Architecture Diagrams

System Overview

```
graph TB
    subgraph UserLayer["User Layer"]
        Parent[Parent User]
        Child1[Child 1 - אדם]
        Child2[Child 2 - ג'ון]
    end

    subgraph FrontendLayer["Frontend Layer (Vercel)"]
        ReactApp[React Application]
        Components[UI Components]
        Charts[Chart.js Charts]
        APIClient[API Client]
    end

    subgraph BackendLayer["Backend Layer (Railway)"]
        ExpressAPI[Express API Server]
        Routes[API Routes]
        BusinessLogic[Business Logic]
    end

    subgraph DataLayer["Data Layer (MongoDB Atlas)"]
        MongoDB[(MongoDB Database)]
        ChildrenCollection[Children Collection]
        TransactionsData[Transactions Data]
    end

    Parent -->|HTTPS| ReactApp
    Child1 -->|HTTPS| ReactApp
    Child2 -->|HTTPS| ReactApp

    ReactApp --> Components
    ReactApp --> Charts
    ReactApp --> APIClient

    APIClient -->|REST API| ExpressAPI
    ExpressAPI --> Routes
    Routes --> BusinessLogic
    BusinessLogic --> MongoDB

    MongoDB --> ChildrenCollection
    MongoDB --> TransactionsData

    style ReactApp fill:#667eea
    style ExpressAPI fill:#10b981
    style MongoDB fill:#f59e0b
```

Component Architecture

```
graph TB
    subgraph App["App.jsx"]
        Router[View Router]
        Nav[Navigation Bar]
    end

    subgraph ParentComponents["Parent Components"]
        Login[ParentLogin]
        Dashboard[ParentDashboard]
        BalanceOverview[Balance Overview]
        TransactionForm[Transaction Form]
        TransactionList[Transaction List]
    end

    subgraph ChildComponents["Child Components"]
        ChildView[ChildView]
        BalanceDisplay[BalanceDisplay]
        PieCharts[Expense Pie Charts]
        RecentTransactions[Recent Transactions]
    end

    subgraph SharedComponents["Shared Components"]
        BalanceDisplayShared[BalanceDisplay]
        TransactionListShared[TransactionList]
    end

    subgraph Utils["Utils"]
        APIClient[API Client]
        Storage[Storage Utils]
    end

    Router --> Login
    Router --> Dashboard
    Router --> ChildView

    Dashboard --> BalanceOverview
    Dashboard --> TransactionForm
    Dashboard --> TransactionList

    ChildView --> BalanceDisplayShared
    ChildView --> PieCharts
    ChildView --> RecentTransactions

    TransactionForm --> APIClient
    TransactionList --> APIClient
    PieCharts --> APIClient

    style Dashboard fill:#667eea
    style ChildView fill:#ec4899
    style APIClient fill:#10b981
```

Data Flow - Complete Transaction Lifecycle

```
sequenceDiagram
    participant P as Parent
    participant F as Frontend
    participant API as Backend API
    participant DB as MongoDB

    P->>F: Open Parent Dashboard
    F->>API: GET /api/children
    API->>DB: Query children collection
    DB-->>API: All children data
    API-->>F: Children data (balances)
    F->>P: Display balances

    P->>F: Fill transaction form
    Note over P,F: Type: Expense<br/>Amount: 50<br/>Category: משחקים
    P->>F: Submit form
    F->>API: POST /api/transactions
    Note over F,API: {childId, type, amount, category}

    API->>DB: Find child by ID
    DB-->>API: Child document

    API->>API: Create transaction object
    API->>API: Add to transactions array
    API->>API: Recalculate balance

    API->>DB: Update child document
    Note over API,DB: {balance, transactions}
    DB-->>API: Update confirmed

    API-->>F: Transaction + new balance
    F->>F: Refresh UI
    F->>P: Show updated balance

    Note over F,DB: Auto-sync to other devices
    F->>API: GET /api/children/:childId
    API->>DB: Fetch updated data
    DB-->>API: Latest data
    API-->>F: Updated balance
```

Expense Analytics Flow

```
sequenceDiagram
    participant C as Child
    participant F as Frontend
    participant API as Backend API
    participant DB as MongoDB
    participant Chart as Chart.js

    C->>F: Open Child View
    F->>API: GET /api/children/:childId
    API->>DB: Fetch child data
    DB-->>API: Child + transactions
    API-->>F: Child data

    F->>API: GET /api/children/:childId/expenses-by-category?days=7
    API->>DB: Fetch transactions
    DB-->>API: All transactions
    API->>API: Filter expenses (last 7 days)
    API->>API: Group by category
    API->>API: Sum amounts per category
    API-->>F: Expenses by category (7 days)

    F->>API: GET /api/children/:childId/expenses-by-category?days=30
    API->>API: Filter expenses (last 30 days)
    API->>API: Group by category
    API-->>F: Expenses by category (30 days)

    F->>Chart: Render pie chart (7 days)
    F->>Chart: Render pie chart (30 days)
    Chart-->>F: Visual charts
    F->>C: Display dashboard with charts

    Note over F: Auto-refresh every 5 seconds
```

Database Schema

```
erDiagram
    CHILDREN ||--o{ TRANSACTIONS : has

    CHILDREN {
        string _id PK
        string name
        number balance
        array transactions
    }

    TRANSACTIONS {
        string id PK
        datetime date
        string type
        number amount
        string description
        string category
        string childId FK
    }
```

Deployment Flow

```
graph LR
    subgraph Development
        Local[Local Code]
        Local -->|git push| GitHub
    end

    subgraph SourceControl
        GitHub[GitHub Repository]
    end

    subgraph FrontendDeploy
        GitHub -->|Webhook| Vercel
        Vercel -->|Build| ViteBuild[Vite Build]
        ViteBuild -->|Deploy| VercelCDN[Vercel CDN]
    end

    subgraph BackendDeploy
        GitHub -->|Webhook| Railway
        Railway -->|Build| NPMInstall[npm install]
        NPMInstall -->|Start| NodeServer[Node.js Server]
    end

    subgraph Database
        MongoDBAtlas[(MongoDB Atlas)]
        NodeServer -->|Connect| MongoDBAtlas
    end

    subgraph Users
        Browser[User Browser]
        Browser -->|HTTPS| VercelCDN
        Browser -->|API Calls| NodeServer
    end

    style VercelCDN fill:#000000,color:#fff
    style NodeServer fill:#0E0E0E,color:#fff
    style MongoDBAtlas fill:#13AA52,color:#fff
```

Request/Response Flow

```
sequenceDiagram
    participant Browser
    participant Vercel
    participant Railway
    participant MongoDB

    Browser->>Vercel: Request page
    Vercel-->>Browser: HTML + JS bundle

    Browser->>Browser: Execute React app
    Browser->>Railway: GET /api/children
    Railway->>MongoDB: Query database
    MongoDB-->>Railway: Data
    Railway-->>Browser: JSON response

    Browser->>Browser: Render UI

    Note over Browser: User interaction
    Browser->>Railway: POST /api/transactions
    Railway->>MongoDB: Update document
    MongoDB-->>Railway: Confirmation
    Railway-->>Browser: Success response
    Browser->>Browser: Update UI
```

State Management Flow

```
graph TB
    subgraph FrontendState["Frontend State"]
        ParentState[Parent Dashboard State]
        ChildState[Child View State]
        LoginState[Login State]
    end

    subgraph BackendState["Backend State"]
        ServerState[Server State]
        DBState[Database State]
    end

    subgraph Sync["Synchronization"]
        AutoRefresh[Auto-refresh 5s]
        ManualRefresh[Manual Refresh]
    end

    ParentState -->|API Call| ServerState
    ChildState -->|API Call| ServerState
    ServerState -->|Query| DBState
    DBState -->|Response| ServerState
    ServerState -->|Update| ParentState
    ServerState -->|Update| ChildState
    AutoRefresh --> ChildState
    ManualRefresh --> ParentState

    style ParentState fill:#667eea
    style ChildState fill:#ec4899
    style DBState fill:#f59e0b
```

Error Handling Flow

```
graph TB
    Start[User Action] --> Try{Try Operation}
    Try -->|Success| Success[Update UI]
    Try -->|Error| Catch[Error Handler]

    Catch --> NetworkError{Network Error?}
    NetworkError -->|Yes| NetworkMsg[Show Connection Error]
    NetworkError -->|No| ServerError{Server Error?}

    ServerError -->|Yes| ServerMsg[Show Server Error]
    ServerError -->|No| ValidationError{Validation Error?}

    ValidationError -->|Yes| ValidationMsg[Show Validation Error]
    ValidationError -->|No| GenericMsg[Show Generic Error]

    NetworkMsg --> Log[Log to Console]
    ServerMsg --> Log
    ValidationMsg --> Log
    GenericMsg --> Log

    Log --> UserAlert[Alert User]
    Success --> End[Complete]
    UserAlert --> End

    style Catch fill:#ef4444
    style Success fill:#22c55e
```

Documentation Version: 1.0
Last Updated: December 2024