POC Implementation Plan: AI-Powered API Generator

Architecture Overview

Your POC needs three core components:

1. **Frontend Interface** - Collects user requirements
2. **AI Processing Layer** - Interprets requirements and orchestrates data collection
3. **API Generation & Serving Layer** - Creates and hosts the endpoint

---

Detailed Implementation Plan

Phase 1: Frontend Interface (User Input Collection)

Technology Stack:

- **React** (Free, Open Source)
- **Next.js** (Free, Open Source)
- **HTML/CSS/JavaScript** (Free, Native Web Technologies)
- **React Hook Form** (Free, Open Source) - Form handling library

Required Input Fields:

1. **Data Description** (text area)
    - What data do you need? (e.g., "Current IPOs listed on Indian stock market")
2. **Data Source** (optional text input)
    - Where is this data available? (e.g., "Chittorgarh.com, Investorgain.com")
3. **Desired Fields** (multi-line input)
    - What fields should the response contain? (e.g., "company_name, listing_date, issue_price, grey_market_premium")
4. **Response Structure** (JSON example or description)
    - Preferred JSON structure
5. **Update Frequency** (dropdown)
    - Real-time, Hourly, Daily, Weekly

UI Flow:

```
[User fills form] → [Submit] → [Loading state] → [Display generated API endpoint + docs]
```

Phase 2: AI Processing Layer

Core Workflow:

Step 1: Requirement Analysis

- Use Claude/GPT API to parse user input
- Extract: data source URLs, required fields, data structure

Prompt Example:

```
Analyze this API requirement and extract:
1. Potential data sources (URLs)
2. Fields to extract
3. Data structure
4. Scraping strategy

Requirement: {user_input}
```

Step 2: Data Acquisition Strategy

The AI should generate:

- Web scraping script (using Puppeteer/Playwright/BeautifulSoup)
- Data extraction logic
- Data transformation rules

Step 3: Code Generation

Generate three components:

1. **Scraper module** - Fetches raw data
2. **Transformer module** - Cleans and structures data
3. **API route handler** - Serves the data

Phase 3: API Generation & Serving

Technology Stack:

- **Node.js** (Free, Open Source)
- **Express** (Free, Open Source) - Web framework
- **Python** (Free, Open Source)
- **FastAPI** (Free, Open Source) - Python web framework
- **PostgreSQL** (Free, Open Source) - Relational database
- **MongoDB** (Free, Open Source) - NoSQL database
- **Node-cron** (Free, Open Source) - Job scheduler for Node.js
- **Celery** (Free, Open Source) - Python job scheduler

Implementation Approach:

Option A: Dynamic Code Execution (Simpler for POC)

javascript

```javascript
// Pseudo-code
app.post('/generate-api', async (req, res) => {
  const userRequirements = req.body;

  // 1. Send to AI for analysis
  const strategy = await analyzeWithAI(userRequirements);

  // 2. Generate unique endpoint ID
  const endpointId = generateUniqueId();

  // 3. Create scraper function
  const scraperCode = await generateScraperCode(strategy);

  // 4. Store configuration in database
  await saveEndpointConfig(endpointId, {
    scraperCode,
    updateFrequency: userRequirements.updateFrequency,
    responseStructure: strategy.structure
  });

  // 5. Schedule first data fetch
  await executeScraperAndCache(endpointId);

  // 6. Return generated API endpoint
  res.json({
    endpoint: `/api/data/${endpointId}`,
    documentation: generateDocs(strategy)
  });
});

// Serve the generated endpoint
app.get('/api/data/:endpointId', async (req, res) => {
  const data = await getCachedData(req.params.endpointId);
  res.json(data);
});
```

Option B: Template-Based Generation (More Robust)

- Pre-built scraper templates for common patterns
- AI selects and customizes appropriate template
- Less flexible but more reliable

Recommended Tech Stack for POC

Frontend

- **Next.js** (Free, Open Source) - Built-in API routes, easy deployment
- **Tailwind CSS** (Free, Open Source) - Quick styling
- **Vercel** (Free tier available) - Deployment platform for Next.js

Backend

- **Node.js + Express** (Both Free, Open Source) - JavaScript throughout
- **Python + FastAPI** (Both Free, Open Source) - Better for web scraping libraries

AI Integration

- **Anthropic Claude API** (Paid - $15/1M input tokens, $75/1M output tokens for Sonnet 4.5)
    - Free tier: $5 credit for new accounts
- **OpenAI GPT-4** (Paid - $2.50/1M input tokens, $10/1M output tokens)
    - Free tier: $5 credit for new accounts
- **Alternative:** Use for requirement analysis, code generation

Web Scraping

- **Puppeteer** (Free, Open Source) - Node.js, handles dynamic content
- **Cheerio** (Free, Open Source) - Node.js, fast HTML parsing
- **BeautifulSoup** (Free, Open Source) - Python HTML parsing
- **Selenium** (Free, Open Source) - Python browser automation
- **Playwright** (Free, Open Source) - Modern browser automation

Database

- **MongoDB** (Free, Open Source; MongoDB Atlas free tier: 512MB)
- **PostgreSQL** (Free, Open Source)
- **Redis** (Free, Open Source; Redis Cloud free tier: 30MB)

Job Scheduling

- **node-cron** (Free, Open Source) - Schedule data updates
- **Bull** (Free, Open Source) - Redis-based job queue

Hosting Options

- **Render** (Free tier available - 750 hours/month)
- **Railway** (Free tier: $5 credit/month)
- **Heroku** (Free tier discontinued, paid plans start at $7/month)
- **DigitalOcean** (Paid - starts at $4/month)
- **AWS/GCP/Azure** (Free tiers available with limitations)

POC Implementation Steps

Week 1: Core Infrastructure

1. Set up Next.js project with API routes
2. Create frontend form for collecting requirements
3. Integrate Claude/GPT API for requirement analysis
4. Build basic scraper template

Week 2: API Generation

1. Implement endpoint generation logic
2. Create data caching mechanism
3. Build API serving layer
4. Add basic authentication/rate limiting

Week 3: Polish & Demo

1. Create documentation generator
2. Add error handling
3. Build demo dashboard showing generated APIs
4. Prepare hackathon presentation

Sample Workflow (IPO Example)

User Input:

```
Data: Current IPOs on Indian stock market with grey market prices
Sources: Investorgain.com
Fields: company_name, listing_date, issue_price, gmp, estimated_listing_price
Format: Array of objects
Update: Daily
```

AI Generates:

1. Scraper targeting Investorgain's IPO page
2. Extraction rules for each field
3. Data transformation to match requested structure
4. API endpoint: `/api/data/ipo-grey-market-prices`

User Receives:

json


```
{
  "endpoint": "https://yourpoc.com/api/data/ipo-grey-market-prices",
  "method": "GET",
  "documentation": "...",
  "sampleResponse": {
    "data": [
      {
        "company_name": "Example Corp",
        "listing_date": "2025-01-15",
        "issue_price": 100,
        "gmp": 25,
        "estimated_listing_price": 125
      }
    ]
  }
}
```

Key Challenges & Solutions

Challenge 1: Website Structure Changes

Solution: AI can regenerate scraper when errors detected

Challenge 2: Rate Limiting/Blocking

Solution: Implement proxy rotation, respectful scraping delays

Challenge 3: Code Execution Security

Solution: Use sandboxed environments

- **Docker** (Free, Open Source)
- **VM2** (Free, Open Source) - Node.js sandbox

Challenge 4: Scaling

Solution: For POC, limit to 10-20 concurrent endpoints

Minimum Viable POC

For the hackathon, focus on:

1. ✅Clean UI for requirement input
2. ✅AI-powered analysis of requirements
3. ✅Generation of ONE working endpoint (your IPO example)
4. ✅Simple dashboard showing the generated endpoint
5. ✅Basic documentation page

Time-saving tip: Pre-build the IPO scraper and have AI "generate" it based on user input (Wizard of Oz technique) - proves concept without building full dynamic generation.

Cost Estimation for POC

Free Tier Setup (Recommended for Hackathon):

- **Frontend Hosting:** Vercel (Free)
- **Backend Hosting:** Render (Free tier)
- **Database:** MongoDB Atlas (512MB free)
- **Cache:** Redis Cloud (30MB free)
- **AI API:** Claude/OpenAI ($5 free credit)
- **All Libraries:** Free and Open Source

Estimated Total Cost: $0 - $10 (only if you exceed AI API free credits)

Production Setup (Post-Hackathon):

- **Hosting:** Railway/Render ($10-20/month)
- **Database:** MongoDB Atlas/PostgreSQL ($15-25/month)
- **AI API Usage:** $20-50/month (depending on usage)
- **Total:** $45-95/month

Additional Resources

Learning Resources (All Free):

- **Next.js Documentation:** https://nextjs.org/docs
- **FastAPI Tutorial:** https://fastapi.tiangolo.com/tutorial/
- **Puppeteer Docs:** https://pptr.dev/
- **Web Scraping Guide:** https://scrapingbee.com/blog/web-scraping-101/

API Documentation Tools (Free):

- **Swagger/OpenAPI** (Free, Open Source)
- **Postman** (Free tier available)

Implementation Priority Order

Must Have (Core POC):

1. Frontend form interface
2. AI requirement analysis
3. One hardcoded working scraper (IPO example)
4. Database to store scraped data
5. API endpoint serving the data

Nice to Have (If Time Permits):

1. Dynamic code generation
2. Multiple data source support
3. API authentication
4. Rate limiting
5. Error handling and retry logic
6. Admin dashboard

Future Enhancements (Post-Hackathon):

1. Template library for common data sources
2. Webhook support for real-time updates
3. GraphQL endpoint generation
4. API analytics and monitoring
5. Collaborative API sharing

Success Metrics for Demo

1. **Functionality:** Successfully generate at least one working API endpoint
2. **Speed:** Generate endpoint in under 30 seconds
3. **Accuracy:** Scraped data matches source website
4. **Usability:** Clean, intuitive interface
5. **Documentation:** Clear API docs auto-generated

Final Checklist Before Hackathon

- [ ] All dependencies installed and tested
- [ ] AI API keys configured (Claude/OpenAI)
- [ ] Database connected and tested
- [ ] Sample IPO scraper working
- [ ] Frontend deployed and accessible
- [ ] Backend API deployed
- [ ] Demo script prepared
- [ ] Presentation slides ready
- [ ] Video demo recorded (backup)
- [ ] GitHub repository public and documented

Contact & Support During Development

If you encounter issues:

1. Check package documentation
2. Search Stack Overflow
3. Review GitHub issues for the library
4. Use AI assistants (Claude, ChatGPT) for debugging

Good luck with your hackathon! 🚀