

ApnaResume - Data Flow Diagrams

Table of Contents

- 1. [System Overview](#)
- 2. [User Authentication Flow](#)
- 3. [Resume Creation Flow](#)
- 4. [Resume Editing Flow](#)
- 5. [AI Content Generation Flow](#)
- 6. [Resume View & Export Flow](#)
- 7. [Resume Deletion Flow](#)
- 8. [Data Layer Architecture](#)
- 9. [Component Data Flow](#)
- 10. [State Management Flow](#)

1. System Overview

High-Level System Data Flow

```
graph TB
    subgraph "Client Layer"
        UI[React UI Components]
        Context[Form Context Provider]
    end

    subgraph "Application Layer"
        Middleware[Clerk Middleware]
        ServerActions[Server Actions]
        API[Next.js App Router]
    end

    subgraph "External Services"
        Clerk[Clerk Auth Service]
        Gemini[Google Gemini AI]
    end

    subgraph "Data Layer"
        Mongoose[Mongoose ODM]
        MongoDB[(MongoDB Database)]
    end

    UI --> Context
    Context --> API
    API --> Middleware
    Middleware --> Clerk
    API --> ServerActions
    ServerActions --> Mongoose
    Mongoose --> MongoDB
    ServerActions --> Gemini

    MongoDB --> Mongoose
    Mongoose --> ServerActions
    ServerActions --> API
```

API --> Context
Context --> UI

Request-Response Cycle

```
sequenceDiagram
    participant User
    participant Browser
    participant NextJS as Next.js Server
    participant Actions as Server Actions
    participant DB as MongoDB
    participant AI as Gemini AI

    User->>Browser: Interact with UI
    Browser->>NextJS: HTTP Request
    NextJS->>NextJS: Middleware Auth Check
    NextJS->>Actions: Call Server Action

    alt Database Operation
        Actions->>DB: Query/Mutation
        DB-->>Actions: Result
    else AI Operation
        Actions->>AI: Generate Content
        AI-->>Actions: AI Response
    end

    Actions-->>NextJS: Action Result
    NextJS-->>Browser: Response
    Browser-->>User: Updated UI
```

2. User Authentication Flow

Sign-Up Flow

```
flowchart TD
    A[User visits /sign-up] --> B{Already authenticated?}
    B -->|Yes| C[Redirect to /dashboard]
    B -->|No| D[Show Clerk Sign-Up Form]
    D --> E[User enters credentials]
    E --> F[Clerk validates credentials]
    F --> G{Valid?}
    G -->|No| H[Show error message]
    H --> E
    G -->|Yes| I[Create Clerk User]
    I --> J[Set session cookie]
    J --> K[Redirect to /dashboard]
```

Sign-In Flow

```
flowchart TD
    A[User visits /sign-in] --> B{Already authenticated?}
    B -->|Yes| C[Redirect to /dashboard]
```

```

B -->|No| D[Show Clerk Sign-In Form]
D --> E[User enters credentials]
E --> F[Clerk authenticates]
F --> G{Success?}
G -->|No| H[Show error message]
H --> E
G -->|Yes| I[Set session cookie]
I --> J[Redirect to /dashboard]

```

Route Protection Flow

```

flowchart TD
    A[Request to protected route] --> B[Middleware intercepts]
    B --> C{Route matches protected pattern?}
    C -->|"/dashboard"| D[Check Auth]
    C -->|"/my-resume/:id/edit"| D
    C -->|Other routes| E[Allow access]
    D --> F{User authenticated?}
    F -->|Yes| G[Allow access to route]
    F -->|No| H[Redirect to /sign-in]

```

3. Resume Creation Flow

Create New Resume

```

flowchart TD
    A[Dashboard Page] --> B[User clicks Add Resume +]
    B --> C[Open AddResume Dialog]
    C --> D[User enters resume title]
    D --> E[Form validation - Zod]
    E --> F{Valid title?}
    F -->|No| G[Show validation error]
    G --> D
    F -->|Yes| H[Generate UUID]
    H --> I[Call createResume action]
    I --> J[connectToDB]
    J --> K[Resume.create in MongoDB]
    K --> L{Success?}
    L -->|No| M[Show error toast]
    L -->|Yes| N[Return resume data]
    N --> O[Redirect to /my-resume/id/edit]

```

Server Action: createResume

```

sequenceDiagram
    participant Client
    participant createResume as createResume()
    participant Mongoose
    participant MongoDB

    Client->>createResume: {resumeId, userId, title}
    createResume->>Mongoose: connectToDB()

```

```
Mongoose->>MongoDB: Connect
MongoDB-->>Mongoose: Connection
createResume->>MongoDB: Resume.create()
MongoDB-->>createResume: New Resume Document
createResume-->>Client: {success: true, data: resume}
```

4. Resume Editing Flow

Overall Edit Flow

```
flowchart TD
    A[/my-resume/id/edit] --> B[FormProvider loads]
    B --> C[fetchResume action]
    C --> D[Populate experience, education, skills]
    D --> E[Set formData state]
    E --> F[Render ResumeEditForm]
    F --> G{Current step?}
    G -->|1| H[PersonalDetailsForm]
    G -->|2| I[SummaryForm]
    G -->|3| J[ExperienceForm]
    G -->|4| K[EducationForm]
    G -->|5| L[SkillsForm]
    G -->|6| M[SectionOrderBoard]

    H --> N[User edits fields]
    I --> N
    J --> N
    K --> N
    L --> N
    M --> N

    N --> O[handleInputChange updates context]
    O --> P[ResumePreview re-renders]
```

Form Step Navigation

```
flowchart LR
    subgraph "Edit Flow Steps"
        S1[1. Personal Details]
        S2[2. Summary]
        S3[3. Experience]
        S4[4. Education]
        S5[5. Skills]
        S6[6. Section Order]
        S7[Finish]
    end

    S1 -->|Next| S2
    S2 -->|Next| S3
    S3 -->|Next| S4
    S4 -->|Next| S5
    S5 -->|Next| S6
    S6 -->|Finish| S7
```

```
S2 -->|Prev| S1
S3 -->|Prev| S2
S4 -->|Prev| S3
S5 -->|Prev| S4
S6 -->|Prev| S5
```

Finish & Save Flow

```
flowchart TD
    A[User clicks Finish] --> B[Set loading state]
    B --> C[Prepare updates object]
    C --> D[updateResume action]
    D --> E[addExperienceToResume action]
    E --> F[addEducationToResume action]
    F --> G[addSkillToResume action]
    G --> H{All successful?}
    H -->|Yes| I[Redirect to /my-resume/id/view]
    H -->|No| J[Show error toast]
    J --> K[Clear loading state]
```

Experience Save Flow

```
sequenceDiagram
    participant Client
    participant Action as addExperienceToResume
    participant Resume as Resume Model
    participant Exp as Experience Model
    participant DB as MongoDB

    Client->>Action: (resumeId, experienceDataArray)
    Action->>Resume: findOne({resumeId})
    Resume->>DB: Query
    DB-->>Resume: Resume Document

    loop Each experience entry
        alt Has _id (existing)
            Action->>Exp: findById(_id)
            alt Found
                Action->>Exp: findByIdAndUpdate(_id, data)
            else Not found
                Action->>Exp: new Experience(data).save()
            end
        else No _id (new)
            Action->>Exp: new Experience(data).save()
        end
    end

    Action->>Resume: Update experience array
    Action->>Resume: resume.save()
    Action-->>Client: {success: true, data: resume}
```

5. AI Content Generation Flow

Summary Generation

```
flowchart TD
    A[User in Summary Form] --> B[Enter Job Title]
    B --> C[Click Generate with AI]
    C --> D[generateSummary action]
    D --> E{Job title provided?}
    E -->|Yes| F[Create job-specific prompt]
    E -->|No| G[Create personality-based prompt]
    F --> H[Send to Gemini AI]
    G --> H
    H --> I[Parse JSON response]
    I --> J[Return 3 summary options]
    J --> K[Display options to user]
    K --> L[User selects option]
    L --> M[Update summary field]
```

AI Request Processing

```
sequenceDiagram
    participant UI as Summary Form
    participant Action as generateSummary()
    participant Gemini as Gemini AI API

    UI->>Action: jobTitle parameter
    Action->>Action: Construct prompt
    Action->>Gemini: chatSession.sendMessage(prompt)
    Note over Action,Gemini: Model: gemini-2.5-flash<br/>Response: JSON format
    Gemini-->>Action: JSON with 3 summaries
    Action->>Action: JSON.parse(response)
    Action-->>UI: Array of {experience_level, summary}
```

Experience Description Generation

```
flowchart TD
    A[Experience Form] --> B[Enter title + company]
    B --> C[Click Generate Description]
    C --> D[generateExperienceDescription]
    D --> E[Create experience prompt]
    E --> F[Gemini AI processes]
    F --> G[Return 3 activity levels]
    G --> H{High Activity}
    G --> I{Medium Activity}
    G --> J{Low Activity}
    H --> K[Display Options]
    I --> K
    J --> K
    K --> L[User selects]
    L --> M[Update workSummary with HTML]
```

6. Resume View & Export Flow

Resume View Flow

```
flowchart TD
    A[/my-resume/id/view] --> B[FormProvider initializes]
    B --> C[fetchResume action]
    C --> D[Load with populated data]
    D --> E{Is owner view?}
    E -->|Yes| F[Show full header + actions]
    E -->|No| G[Show preview header]
    F --> H[Render ResumePreview]
    G --> H
    H --> I[Get current template]
    I --> J{Template type?}
    J -->|classic| K[ClassicTemplate]
    J -->|modern| L[ModernTemplate]
    J -->|minimal| M[MinimalTemplate]
```

PDF Export Flow

```
flowchart TD
    A[User clicks Download PDF] --> B[Set isGenerating = true]
    B --> C[Get resumeRef element]
    C --> D[Dynamic import html2pdf.js]
    D --> E[Configure PDF options]
    E --> F[html2canvas renders at 2x]
    F --> G[jsPDF creates A4 PDF]
    G --> H[Trigger file download]
    H --> I[Set isGenerating = false]
```

PDF Generation Details

```
sequenceDiagram
    participant Button as Download Button
    participant Handler as handleDownload()
    participant HTML2PDF as html2pdf.js
    participant Canvas as html2canvas
    participant PDF as jsPDF

    Button->>Handler: onClick
    Handler->>Handler: Set loading state
    Handler->>HTML2PDF: Dynamic import
    Handler->>HTML2PDF: Configure options
    Note over Handler,HTML2PDF: Scale: 2x<br/>Format: A4<br/>Quality: 0.98
    HTML2PDF->>Canvas: Render DOM to canvas
    Canvas-->>HTML2PDF: Canvas image
    HTML2PDF->>PDF: Create PDF from canvas
    PDF-->>Handler: PDF blob
    Handler->>Handler: Trigger download
    Handler->>Handler: Clear loading state
```

Share URL Flow

```
flowchart TD
    A[User clicks Share URL] --> B[RWebShare component]
    B --> C[Prepare share data]
    C --> D{Browser supports Web Share?}
    D -->|Yes| E[Open native share dialog]
    D -->|No| F[Copy to clipboard fallback]
    E --> G[User selects share target]
    G --> H[Share completes]
```

7. Resume Deletion Flow

```
flowchart TD
    A[Dashboard - Resume Card] --> B[Click menu ...]
    B --> C[Select Delete]
    C --> D[Open Alert Dialog]
    D --> E{User confirms?}
    E -->|Cancel| F[Close dialog]
    E -->|Delete| G[Set loading state]
    G --> H[deleteResume action]
    H --> I[Resume.findOneAndDelete]
    I --> J[revalidatePath]
    J --> K{Success?}
    K -->|Yes| L[Show success toast]
    K -->|No| M[Show error toast]
    L --> N[Refresh resume list]
    M --> O[Close dialog]
    N --> O
    O --> O
```

Delete Server Action

```
sequenceDiagram
    participant Card as ResumeCard
    participant Action as deleteResume()
    participant DB as MongoDB
    participant Cache as Next.js Cache

    Card->>Action: (resumeId, path)
    Action->>Action: connectToDB()
    Action->>DB: Resume.findOneAndDelete({resumeId})
    DB-->>Action: Deleted document
    Action->>Cache: revalidatePath(path)
    Action-->>Card: {success: true}
    Card->>Card: refreshResumes()
```

8. Data Layer Architecture

Database Connection Flow

```

flowchart TD
    A[Server Action called] --> B[connectToDB]
    B --> C{NEXT_MONGODB_URL exists?}
    C -->|No| D[Log error, return]
    C -->|Yes| E{Already connected?}
    E -->|Yes| F[Return existing connection]
    E -->|No| G[mongoose.connect]
    G --> H[Set isConnected = true]
    H --> I[Log success]
    I --> F

```

Data Relationship Flow

```

graph LR
    subgraph Resume Document
        R[Resume]
    end

    subgraph Related Documents
        E1[Experience 1]
        E2[Experience 2]
        ED1[Education 1]
        S1[Skill Category 1]
        S2[Skill Category 2]
    end

    R -->|experience[]| E1
    R -->|experience[]| E2
    R -->|education[]| ED1
    R -->|skills[]| S1
    R -->|skills[]| S2

```

Populate Flow

```

sequenceDiagram
    participant Action as fetchResume()
    participant Resume as Resume Model
    participant Exp as Experience Model
    participant Edu as Education Model
    participant Skill as Skill Model
    participant DB as MongoDB

    Action->>Resume: findOne({resumeId})
    Resume->>DB: Query resume
    DB-->>Resume: Resume with ObjectId refs
    Resume->>Exp: populate("experience")
    Resume->>Edu: populate("education")
    Resume->>Skill: populate("skills")
    Exp->>DB: Query experiences
    Edu->>DB: Query education
    Skill->>DB: Query skills
    DB-->>Action: Fully populated resume

```

9. Component Data Flow

Form Context Data Flow

```
graph TD
    subgraph FormProvider
        FP[FormProvider Component]
        FD[formData State]
        HI[handleInputChange]
    end

    subgraph FormComponents
        PD[PersonalDetailsForm]
        SF[SummaryForm]
        EF[ExperienceForm]
        ED[EducationForm]
        SK[SkillsForm]
        SO[SectionOrderBoard]
    end

    subgraph PreviewComponents
        RP[ResumePreview]
        TP[Template Component]
    end

    FP --> FD
    FP --> HI

    PD -- "reads" --> FD
    SF -- "reads" --> FD
    EF -- "reads" --> FD
    ED -- "reads" --> FD
    SK -- "reads" --> FD
    SO -- "reads" --> FD

    PD -- "calls" --> HI
    SF -- "calls" --> HI
    EF -- "calls" --> HI
    ED -- "calls" --> HI
    SK -- "calls" --> HI
    SO -- "calls" --> HI

    HI -- "updates" --> FD
    FD -- "triggers re-render" --> RP
    RP --> TP
```

Template Rendering Flow

```
flowchart TD
    A[ResumePreview] --> B[useFormContext]
    B --> C[Get formData.template]
    C --> D{Template type?}
    D -->|classic| E[ClassicTemplate]
    D -->|modern| F[ModernTemplate]
```

```

D -->|minimal| G[MinimalTemplate]

E --> H[Map sectionOrder]
F --> H
G --> H

H --> I{Section key?}
I -->|summary| J[Render Summary]
I -->|experience| K[Render Experiences]
I -->|education| L[Render Education]
I -->|skills| M[Render Skills]

```

10. State Management Flow

Form State Update Flow

```

flowchart TD
    A[User types in input] --> B[onChange event]
    B --> C[handleInputChange]
    C --> D{Value type?}
    D -->|Array| E[Direct array assignment]
    D -->|Object| F[Merge with existing object]
    D -->|Primitive| G[Direct value assignment]
    E --> H[setFormData]
    F --> H
    G --> H
    H --> I[React re-render]
    I --> J[Preview updates]

```

Loading State Flow

```

stateDiagram-v2
    [*] --> Idle
    Idle --> Loading: User action
    Loading --> Success: Operation complete
    Loading --> Error: Operation failed
    Success --> Idle: Reset
    Error --> Idle: Dismiss

    state Loading {
        [*] --> ShowSpinner
        ShowSpinner --> DisableButtons
    }

    state Success {
        [*] --> ShowToast
        ShowToast --> Redirect
    }

    state Error {
        [*] --> ShowErrorToast
    }

```

Section Order State Flow

```
flowchart TD
    A[SectionOrderBoard loads] --> B[Get sectionOrder from formData]
    B --> C[Initialize DnD state]
    C --> D[User drags section]
    D --> E[onDragEnd callback]
    E --> F[Reorder array]
    F --> G[handleInputChange with new order]
    G --> H[Update formData.sectionOrder]
    H --> I[Preview re-renders with new order]
```

Summary

This document illustrates the complete data flow throughout the ApnaResume application:

1. **Authentication Flow:** Clerk-managed auth with middleware protection
2. **CRUD Operations:** Server actions with MongoDB via Mongoose
3. **AI Integration:** Gemini API for content generation
4. **State Management:** React Context for form data
5. **Export:** Client-side PDF generation with html2pdf.js

Each flow is designed to provide:

- Clear separation of concerns
- Optimistic UI updates where applicable
- Proper error handling and user feedback
- Efficient data loading with population