Crawlio Application Documentation

Tech Stack

Frontend

- **Framework**: Next.js 14.2.32
- **Library**: React
- **Styling**: Tailwind CSS
- **Development**: Concurrently for running multiple processes

Backend

- **Runtime**: Node.js 18.17.0+
- **Framework**: Express.js 4.18.2
- **Web Scraping**: Playwright 1.39.0 (with Chromium browser)
- **HTML Parsing**: Cheerio 1.0.0-rc.12
- **Authentication**: JSON Web Tokens (jsonwebtoken 9.0.2), bcryptjs 2.4.3
- **Database**: PostgreSQL (pg 8.11.3)
- **Cache**: Redis 4.6.10
- **Job Queue**: Bull 4.12.0
- **Logging**: Winston 3.11.0
- **Security**: Helmet 7.1.0, CORS 2.8.5, Compression 1.7.4
- **Rate Limiting**: express-rate-limit 7.1.5
- **Validation**: Joi 17.11.0
- **File Upload**: Multer 1.4.5-lts.1
- **Payments**: Stripe 14.7.0
- **Cloud Storage**: AWS SDK 2.1490.0
- **Utilities**: UUID 9.0.1

Python Client

- **Runtime**: Python 3.6+
- **HTTP Client**: requests library
- **Data Handling**: JSON, typing

Infrastructure

- **Containerization**: Docker & Docker Compose
- **Database Container**: PostgreSQL 15 Alpine
- **Cache Container**: Redis 7 Alpine

Features

Core Functionality

1. **Web Crawling & Scraping**
    - JavaScript rendering support using Playwright

- Headless browser automation
- Dynamic content extraction

2. **User Management**
    - User registration with email/password
    - JWT-based authentication
    - Secure password hashing with bcrypt
    - User profile management
3. **API Key Management**
    - Automatic API key generation upon registration
    - Secure API key authentication for crawling requests
    - Key-based access control
4. **Crawl History**
    - Track all crawling jobs
    - Job status monitoring
    - Historical data retrieval

Extraction Capabilities

5. **Content Extraction**
    - Text content extraction
    - Link discovery and extraction
    - Image URL collection
    - Meta tag parsing
    - Structured data extraction
6. **Media & Assets**
    - Screenshot capture capability
    - Image extraction from web pages
    - Asset URL collection

Advanced Features

7. **Professional UI/UX**
    - Clean, emoji-free interface
    - Responsive design with Tailwind CSS
    - Authentication-protected dashboard
    - Real-time crawl status updates
8. **API Integration**
    - RESTful API endpoints
    - JSON response format
    - Comprehensive error handling
    - Rate limiting and security
9. **Python Client Library**
    - Easy-to-use Python wrapper
    - Automatic user registration
    - Crawl result processing
    - Professional console output

10. **Infrastructure & DevOps**
    - Docker containerization
    - Health checks for all services
    - Persistent data storage
    - Production-ready configuration
    - Logging and monitoring
11. **Security & Performance**
    - Helmet.js security headers
    - CORS configuration
    - Request compression
    - Rate limiting
    - Input validation with Joi
12. **Cloud Integration**
    - AWS S3 storage support
    - Stripe payment processing
    - Scalable architecture

Crawl URL Parameters

The following parameters can be configured when submitting a crawl request:

Boolean Parameters

- **extractText** (boolean): Extract text content from the webpage
  - Default: true
  - Description: Retrieves all readable text content
- **extractLinks** (boolean): Extract all hyperlinks from the page
  - Default: true
  - Description: Collects all anchor tag URLs
- **extractImages** (boolean): Extract image URLs from the page
  - Default: false
  - Description: Collects all image source URLs
- **extractMeta** (boolean): Extract meta tags and page metadata
  - Default: false
  - Description: Retrieves meta tags, title, description, etc.
- **screenshot** (boolean): Capture a screenshot of the webpage
  - Default: false
  - Description: Takes a full-page screenshot

Usage Examples

JavaScript (Frontend)

```
const crawlOptions = {
  extractText: true,
  extractLinks: true,
  extractImages: false,
  extractMeta: true,
  screenshot: false
};
```

Python Client

```
crawl_options = {
    "extractText": True,
    "extractLinks": True,
    "extractMeta": True,
    "screenshot": False
}

result = client.crawl_url("https://example.com", options=crawl_options)
```

Response Structure

The API returns crawl results in the following format:

```
{
  "success": true,
  "job_id": "uuid-string",
  "data": {
    "text": "Extracted text content...",
    "links": ["https://link1.com", "https://link2.com"],
    "images": ["https://image1.jpg", "https://image2.png"],
    "meta": {
      "title": "Page Title",
      "description": "Page Description",
      "keywords": "keyword1, keyword2"
    }
  },
  "metadata": {
    "finalUrl": "https://example.com",
    "loadTime": 2500,
    "contentLength": 125000
  }
}
```

Getting Started

Prerequisites

- Node.js 18.17.0+
- Docker & Docker Compose
- Python 3.6+ (for client)

Installation

1. Clone the repository
2. Install dependencies: `npm install`
3. Set up environment variables
4. Start services: `docker-compose up -d`
5. Run frontend: `cd frontend && npm run dev`
6. Use Python client: `python crawlio_client.py`

API Endpoints

- `POST /api/auth/register` - User registration
- `POST /api/auth/login` - User login
- `POST /api/crawl/url` - Submit crawl job
- `GET /api/crawl/history` - Get crawl history
- `GET /health` - Health check

Generated on: September 3, 2025 Version: 1.0.0