Deep Research Agent - API Configuration Guide

🔑 Required API Keys and Setup

This guide will help you obtain and configure the necessary API keys for the Deep Research Agent to function properly.

📋 Essential APIs

1. OpenAI API (Required for AI Processing)

Purpose: Natural language processing, patent analysis, and report generation

Setup Steps:
1. Visit OpenAI API Platform (https://platform.openai.com/)
2. Create an account or sign in
3. Navigate to “API Keys” section
4. Click “Create new secret key”
5. Copy the key and add to `.env.local`:
`OPENAI_API_KEY=sk-your-openai-api-key-here`

Cost: Pay-per-use, approximately $0.002 per 1K tokens for GPT-4
Rate Limits: 3 RPM, 40,000 TPM for free tier

2. Google Patents API (Free - Recommended)

Purpose: Primary patent search and retrieval

Setup Steps:
1. No API key required for basic searches
2. For advanced features, visit Google Cloud Console (https://console.cloud.google.com/)
3. Enable the Patents API
4. Create credentials if needed
5. Configuration (basic usage):
`GOOGLE_PATENTS_API_URL=https://patents.googleapis.com/v1/patents`

Cost: Free for basic usage
Rate Limits: 100 requests per 100 seconds

3. USPTO API (Free)

Purpose: US patent data and detailed information

Setup Steps:
1. Visit USPTO Developer Portal (https://developer.uspto.gov/)
2. Register for an account
3. Create an application
4. Get your API key
5. Add to `.env.local`:
`USPTO_API_KEY=your-uspto-api-key-here`

Cost: Free
Rate Limits: 1000 requests per hour

🔧 Optional APIs (For Enhanced Features)

4. European Patent Office (EPO) API

Purpose: European patent data access

Setup Steps:
1. Visit EPO Open Patent Services (https://www.epo.org/searching-for-patents/data/web-services.html)
2. Register for access
3. Obtain API credentials
4. Add to `.env.local`:
`EPO_API_KEY=your-epo-api-key-here`

5. WIPO Global Brand Database API

Purpose: International patent and trademark data

Setup Steps:
1. Visit WIPO API Portal (https://www.wipo.int/branddb/en/)
2. Request API access
3. Configure credentials
4. Add to `.env.local`:
`WIPO_API_KEY=your-wipo-api-key-here`

📧 Email Service Configuration

Option 1: SendGrid (Recommended)

Purpose: Email notifications for completed research

Setup Steps:
1. Visit SendGrid (https://sendgrid.com/)
2. Create free account (100 emails/day free)
3. Create API key
4. Add to `.env.local`:
`EMAIL_SERVICE=sendgrid`
`SENDGRID_API_KEY=your-sendgrid-api-key-here`

Option 2: AWS SES

Purpose: Scalable email service

Setup Steps:
1. Set up AWS account
2. Configure SES service
3. Get access credentials
4. Add to `.env.local`:
`EMAIL_SERVICE=ses`
`AWS_SES_ACCESS_KEY_ID=your-aws-access-key`
`AWS_SES_SECRET_ACCESS_KEY=your-aws-secret-key`

💾 Database Setup

PostgreSQL Installation

Required for data storage

Setup Steps:
1. **Install PostgreSQL**:
```bash
# Ubuntu/Debian
sudo apt update
sudo apt install postgresql postgresql-contrib

# macOS (with Homebrew)
brew install postgresql

# Windows - Download from postgresql.org
```

1. **Create Database and User**:
   ```sql
   – Connect to PostgreSQL as superuser
   sudo -u postgres psql

– Create database
CREATE DATABASE deep_research_agent;

– Create user
CREATE USER research_user WITH PASSWORD ‘research_password’;

– Grant privileges
GRANT ALL PRIVILEGES ON DATABASE deep_research_agent TO research_user;
```

1. **Update .env.local**:
   `DATABASE_URL=postgresql://research_user:research_password@localhost:5432/deep_research_agent`

Redis Installation

Required for job queues and caching

Setup Steps:
1. **Install Redis**:
```bash
# Ubuntu/Debian
sudo apt install redis-server

# macOS (with Homebrew)
brew install redis

# Windows - Download from redis.io
```

1. **Start Redis Service**:
   ```bash
   ```

```
    # Ubuntu/Debian
    sudo systemctl start redis-server

# macOS
brew services start redis
```

1. **Verify Installation**:
   `bash`
   `redis-cli ping`
   `# Should return: PONG`

🔐 Security Configuration

JWT Secrets

Generate secure JWT secrets for authentication:

```
# Generate a secure JWT secret
node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"
```

Add to `.env.local`:

```
JWT_SECRET=your-generated-secure-secret-here
```

Environment Security

1. **Never commit `.env.local` to version control**
2. **Use different secrets for production**
3. **Rotate API keys regularly**
4. **Monitor API usage for suspicious activity**

🚀 Quick Development Setup

Minimal Configuration for Development

Create `.env.local` with essential settings:

```
# Copy the example file
cp .env.example .env.local

# Edit the file with your API keys
nano .env.local
```

Minimum required for development:

```
NODE_ENV=development
PORT=3000
DATABASE_URL=postgresql://research_user:research_password@localhost:5432/deep_research
_agent
REDIS_URL=redis://localhost:6379
JWT_SECRET=your-jwt-secret-here
OPENAI_API_KEY=sk-your-openai-api-key-here
```

📊 API Usage Monitoring

Rate Limit Management

The application includes automatic rate limiting to respect API quotas:

- **Google Patents**: 100 requests per 100 seconds
- **USPTO**: 1000 requests per hour
- **OpenAI**: Based on your plan (3 RPM for free tier)

Cost Monitoring

Monitor your API usage to avoid unexpected charges:

1. **OpenAI Dashboard**: Track token usage and costs
2. **Google Cloud Console**: Monitor API calls if using advanced features
3. **AWS Console**: Track SES usage if using AWS email service

🔧 Troubleshooting

Common Issues

1. **Database Connection Failed**:
   - Verify PostgreSQL is running
   - Check credentials in `.env.local`
   - Ensure database exists

2. **Redis Connection Failed**:
   - Verify Redis is running: `redis-cli ping`
   - Check Redis URL in `.env.local`

3. **API Authentication Failed**:
   - Verify API keys are correct
   - Check rate limits haven't been exceeded
   - Ensure API services are active

4. **OpenAI API Errors**:
   - Check API key validity
   - Verify sufficient credits/quota
   - Monitor rate limits

Testing API Configuration

Use the built-in API testing endpoints:

```
# Test patent search
curl http://localhost:3000/api/test/patents

# Test OpenAI connection
curl http://localhost:3000/api/test/ai

# Test database connection
curl http://localhost:3000/api/test/database
```

📈 Production Considerations

Scaling API Usage

For production deployment:

1. **Upgrade API Plans**: Move from free tiers to paid plans
2. **Implement Caching**: Reduce redundant API calls
3. **Load Balancing**: Distribute requests across multiple instances
4. **Monitoring**: Set up alerts for rate limits and errors

Security Hardening

1. **Use environment-specific secrets**
2. **Implement API key rotation**
3. **Set up monitoring and alerting**
4. **Use secure secret management services**

This configuration guide ensures your Deep Research Agent has all necessary API access while maintaining security and cost efficiency.