio Health Scoring

```
function calculatePortfolioHealth(records: ClassificationRecord[]): PortfolioHealth {
  const totalRevenue = sum(records.map(r => r.totalRevenue));

  // Revenue at risk = Revenue from sources with pause/warning actions
  const atRiskRevenue = sum(
    records
      .filter(r => ['pause_immediate', 'warning_14_day', 'demote_with_warning'].in-
cludes(r.action))
      .map(r => r.totalRevenue)
  );

  // Diversification = 1 - Herfindahl index
  const revenueShares = records.map(r => r.totalRevenue / totalRevenue);
  const herfindahl = sum(revenueShares.map(s => s * s));
  const diversificationScore = (1 - herfindahl) * 100;

  // Overall health = weighted average
  return {
    overallHealthScore: 0.4 * qualityScore + 0.3 * diversificationScore + 0.3 * (100
- atRiskPercent),
    revenueAtRisk: atRiskRevenue,
    ...
  };
}
```

---

5. Code Architecture & Key Functions

5.1 Directory Structure

```
/home/ubuntu/quality_tier_classifier/nextjs_space/
├── app/                          # Next.js App Router pages
│   ├── api/                      # API route handlers
│   │   ├── actions/route.ts      # Action logging CRUD
│   │   ├── ai-insights/route.ts  # ML analytics endpoint
│   │   ├── classify/route.ts     # Main classification API
│   │   ├── runs/route.ts         # Analysis run history
│   │   └── sql/route.ts          # BigQuery SQL generator
│   ├── history/page.tsx          # Dedicated history log page
│   ├── settings/page.tsx         # Settings/configuration page
│   ├── layout.tsx                # Root layout with ThemeProvider
│   ├── page.tsx                  # Home page (ClassifierClient)
│   └── globals.css               # Global styles + Tailwind
├── components/                   # React components
│   ├── classifier-client.tsx     # Main client component (state management)
│   ├── mapping-step.tsx          # Column mapping UI
│   ├── results-dashboard.tsx     # Results table + tabs + AI insights
│   ├── sql-modal.tsx             # BigQuery SQL dialog
│   ├── theme-context.tsx         # Theme provider + hook
│   ├── theme-provider.tsx        # Wrapper component
│   ├── upload-step.tsx           # File upload UI
│   └── ui/                       # Reusable UI components
├── lib/                          # Core business logic
│   ├── classification-engine.ts  # Classification rules engine
│   ├── db.ts                     # Prisma client singleton
│   ├── ml-analytics.ts           # ML/AI analytics engine
│   ├── quality-targets.ts        # Threshold configurations
│   ├── sql-generator.ts          # BigQuery SQL builder
│   ├── theme-config.ts           # Theme color definitions
│   ├── types.ts                  # TypeScript type definitions
│   └── utils.ts                  # Utility functions
├── prisma/
│   └── schema.prisma             # Database schema definition
└── public/
    └── example_data.csv          # Sample data file
```

5.2 Key TypeScript Interfaces

```typescript
// Primary classification input
export interface ClassificationInput {
  subId: string;
  vertical: string;
  trafficType: string;
  internalChannel: string | null;
  currentClassification?: 'Premium' | 'Standard' | null;
  isUnmapped?: boolean;
  totalCalls: number;
  callsOverThreshold: number;
  callQualityRate?: number | null;
  totalLeadsDialed?: number;
  leadsTransferred?: number;
  leadTransferRate?: number | null;
  totalRevenue?: number;
}

// Classification result output
export interface ClassificationResult {
  currentTier: 'Premium' | 'Standard' | null;
  recommendedTier: 'Premium' | 'Standard' | 'PAUSE';
  action: ActionType;
  actionLabel: string;
  reason: string;
  hasWarning: boolean;
  warningReason?: string;
  callClassification?: MetricClassification;
  leadClassification?: MetricClassification;
  hasInsufficientVolume: boolean;
  isPaused: boolean;
}

// Action types with business meaning
export type ActionType =
  | 'keep_premium'        // Keep at Premium (meeting targets)
  | 'keep_premium_watch'  // Keep at Premium but one metric slipping
  | 'demote_to_standard'  // Premium → Standard (both metrics dropped)
  | 'demote_with_warning' // Premium → Standard + 14-day clock
  | 'upgrade_to_premium'  // Standard → Premium (both metrics Premium)
  | 'keep_standard_close' // Keep Standard, almost Premium
  | 'keep_standard'       // Keep at Standard
  | 'warning_14_day'      // Standard with one metric in Pause
  | 'pause_immediate'     // Standard with BOTH metrics in Pause → STOP
  | 'insufficient_volume' // Not enough data
  | 'review';             // Needs manual review
```

5.3 Core Function Signatures

Classification Engine (`lib/classification-engine.ts`)

```
// Main classification function
export function classifyRecord(input: ClassificationInput): ClassificationResult;

// Metric-level classification
function classifyMetric(
  metricType: MetricType,
  value: number | null,
  volume: number,
  thresholds: TrafficTypeThresholds | null
): MetricClassification;

// Threshold lookup
function getThresholds(vertical: string, trafficType: string): TrafficTypeThresholds
| null;
```

ML Analytics Engine (`lib/ml-analytics.ts`)

```
// Main analytics function
export function runMLAnalytics(records: ClassificationRecord[]): MLInsightsResult {
  return {
    anomalies: detectAnomalies(records),
    clusters: performClustering(records),
    clusterSummary: buildClusterSummary(records),
    riskScores: calculateRiskScores(records),
    peerComparisons: buildPeerComparisons(records),
    revenueImpact: calculateRevenueImpact(records),
    opportunityMatrix: buildOpportunityMatrix(records),
    cohortIntelligence: buildCohortIntelligence(records),
    momentumIndicators: calculateMomentumIndicators(records),
    portfolioHealth: calculatePortfolioHealth(records),
    smartAlerts: generateSmartAlerts(records)
  };
}
```

API Route Handlers

```
// POST /api/classify
export async function POST(request: NextRequest): Promise<NextResponse> {
  const { csvData, mapping, dimension } = await request.json();
  const results = await processClassification(csvData, mapping, dimension);
  return NextResponse.json({ results, stats });
}

// POST /api/actions
export async function POST(request: NextRequest): Promise<NextResponse> {
  const { subId, action, notes, ... } = await request.json();
  const record = await prisma.actionHistory.create({ data: { ... } });
  return NextResponse.json(record);
}

// GET /api/actions?subId=XXX
export async function GET(request: NextRequest): Promise<NextResponse> {
  const history = await prisma.actionHistory.findMany({ ... });
  return NextResponse.json(history);
}
```

5.4 React Component Hierarchy

```
<ThemeProvider>                      # Theme context wrapper
  <Layout>                           # Root layout
    <ClassifierClient>               # Main state manager
      ├─ <UploadStep />              # Step 1: File upload
      ├─ <MappingStep />             # Step 2: Column mapping
      └─ <ResultsDashboard>          # Step 3: Results display
           ├─ <NavigationTabs />     # Top navigation
           ├─ <FilterBar />          # Filters + actions
           ├─ <ClassificationTable>
           │    └─ <ExpandedRowContent>
           │         ├─ <AnalysisTab />
           │         ├─ <RevenueTab />
           │         ├─ <NotesTab />
           │         └─ <LogActionTab />
           ├─ <AIInsightsPanel>     # Smart Insights tab
           │    ├─ <ClusterOverview />
           │    ├─ <ClusterDrillDown />
           │    └─ <PriorityMatrix />
           └─ <EmbeddedHistoryPanel>  # History tab
```

---

6. Platform Capabilities & Scope

6.1 Supported Verticals

Vertical	Call Duration Threshold	Special Rules
Medicare	45+ minutes (2700s)	Highest thresholds, strictest quality standards
Life	35+ minutes (2100s)	Second highest thresholds
Health	20+ minutes (1200s)	Standard thresholds, Partial O&O has Premium
Auto	20+ minutes (1200s)	Higher call quality require-ments (20-25%)
Home	20+ minutes (1200s)	Higher call quality require-ments (20-25%)

6.2 Supported Traffic Types

Type	Premium Available	Description
Full O&O	✓ Yes (All Verticals)	Owned & Operated, highest quality expectations
Partial O&O	✓ Yes (Health, Life)	Mixed ownership; Premium available for Health & Life
Partial O&O	✗ No (Medicare, Auto, Home)	Mixed ownership; Standard max for these verticals
Non O&O	✗ No	Third-party traffic, Standard tier maximum

6.3 Aggregation Dimensions

Dimension	Group By Fields	Use Case
`sub_id`	Individual sub ID	Most granular analysis
`source_name`	Source + Vertical + Traffic + Channel	Publisher-level view
`placement`	Placement + Vertical + Traffic + Channel	Placement optimization
`media_type`	Media Type + Vertical + Traffic + Channel	Channel mix analysis
`overall`	Vertical + Traffic + Channel	High-level summary

6.4 Metric Coverage

Quality Metrics:
- Call Quality Rate (% of calls meeting duration threshold)
- Lead Transfer Rate (% of leads successfully transferred)

Volume Metrics:
- Total Calls / Paid Calls / Calls Over Threshold
- Total Leads Dialed / Leads Transferred
- Click Volume
- Redirect Volume

Revenue Metrics:
- Call Revenue / Lead Revenue / Click Revenue / Redirect Revenue
- Total Revenue
- RP (Revenue Per) calculations: RPQCall, RPLead, RPClick, RPRedir

6.5 Export Capabilities

1. **CSV Export**: Download filtered results as CSV
2. **BigQuery SQL**: Generate extraction query for custom date ranges
3. **Action History**: Export audit trail

6.6 Filtering Options

- By Vertical (Medicare, Health, Home, Life, Auto)
- By Traffic Type (Full O&O, Partial O&O, Non O&O)
- By Media Type (dynamic based on data)
- By Action (Pause, Promote, Demote, Below MIN, Correct, Review)
- By Date Range (history page)
- By User (who logged action)
- Search by Sub ID or notes

---

7. Backend Composition

7.1 Database Schema (PostgreSQL + Prisma)

```
// Analysis run metadata
model AnalysisRun {
  id            String   @id @default(cuid())
  name          String?
  startDate     String
  endDate       String
  fileName      String?
  totalRecords  Int
  promoteCount  Int      @default(0)
  demoteCount   Int      @default(0)
  belowMinCount Int      @default(0)
  correctCount  Int      @default(0)
  reviewCount   Int      @default(0)
  createdAt     DateTime @default(now())
  updatedAt     DateTime @updatedAt
  results       ClassificationResult[]
}

// Individual classification records
model ClassificationResult {
  id                   String      @id @default(cuid())
  runId                String
  run                  AnalysisRun @relation(fields: [runId], references: [id], onDel
ete: Cascade)
  subId                String
  vertical             String
  trafficType          String
  currentTier          Int?
  currentTierLabel     String?
  recommendedTier      String
  recommendedTierNum   Int?
  action               String
  actionLabel          String
  totalCalls           Int         @default(0)
  callsOverThreshold   Int         @default(0)
  callQualityRate      Float?
  totalLeads           Int         @default(0)
  totalRevenue         Float       @default(0)
  classificationReason String?
  createdAt            DateTime    @default(now())

  @@index([runId])
  @@index([action])
  @@index([vertical])
}

// Action audit trail
model ActionHistory {
  id              String   @id @default(cuid())
  subId           String
  vertical        String
  trafficType     String
  mediaType       String?  // For filtering
  actionTaken     String   // promote, demote, pause, etc.
  actionLabel     String   // Human-readable label
  previousState   String?  // Premium, Standard, etc.
  newState        String?  // Premium, Standard, PAUSED
  metricMode      String?  // call, lead, both
  callQuality     Float?
  leadQuality     Float?
  totalRevenue    Float?
  notes           String?  // User notes
```

```
  takenBy         String?  // User who took action
  createdAt       DateTime @default(now())

  @@index([subId])
  @@index([vertical])
  @@index([trafficType])
  @@index([takenBy])
  @@index([createdAt])
}
```

7.2 API Endpoints

Method	Endpoint	Purpose
POST	`/api/classify`	Process CSV and return classifications
GET	`/api/runs`	List recent analysis runs
GET	`/api/runs/[id]`	Get specific run details
POST	`/api/actions`	Log an action
GET	`/api/actions`	Get action history
POST	`/api/sql`	Generate BigQuery SQL
POST	`/api/ai-insights`	Generate ML analytics

7.3 Environment Variables

```
# Database
DATABASE_URL="postgresql://user:pass@host:5432/dbname"

# Next.js
NEXTAUTH_SECRET="..."
NEXTAUTH_URL="https://equalitybeta.abacusai.app"

# Optional: LLM API (for enhanced AI insights)
ABACUSAI_API_KEY="..."
```

---

8. Technology Stack

8.1 Frontend

Technology	Version	Purpose
React	18.2.0	UI library
Next.js	14.2.28	Full-stack framework (App Router)
TypeScript	5.2.2	Type safety
Tailwind CSS	3.3.3	Utility-first styling
Ant Design Icons	6.1.0	Icon library
Radix UI	Various	Accessible UI primitives
Framer Motion	10.18.0	Animations

8.2 Backend

Technology	Version	Purpose
Next.js API Routes	14.2.28	REST API endpoints
Prisma	6.7.0	Database ORM
PostgreSQL	Latest	Relational database
Node.js	20.x	Runtime environment

8.3 Build & Development

Tool	Version	Purpose
Yarn	1.22.x	Package manager
ESLint	9.24.0	Code linting
PostCSS	8.4.30	CSS processing
tsx	4.20.3	TypeScript execution

8.4 Deployment

- **Platform**: Abacus.AI App Hosting
- **Build**: `next build` with standalone output
- **Domain**: `equalitybeta.abacusai.app`

- **Environment**: Production with PostgreSQL database

Appendix A: Platform BigQuery SQL

The following SQL is generated by the platform (`lib/sql-generator.ts`) for data extraction:

```sql
-- Sub ID Performance Report - 30-Day Rolling Window (ENHANCED + OUTBOUND DIAL QUAL-
ITY)
-- Window: 30 days ending YESTERDAY (excludes today)
-- Tables: unified_revenue + reference.sub_ids
-- Includes:
--   - calls_over_threshold with vertical-specific duration thresholds
--   - Outbound dial quality metrics for lead sub_ids (calls linked via session_id)

WITH date_params AS (
  SELECT
    DATE_SUB(CURRENT_DATE(), INTERVAL 1 DAY) AS end_date,
    DATE_SUB(CURRENT_DATE(), INTERVAL 31 DAY) AS start_date
),

-- Get latest snapshot of sub_id reference data
sub_id_reference AS (
  SELECT
    subid,
    tier,
    description,
    channel,
    traffic_type,
    vertical_name,
    source_name,
    media_type_name,
    campaign_type
  FROM `dwh-production-352519.reference.sub_ids`
  WHERE snapshot_date = (
    SELECT MAX(snapshot_date)
    FROM `dwh-production-352519.reference.sub_ids`
  )
),

-- Step 1: Get ALL leads with their session_ids by sub_id
leads_by_subid AS (
  SELECT
    sub_id,
    vertical,
    session_id,
    user_id,
    1 AS lead_count
  FROM `dwh-production-352519.unified.unified_revenue`, date_params
  WHERE date_platform BETWEEN date_params.start_date AND date_params.end_date
    AND transaction_type = 'Lead'
    AND vertical IN ('Medicare', 'Health', 'Life', 'Auto', 'Home')
    AND session_id IS NOT NULL
),

-- Step 2: Get OUTBOUND calls only (these are dials on leads)
outbound_calls_data AS (
  SELECT
    sub_id AS call_sub_id,
    vertical AS call_vertical,
    session_id,
    user_id,
    call_transfers,
    paid_calls,
    call_duration,
    revenue AS call_revenue
  FROM `dwh-production-352519.unified.unified_revenue`, date_params
  WHERE date_platform BETWEEN date_params.start_date AND date_params.end_date
    AND transaction_type = 'Call'
```

```
    AND call_category = 'Outbound'  -- Only outbound dials
    AND vertical IN ('Medicare', 'Health', 'Life', 'Auto', 'Home')
),

-- Step 3: Link OUTBOUND calls back to the LEAD's sub_id via session_id
-- Outbound Transfer Rate = Outbound Transfers / Total Leads
outbound_dials_on_leads AS (
  SELECT
    l.sub_id,
    l.vertical,
    -- Denominator: Total leads
    COUNT(DISTINCT l.session_id) AS lead_count,
    -- Numerator: Outbound transferred calls linked to those leads
    SUM(COALESCE(c.call_transfers, 0)) AS outbound_transfers,
    SUM(COALESCE(c.paid_calls, 0)) AS outbound_paid_calls,
    -- Outbound calls over duration threshold
    SUM(CASE
      WHEN l.vertical = 'Medicare' AND c.call_duration >= 2700 THEN c.call_transfers
      WHEN l.vertical = 'Life' AND c.call_duration >= 2100 THEN c.call_transfers
      WHEN l.vertical IN ('Health', 'Auto', 'Home') AND c.call_duration >= 1200 THEN
c.call_transfers
      ELSE 0
    END) AS outbound_calls_over_threshold
  FROM leads_by_subid l
  LEFT JOIN outbound_calls_data c
    ON l.session_id = c.session_id  -- Link via session_id
  GROUP BY l.sub_id, l.vertical
),

-- Step 4: Direct metrics by sub_id (calls, leads, clicks, redirects, revenue)
direct_metrics AS (
  SELECT
    sub_id,
    vertical,

    -- Top placement by revenue for this sub_id
    ARRAY_AGG(placement ORDER BY revenue DESC LIMIT 1)[SAFE_OFFSET(0)] AS top_placemen
t,

    -- Top channel by revenue for this sub_id (from unified_revenue, not reference
table)
    ARRAY_AGG(channel ORDER BY revenue DESC LIMIT 1)[SAFE_OFFSET(0)] AS top_channel,

    -- Direct Call metrics (calls where this sub_id is the call's sub_id)
    SUM(CASE WHEN transaction_type = 'Call' THEN call_transfers ELSE 0 END) AS dir-
ect_calls,
    SUM(CASE WHEN transaction_type = 'Call' THEN paid_calls ELSE 0 END) AS dir-
ect_paid_calls,
    SUM(CASE
      WHEN transaction_type = 'Call' AND vertical = 'Medicare' AND call_duration >= 27
00 THEN call_transfers
      WHEN transaction_type = 'Call' AND vertical = 'Life' AND call_duration >= 2100 T
HEN call_transfers
      WHEN transaction_type = 'Call' AND vertical IN ('Health', 'Auto', 'Home') AND ca
ll_duration >= 1200 THEN call_transfers
      ELSE 0
    END) AS direct_calls_over_threshold,

    -- Lead metrics
    SUM(CASE WHEN transaction_type = 'Lead' THEN 1 ELSE 0 END) AS total_leads,
    SUM(CASE WHEN transaction_type = 'Lead' THEN COALESCE(revenue, 0) ELSE 0 END) AS l
ead_revenue,
```

```
    -- Click metrics
    SUM(CASE WHEN transaction_type = 'Click' THEN COALESCE(clicks, 0) ELSE 0 END) AS t
otal_clicks,
    SUM(CASE WHEN transaction_type = 'Click' THEN COALESCE(revenue, 0) ELSE 0 END) AS
click_revenue,

    -- Call revenue
    SUM(CASE WHEN transaction_type = 'Call' THEN COALESCE(revenue, 0) ELSE 0 END) AS c
all_revenue,

    -- Redirect metrics
    SUM(CASE WHEN transaction_type = 'Redirect' THEN 1 ELSE 0 END) AS redirect_volume,
    SUM(CASE WHEN transaction_type = 'Redirect' THEN COALESCE(revenue, 0) ELSE 0 END)
AS redirect_revenue,

    -- Total Revenue (all transaction types)
    SUM(COALESCE(revenue, 0)) AS total_revenue

  FROM `dwh-production-352519.unified.unified_revenue`, date_params
  WHERE date_platform BETWEEN date_params.start_date AND date_params.end_date
    AND vertical IN ('Medicare', 'Health', 'Life', 'Auto', 'Home')
  GROUP BY sub_id, vertical
)

SELECT
  -- ===== CORE FIELDS =====
  COALESCE(s.subid, d.sub_id) AS sub_id,
  CASE s.tier WHEN 1 THEN 'Premium' WHEN 2 THEN 'Standard' ELSE '' END AS intern-
al_channel,
  COALESCE(s.traffic_type, 'Unknown') AS traffic_type,
  COALESCE(s.vertical_name, d.vertical) AS vertical,

  -- ===== CALL QUALITY FIELDS =====
  d.direct_calls AS total_calls,
  d.direct_paid_calls AS paid_calls,
  ROUND(SAFE_DIVIDE(d.direct_paid_calls, d.direct_calls), 4) AS qr_rate,
  d.direct_calls_over_threshold AS calls_over_threshold,
  ROUND(SAFE_DIVIDE(d.direct_calls_over_threshold, d.direct_paid_calls), 4) AS call_qu
ality_rate,

  -- ===== LEAD QUALITY FIELDS (OB Transfer Rate) =====
  d.total_leads AS total_leads_dialed,
  COALESCE(o.outbound_transfers, 0) AS leads_transferred,
  ROUND(SAFE_DIVIDE(o.outbound_transfers, d.total_leads), 4) AS lead_transfer_rate,

  -- ===== METADATA FIELDS =====
  d.top_placement AS placement,
  d.top_channel AS channel,
  s.description,
  s.source_name,
  s.media_type_name AS media_type,
  s.campaign_type,

  -- ===== VOLUME METRICS =====
  d.total_leads AS lead_volume,
  d.direct_calls AS call_volume,
  d.total_clicks AS click_volume,
  d.redirect_volume,

  -- ===== REVENUE BY TYPE =====
  ROUND(d.lead_revenue, 2) AS lead_revenue,
  ROUND(d.call_revenue, 2) AS call_revenue,
  ROUND(d.click_revenue, 2) AS click_revenue,
```

```
  ROUND(d.redirect_revenue, 2) AS redirect_revenue,

  -- ===== RP METRICS =====
  ROUND(SAFE_DIVIDE(d.lead_revenue, d.total_leads), 2) AS rp_lead,
  ROUND(SAFE_DIVIDE(d.call_revenue, d.direct_paid_calls), 2) AS rp_qcall,
  ROUND(SAFE_DIVIDE(d.click_revenue, d.total_clicks), 2) AS rp_click,
  ROUND(SAFE_DIVIDE(d.redirect_revenue, d.redirect_volume), 2) AS rp_redirect,

  -- Total Revenue
  ROUND(d.total_revenue, 2) AS total_revenue

FROM direct_metrics d
LEFT JOIN sub_id_reference s ON d.sub_id = s.subid
LEFT JOIN outbound_dials_on_leads o ON d.sub_id = o.sub_id AND d.vertical = o.vertical

WHERE d.sub_id IS NOT NULL
  AND d.sub_id != ''
  AND LOWER(d.sub_id) != 'unknown'

ORDER BY total_revenue DESC
```

Key SQL Features:
- **30-Day Rolling Window**: Automatically calculates dates ending yesterday
- **Vertical-Specific Duration Thresholds**: Medicare ≥45min, Life ≥35min, Health/Auto/Home ≥20min
- **Outbound Dial Quality**: Links outbound calls to leads via `session_id` for accurate transfer rate calculation
- **Comprehensive Metrics**: Call, Lead, Click, Redirect volumes and revenues
- **RP Calculations**: Revenue Per Lead, Revenue Per Qualified Call, Revenue Per Click, Revenue Per Redirect

---

Appendix B: Action Type Reference

Action	Label	Meaning	Color
`keep_premium`	✓ Premium	Meeting all Premium targets	Green
`keep_premium_watch`	✓ Premium (Watch)	Premium but one metric slipping	Green/Yellow
`upgrade_to_premium`	↑ Promote	Eligible for upgrade to Premium	Blue
`keep_standard`	✓ Standard	Meeting Standard targets	Purple
`keep_standard_close`	✓ Standard (Close)	Almost Premium eligible	Purple
`demote_to_standard`	↓ Demote	Dropping from Premium to Standard	Orange
`demote_with_warning`	↓ Demote + 14d Warning	Demote with remediation period	Orange
`warning_14_day`	⚠ Warning	14-day warning period started	Yellow
`pause_immediate`	🛑 PAUSE TODAY	Immediate pause required	Red
`insufficient_volume`	📊 Low Volume	Not enough data to classify	Gray
`review`	👀 Review	Needs manual review	Purple

Document Version: 2026.1
Generated: January 22, 2026
Platform: Quality Tier Classifier by Abacus.AI