Building an Invoice Management Tool with AI SDK and QuickBooks

Overview

This document outlines how to build an invoice management application that integrates the Vercel AI SDK with QuickBooks Online. The application will feature a dual-panel interface displaying invoice data alongside an AI chat interface, allowing users to interact with their invoices through natural language.

Architecture

The application will consist of:

1. A QuickBooks data connector
2. AI tools built with Vercel AI SDK
3. A React-based UI with dual panels

Vercel AI SDK Tool Implementation

Creating Tools with AI SDK

The Vercel AI SDK provides a structured way to create tools that can be called by AI models. Each tool has these key components:

1. **Description**: Helps the AI model understand when to use the tool
2. **Parameters**: Defines the expected inputs using Zod schema
3. **Execute function**: Implements the actual functionality when the tool is called

Invoice Tool Implementation

JavaScript
```javascript
// Example tool definition pattern (not to be implemented
directly)
import { z } from 'zod';
import { generateText, tool } from 'ai';
```

```
// Define your invoice tools using the tool helper function
const invoiceTools = {
  getInvoice: tool({
    description: 'Get details of a specific invoice by ID',
    parameters: z.object({
      invoiceId: z.string().describe('The ID of the invoice to
retrieve'),
    }),
    execute: async ({ invoiceId }) => {
      // Implementation will connect to QuickBooks API
      // Return invoice data
    }),
  }),

  // Other invoice tools would follow the same pattern
};
```

Multi-Step Interactions

For complex invoice operations, set up multi-step interactions:

JavaScript
```javascript
// Example pattern for multi-step interactions (not to be
implemented directly)
const { text, steps } = await generateText({
  model: yourModel,
  tools: invoiceTools,
  maxSteps: 5, // Allow multiple steps for complex invoice
workflows
  prompt: userMessage,
});
```

Tool Choice Configuration

Control when invoice tools are selected:

```javascript
// Example tool choice configuration (not to be implemented directly)
const result = await generateText({
  model: yourModel,
  tools: invoiceTools,
  toolChoice: 'auto', // Let the model decide which invoice tool to use
  // Alternative options: 'required', 'none', or specify a tool
  prompt: userMessage,
});
```

QuickBooks Integration

Invoice CRUD Operations

Connect the tool execution functions to QuickBooks APIs:

1. **Read Invoice**:
   - Retrieve invoice details using `qbo.getInvoice(id, callback)`
   - List invoices with `qbo.findInvoices(criteria, callback)`
2. **Create Invoice**:
   - Create new invoices via `qbo.createInvoice(object, callback)`
3. **Update Invoice**:
   - Update existing invoices with `qbo.updateInvoice(object, callback)`
   - Mark invoices as void by including `void: true` property
4. **Delete Invoice**:
   - Delete invoices using `qbo.deleteInvoice(idOrEntity, callback)`
5. **Email Invoice PDF**:
   - Send invoice PDFs via `qbo.sendInvoicePdf(id, sendTo, callback)`

UI Integration with AI SDK

Streaming AI Responses

Implement streaming for a responsive chat experience:

```javascript
// Example streaming pattern (not to be implemented directly)
const stream = streamText({
  model: yourModel,
  tools: invoiceTools,
  maxSteps: 3,
  messages: [{ role: 'user', content: userMessage }],
});

// Handle the stream to update UI
for await (const chunk of stream) {
  if (chunk.type === 'text') {
    // Update chat UI with text
  } else if (chunk.type === 'tool-call') {
    // Update invoice panel based on tool call
  } else if (chunk.type === 'tool-result') {
    // Update UI with result of invoice operation
  }
}
```

Managing Tool State

Keep track of all tool interactions:

```javascript
// Example tool state tracking (not to be implemented directly)
const { steps } = await generateText({
  model: yourModel,
  tools: invoiceTools,
  maxSteps: 5,
  prompt: userMessage,
  onStepFinish({ text, toolCalls, toolResults }) {
    // Update UI based on each step's completion
    // e.g., show loading state during invoice retrieval
    // then update invoice panel with results
  },
});
```

Error Handling

Handle specific tool-related errors:

```javascript
// Example error handling (not to be implemented directly)
try {
  const result = await generateText({
    model: yourModel,
    tools: invoiceTools,
    prompt: userMessage,
  });
} catch (error) {
  if (error.name === 'NoSuchToolError') {
    // Handle invalid tool request
  } else if (error.name === 'InvalidToolArgumentsError') {
    // Handle invalid invoice parameters
  } else if (error.name === 'ToolExecutionError') {
    // Handle QuickBooks API errors
  }
}
```

Implementation Roadmap

1. Set up QuickBooks OAuth flow
2. Define invoice tools using the AI SDK tool helper
3. Implement the execute functions with QuickBooks API calls
4. Create the dual-panel UI
5. Integrate streaming AI responses
6. Add error handling for tool execution

7. Implement multi-step workflows for complex invoice operations

This document serves as a high-level guide for developers to implement an AI-powered invoice management tool using the Vercel AI SDK tool functionality and QuickBooks integration.