Enhanced Code Analyzer with Intelligent Chunking

A powerful web application that analyzes code files of any size using Claude AI with automatic intelligent chunking for large files. This enhanced version extends the original code analyzer to handle large codebases by breaking them into manageable chunks while maintaining context and relationships.

🆕 New Features

API Key Validation

- **Real-time Validation**: Automatically validates API keys when entered
- **Status Indicators**: Visual feedback showing key validity and account status
- **Error Details**: Specific error messages for invalid keys, rate limits, or insufficient credits
- **Secure Storage**: Keys are validated before being saved locally

Email Integration

- **Automatic Email Reports**: Send analysis results directly to your email
- **Professional Formatting**: Clean, readable email format with summary statistics
- **Configurable Preferences**: Choose whether to receive email notifications
- **Real-time Status**: Shows email sending progress and confirmation

Intelligent Chunking System

- **Automatic Size Detection**: Files exceeding 8,000 characters are automatically chunked
- **Language-Aware Breaking**: Chunks are split at logical boundaries (functions, classes, statements)
- **Context Preservation**: Each chunk includes contextual information from previous chunks
- **Cross-Chunk Analysis**: Detects and reports dependencies between chunks

Enhanced Analysis Capabilities

- **Multi-Language Support**: Optimized chunking for JavaScript, Python, Java, C++, and 25+ languages
- **Complexity Analysis**: Determines chunking strategy based on code complexity
- **Relationship Mapping**: Identifies structural relationships across chunks
- **Merged Results**: Combines analysis from multiple chunks into coherent reports

📊 Chunking Algorithm

How It Works

1. **Size Check**: Files over 8,000 characters trigger chunking
2. **Complexity Analysis**: Calculates code complexity score based on:
   - Line count
   - Function/class density
   - Nesting levels
   - Language-specific patterns
3. **Intelligent Breaking**: Finds optimal split points at:
   - Function boundaries
   - Class definitions
   - Statement endings
   - Natural code breaks
4. **Context Addition**: Each chunk includes:
   - Previous chunk context (200 characters)
   - Line number mapping
   - Structural relationship info
5. **Analysis & Merging**:
   - Analyzes each chunk independently
   - Identifies cross-chunk dependencies
   - Merges results into unified report

Chunking Example

```
Original File (15,000 chars)
├─ Chunk 1 (8,000 chars) - Functions 1-5
├─ Chunk 2 (7,000 chars) - Functions 6-10 + context from Chunk 1
└─ Analysis merged into single report
```

🚀 Quick Start

Prerequisites

- Node.js 16+ and npm
- Anthropic API key (Get one here)

Installation

```bash
# Clone the repository
git clone <repository-url>
cd enhanced-code-analyzer

# Install dependencies
npm install

# Configure your API key
cp .env.example .env
# Add your key to .env:
# VITE_ANTHROPIC_API_KEY=sk-ant-api03-xxxxx...

# Configure EmailJS (optional)
# See EmailJS Setup Guide for detailed instructions

# Start development server
npm run dev
```

EmailJS Setup (Optional)

To enable email functionality:

1. Sign up at EmailJS.com
2. Create an email service and template
3. Update the EmailJS configuration in the code:
   - Replace `YOUR_EMAILJS_USER_ID` with your User ID
   - Replace `YOUR_EMAIL_SERVICE_ID` with your Service ID
   - Replace `YOUR_EMAIL_TEMPLATE_ID` with your Template ID

See the EmailJS Setup Guide artifact for detailed instructions.

📁 Usage

1. Configure Settings

- **API Key**: Enter and validate your Anthropic API key
- **Email (Optional)**: Enter your email to receive analysis results
- **Preferences**: Choose whether to email results automatically

2. Upload Files

- **Drag & Drop**: Drop files/folders onto the upload area
- **File Browser**: Click "Select Files" to browse
- **Supported Formats**: JavaScript, Python, Java, C++, and 25+ languages

3. Choose Analysis Mode

- **Individual Files**: Detailed per-file analysis with chunking
- **System Analysis**: Architecture overview across multiple files
- **Error Focus**: Bug detection and security vulnerabilities
- **Business Report**: Non-technical summary for stakeholders

4. Start Analysis

- Click "Start Analysis" (button shows validation status)
- Watch progress indicators for analysis and email sending
- Results appear on screen and in your email (if configured)

5. Review Results

For Chunked Files:

- **Overall Summary**: Aggregated statistics across all chunks
- **Issue Breakdown**: Issues categorized by severity and type
- **Cross-Chunk Notes**: Dependencies and relationships between chunks
- **Recommendations**: Prioritized action items

Chunking Indicators:

- Files show "Will be chunked" during upload
- Results display "X chunks analyzed" badge
- Cross-chunk dependencies highlighted

🔧 Configuration

Chunking Parameters

```
javascript
```

```
const CHUNK_CONFIG = {
  MAX_CHUNK_SIZE: 8000,     // Characters per chunk
  OVERLAP_SIZE: 200,        // Context overlap between chunks
  MAX_FILE_SIZE: 500000,    // Maximum processable file size
  COMPLEXITY_THRESHOLD: 1000 // Complexity score triggering chunking
};
```

Language-Specific Settings

Each language has optimized chunking rules:

javascript

```
javascript: {
  breakPoints: ['}', ';', '\n'],
  functionPattern: /function\s+\w+|=>\s*{|class\s+\w+/g,
  complexity: { functionWeight: 10, classWeight: 15 }
}
```

📊 Analysis Output

Chunked File Results

json

```
{
  "fileName": "large-component.js",
  "totalChunks": 3,
  "overallSummary": {
    "totalIssues": 12,
    "highSeverityIssues": 2,
    "securityIssues": 1,
    "issuesByCategory": {
      "performance": 5,
      "security": 1,
      "maintainability": 6
    }
  },
  "crossChunkNotes": [
    "Function getUser() called across chunks 1-2",
    "State management patterns span multiple chunks"
  ],
  "recommendations": [
    {
      "priority": "high",
      "action": "Address security vulnerability in chunk 2",
      "details": "Potential XSS in user input handling"
    }
  ]
}
```


🎯 Best Practices

File Organization

- **Large Files**: Consider refactoring files >500KB
- **Modular Code**: Well-structured code chunks more effectively
- **Clear Boundaries**: Functions and classes create natural chunk breaks

Analysis Optimization

- **Multiple Files**: Upload related files together for better context
- **Incremental Analysis**: Start with error focus, then expand
- **Review Dependencies**: Pay attention to cross-chunk notes

🔍 Troubleshooting

API Key Issues

Invalid API Key Format:

- Ensure key starts with `sk-ant-`
- Check for extra spaces or characters
- Get a new key from console.anthropic.com

API Key Validation Failed:

- **401 Error**: Invalid or expired API key
- **429 Error**: Rate limit exceeded or insufficient credits
- **Network Error**: Check internet connection

API Key Status Indicators:

- ✓ Green: Valid and active
- ✗ Red: Invalid or has issues
- Spinning: Currently validating

Email Issues

Email Not Sending:

- Verify EmailJS configuration is complete
- Check Service ID, Template ID, and User ID are correct
- Ensure email service is connected in EmailJS dashboard

Invalid Email Address:

- Use a valid email format
- Check for typos in email address
- Verify email preferences are saved

EmailJS Limits:

- Free plan: 200 emails/month
- Rate limit: 1 email per second
- Template size limit: 50KB

File Analysis Issues

Files Not Chunking:

- Check file size (must exceed 8,000 characters)
- Verify file format is supported
- Review complexity threshold settings

Analysis Errors:

- Ensure API key is validated (green checkmark)
- Check internet connection
- Verify file is text-based, not binary

Missing Dependencies:

- Cross-chunk relationships require full file context
- Consider uploading complete modules together
- Review architectural analysis mode for better insights

Performance Tips

For Large Codebases:

- Upload files in logical groups (by feature/module)
- Use system analysis mode for architectural overview
- Process critical files first with error focus mode

📈 Metrics & Limits

Metric	Value	Notes
Max Chunk Size	8,000 chars	Configurable
Max File Size	500KB	Hard limit
Overlap Size	200 chars	Context preservation
Supported Languages	25+	Full list in config
Analysis Modes	4	Individual, System, Error, Business

🛠️ Development

Project Structure

```
enhanced-code-analyzer/
├── src/
│   ├── App.jsx              # Main application
│   ├── chunking-utils.js    # Chunking algorithms
│   ├── config.js            # Configuration
│   └── index.css            # Styles
├── public/                  # Static assets
└── package.json             # Dependencies
```

Building

```bash
npm run build    # Production build
npm run preview  # Preview build
npm run lint     # Code linting
```

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/chunking-enhancement`)
3. Commit changes (`git commit -am 'Add intelligent chunking'`)
4. Push to branch (`git push origin feature/chunking-enhancement`)
5. Create Pull Request

📄 License

MIT License - see LICENSE file for details

🆘 Support

- **Documentation**: Check this README and inline code comments
- **Issues**: Open GitHub issues for bugs and feature requests
- **API Support**: Visit Anthropic Support

Enhanced Code Analyzer - Making large codebase analysis accessible and intelligent.