**Backend Microservice (Node.js):**

**Set up Node.js environment:** Install Node.js and npm (or yarn).

**Choose a framework:** Consider Express for efficient API development.

**Implement API endpoints:**

**User authentication:**

Use the passport-google-oauth20 library for OAuth integration.

Store user information securely (e.g., hashed passwords, access tokens).

**Usage details:**

Fetch usage data from a database or external service.

Define relevant metrics to be exposed in the API response.

**Billing information:**

Calculate billing details based on usage and pricing plans.

Store billing history for reference.

**Invoice generation:**

Use a library like pdfkit or jsPDF to create invoices.

Store generated invoices for download or sending.

**Integrate with Zapier:**

Use Zapier's webhook trigger for usage updates.

Create Zapier workflows to initiate billing actions based on usage metrics.

**Frontend (React):**

**Set up React project:** Use create-react-app to initialize.

**Implement UI components:**

**Login component:** Integrate Google OAuth using react-google-login.

**Usage details component:** Display usage metrics fetched from the backend.

**Billing information component:** Show billing details and invoice options.

**Invoice generation component:** Trigger invoice generation on the backend.

**Handle authentication:** Store access tokens and manage user sessions.

**Fetch data from backend:** Use fetch or axios to retrieve usage and billing information.

**Display data:** Render fetched data in appropriate UI elements.

**Integration with Zapier.com:**

**Set up Zapier triggers:** Create a Zap triggered by usage updates from the backend.

**Configure Zapier actions:** Perform billing actions (e.g., create invoices, send notifications).

**Reflect updates in frontend:** Use WebSockets or polling to update billing information in real-time.