BTL Calculator - System Architecture

High-Level System Architecture

```
graph TB
    subgraph "Client Layer"
        Browser[Web Browser]
    end

    subgraph "Frontend - React + Vite"
        UI[React Components]
        Router[React Router]
        Context[Context Providers]
        CalcEngine[Calculation Engines]

        UI --> Router
        UI --> Context
        UI --> CalcEngine
    end

    subgraph "Backend - Express API"
        API[Express Server]
        Auth[Auth Middleware]
        RateLimit[Rate Limiter]
        Routes[API Routes]

        API --> Auth
        API --> RateLimit
        API --> Routes
    end

    subgraph "Database"
        Supabase[(Supabase PostgreSQL)]
        Tables[Tables: rates, quotes, users, app_constants]
        RLS[Row Level Security]

        Supabase --> Tables
        Supabase --> RLS
    end

    subgraph "External Services"
        PostcodeAPI[Postcode Lookup API]
        PDFGen[PDF Generation]
    end

    Browser -->|HTTPS| UI
    UI -->|API Calls| API
    Routes -->|Queries| Supabase
    Auth -->|Validate| Supabase
    Routes -->|Lookup| PostcodeAPI
    Routes -->|Generate| PDFGen

    style Browser fill:#e1f5ff
    style UI fill:#fff4e1
    style API fill:#e8f5e9
    style Supabase fill:#f3e5f5
```

Frontend Architecture

```
graph LR
    subgraph "Entry Point"
        Index[index.jsx]
        App[App.jsx]
    end

    subgraph "Routing & Layout"
        Router[React Router]
        Layout[Layout Components]
        Nav[Navigation]
    end

    subgraph "Pages"
        Calculator[Calculator Page]
        Admin[Admin Pages]
        Settings[Settings Page]
        Quotes[Quotes Page]
    end

    subgraph "Components"
        BTLCalc[BTL Calculator]
        BridgeCalc[Bridge Calculator]
        Results[Results Display]
        Forms[Form Inputs]
    end

    subgraph "State Management"
        SupabaseCtx[Supabase Context]
        AuthCtx[Auth Context]
        ThemeCtx[Theme Context]
    end

    subgraph "Business Logic"
        BTLEngine[BTL Calculation Engine]
        BridgeEngine[Bridge Calculation Engine]
        Utils[Utility Functions]
    end

    Index --> App
    App --> Router
    Router --> Layout
    Layout --> Pages
    Pages --> Components
    Components --> SupabaseCtx
    Components --> AuthCtx
    Components --> BTLEngine
    Components --> BridgeEngine

    style Index fill:#ffebee
    style BTLEngine fill:#e8f5e9
    style SupabaseCtx fill:#fff3e0
```

Backend API Architecture

```
graph TB
    subgraph "Server Entry"
        Server[server.js]
        Config[Configuration]
        Middleware[Middleware Stack]
    end

    subgraph "Middleware Layer"
        CORS[CORS Handler]
        Logger[HTTP Logger]
        RateLimit[Rate Limiter]
        ErrorH[Error Handler]
    end

    subgraph "API Routes"
        RatesRoute[API Rates]
        QuotesRoute[API Quotes]
        AuthRoute[API Auth]
        PostcodeRoute[API Postcode]
        DIPRoute[API DIP PDF]
        QuotePDFRoute[API Quote PDF]
        ExportRoute[API Export]
    end

    subgraph "Database Layer"
        SupabaseClient[Supabase Client]
        RatesTable[(rates table)]
        QuotesTable[(quotes table)]
        UsersTable[(users table)]
        ConstantsTable[(app_constants table)]
    end

    Server --> Config
    Server --> Middleware
    Middleware --> CORS
    Middleware --> Logger
    Middleware --> RateLimit
    Middleware --> ErrorH

    Server --> RatesRoute
    Server --> QuotesRoute
    Server --> AuthRoute
    Server --> PostcodeRoute
    Server --> DIPRoute
    Server --> QuotePDFRoute
    Server --> ExportRoute

    RatesRoute --> SupabaseClient
    QuotesRoute --> SupabaseClient
    AuthRoute --> SupabaseClient

    SupabaseClient --> RatesTable
    SupabaseClient --> QuotesTable
    SupabaseClient --> UsersTable
    SupabaseClient --> ConstantsTable

    style Server fill:#e3f2fd
    style Middleware Layer fill:#fff3e0
    style API Routes fill:#e8f5e9
    style Database Layer fill:#f3e5f5
```

Calculation Engine Architecture

```
graph TB
    subgraph "User Input"
        FormData[Form Data]
        PropertyValue[Property Value]
        Rent[Monthly Rent]
```

```
    ProductType[Product Type]
end

subgraph "Core Engine Selection"
    Router{Product Type?}
end

subgraph "BTL Calculation Engine"
    BTLInit[Initialize BTL Engine]
    BTLCompute[Compute Loan]
    BTLEval[Evaluate Scenarios]
    BTLResult[Best Loan Result]

    BTLInit --> BTLCompute
    BTLCompute --> BTLEval
    BTLEval --> BTLResult
end

subgraph "Bridge/Fusion Engine"
    BridgeInit[Initialize Bridge Engine]
    BridgeCompute[Compute Loan]
    BridgeEval[Evaluate Properties]
    BridgeResult[Best Loan Result]

    BridgeInit --> BridgeCompute
    BridgeCompute --> BridgeEval
    BridgeEval --> BridgeResult
end

subgraph "Rate Logic"
    CoreRates[Core Rates]
    SpecialistRates[Specialist Rates]
    FloorRate[Floor Rate - Core Only]
    StressBBR[Stress BBR - Both]

    CoreRates --> FloorRate
    CoreRates --> StressBBR
    SpecialistRates --> StressBBR
end

subgraph "Calculations"
    GrossLoan[Gross Loan - Uses Floor + Stress]
    PayRate[Pay Rate - Uses Actual Rate]
    ICR[ICR Display - Uses Actual Rate]
    DirectDebit[Direct Debit - Uses Actual Rate]
end

subgraph "Output"
    DisplayResults[Display Results]
    PayRateDisplay[Pay Rate: Actual - Deferred]
    RateDisplay[Rate: Actual + BBR for Tracker]
end

FormData --> Router
PropertyValue --> Router
Rent --> Router
ProductType --> Router

Router -->|BTL| BTLInit
Router -->|Bridge/Fusion| BridgeInit

BTLResult --> CoreRates
BTLResult --> SpecialistRates
BridgeResult --> CoreRates
BridgeResult --> SpecialistRates

FloorRate --> Calculations
StressBBR --> Calculations
```

```
    GrossLoan --> DisplayResults
    PayRate --> PayRateDisplay
    ICR --> DisplayResults
    DirectDebit --> DisplayResults

    PayRateDisplay --> DisplayResults
    RateDisplay --> DisplayResults

    style Router fill:#fff3e0
    style Rate Logic fill:#e8f5e9
    style Calculations fill:#e1f5ff
    style Output fill:#f3e5f5
```

Data Flow Diagram

```
sequenceDiagram
    participant User
    participant Frontend
    participant CalcEngine
    participant Backend
    participant Supabase

    User->>Frontend: Enter loan details
    Frontend->>Backend: GET /api/rates
    Backend->>Supabase: Query rates table
    Supabase-->>Backend: Return rates data
    Backend-->>Frontend: Rates JSON

    Frontend->>CalcEngine: Calculate with inputs
    CalcEngine->>CalcEngine: Evaluate scenarios
    CalcEngine->>CalcEngine: Apply floor rate (Core only)
    CalcEngine->>CalcEngine: Apply stress BBR
    CalcEngine->>CalcEngine: Compute gross loan
    CalcEngine->>CalcEngine: Compute payments (actual rate)
    CalcEngine-->>Frontend: Best loan result

    Frontend->>User: Display results

    User->>Frontend: Save quote
    Frontend->>Backend: POST /api/quotes
    Backend->>Supabase: Insert quote
    Supabase-->>Backend: Success
    Backend-->>Frontend: Quote saved
    Frontend->>User: Confirmation
```

Database Schema

```
erDiagram
    rates {
        text key PK
        jsonb data
        timestamp created_at
    }

    quotes {
        uuid id PK
        uuid user_id FK
        text reference_number
        text loan_type
        jsonb btl_quote_data
        jsonb bridging_quote_data
        jsonb client_details
        timestamp created_at
    }

    users {
        uuid id PK
        text email
        text password_hash
        text company_name
        timestamp created_at
    }

    app_constants {
        uuid id PK
        text key
        jsonb value
        jsonb product_lists
        jsonb fee_columns
        jsonb market_rates
        timestamp updated_at
    }

    users ||--o{ quotes : creates
```

Component Hierarchy

```
graph TB
    App[App.jsx]

    subgraph "Layout"
        Header[Header]
        Sidebar[Sidebar]
        Footer[Footer]
    end

    subgraph "Calculator Pages"
        BTLPage[BTL Calculator Page]
        BridgePage[Bridge Calculator Page]
    end

    subgraph "BTL Components"
        BTLForm[BTL Form]
        BTLResults[BTL Results]
        BTLBreakdown[BTL Breakdown]
    end

    subgraph "Bridge Components"
        BridgeForm[Bridge Form]
        BridgeResults[Bridge Results]
        BridgeBreakdown[Bridge Breakdown]
    end

    subgraph "Admin Components"
        Constants[Constants Manager]
        RatesAdmin[Rates Admin]
        UsersAdmin[Users Admin]
    end

    subgraph "Shared Components"
        Modal[Modal]
        Button[Button]
        Input[Input Fields]
        Card[Card]
    end

    App --> Header
    App --> Sidebar
    App --> Footer
    App --> BTLPage
    App --> BridgePage

    BTLPage --> BTLForm
    BTLPage --> BTLResults
    BTLPage --> BTLBreakdown

    BridgePage --> BridgeForm
    BridgePage --> BridgeResults
    BridgePage --> BridgeBreakdown

    App --> Constants
    App --> RatesAdmin

    BTLForm --> Modal
    BTLForm --> Button
    BridgeForm --> Modal
    Constants --> Modal

    style App fill:#ffebee
    style Header fill:#e3f2fd
    style BTLPage fill:#e8f5e9
    style Modal fill:#fff3e0
```

Security Architecture

```
graph TB
    subgraph "Frontend Security"
        AuthCheck[Auth Check]
        TokenStore[Token Storage]
        RouteGuard[Route Guards]
    end

    subgraph "Backend Security"
        RateLimit[Rate Limiting]
        CORS[CORS Policy]
        AuthMiddleware[Auth Middleware]
        Validation[Input Validation]
    end

    subgraph "Database Security"
        RLS[Row Level Security]
        EncryptedFields[Encrypted Fields]
        Policies[Access Policies]
    end

    subgraph "API Security"
        HTTPS[HTTPS Only]
        APIKeys[API Keys]
        ServiceRole[Service Role Key]
    end

    AuthCheck --> TokenStore
    TokenStore --> RouteGuard

    RateLimit --> CORS
    CORS --> AuthMiddleware
    AuthMiddleware --> Validation

    Validation --> RLS
    RLS --> EncryptedFields
    EncryptedFields --> Policies

    HTTPS --> APIKeys
    APIKeys --> ServiceRole

    style Frontend Security fill:#e8f5e9
    style Backend Security fill:#fff3e0
    style Database Security fill:#f3e5f5
    style API Security fill:#e1f5ff
```

Deployment Architecture

```
graph TB
    subgraph "Development"
        DevFE[Local Frontend<br/>Vite Dev Server<br/>Port 3000]
        DevBE[Local Backend<br/>Express Server<br/>Port 3001]
    end

    subgraph "Production"
        ProdFE[Vercel<br/>Frontend Deployment<br/>CDN + Edge Functions]
        ProdBE[Vercel/Render<br/>Backend API<br/>Serverless/Container]
    end

    subgraph "Database"
        SupabaseDB[Supabase<br/>PostgreSQL<br/>Managed Cloud]
    end

    subgraph "Version Control"
        GitHub[GitHub Repository<br/>main branch]
    end

    GitHub -->|Deploy| ProdFE
    GitHub -->|Deploy| ProdBE

    DevFE -.->|API Calls| DevBE
    DevBE -.->|Queries| SupabaseDB

    ProdFE -->|API Calls| ProdBE
    ProdBE -->|Queries| SupabaseDB

    style Development fill:#e8f5e9
    style Production fill:#e3f2fd
    style Database fill:#f3e5f5
    style Version Control fill:#fff3e0
```

Key Architecture Principles

1. Separation of Concerns

- Frontend handles UI and user interaction
- Backend handles API, authentication, and data access
- Calculation engines are pure business logic (no UI)

2. Rate Calculation Rules

- **Floor Rate**: Applied ONLY to core products for gross loan calculation
- **Stress BBR**: Applied to BOTH core and specialist products for gross loan calculation
- **Actual Rate**: Used for all payment calculations and display
- **Pay Rate Display**: Always shows `actual rate - deferred rate` (no floor)

3. Security Layers

- Frontend: Route guards, auth context
- Backend: Rate limiting, CORS, auth middleware
- Database: Row Level Security (RLS), encrypted fields
- Communication: HTTPS only, token-based auth

4. Scalability

- Stateless API design
- Client-side calculation engines (reduces server load)
- Database connection pooling
- Rate limiting prevents abuse

5. Maintainability

- Modular component structure
- Separate calculation engines per product type
- Centralized configuration (app_constants)
- Environment-based configuration