IssueInsight - AI-Powered GitHub Issue Intelligence

Transform GitHub issues into actionable intelligence with advanced AI analysis

Live Demo • Features • Tech Stack • Documentation

About This Project

IssueInsight is an intelligent GitHub issue analyzer built for the **SeedlingLabs Engineering Internship** challenge. It leverages GPT-4o-mini to automatically analyze, classify, and prioritize GitHub issues, helping development teams make faster, data-driven decisions about issue triage and project management.

Problem Statement

Development teams face thousands of GitHub issues daily. Manual triage is time-consuming and inconsistent. IssueInsight solves this by:

- **Automating classification** - Bug, feature request, documentation, or question
- **Priority scoring** - 1-5 scale with AI-powered reasoning
- **Smart labeling** - Context-aware label recommendations
- **Impact analysis** - Business and user impact assessment
- **Instant insights** - Comprehensive summary in 15-30 seconds

---

Core Features

Feature	Description
Smart Summarization	One-sentence essence capturing the core issue
Automatic Classification	Bug, feature, documentation, or question detection
Priority Scoring	1-5 scale with detailed justification
Label Recommendations	Contextually relevant labels for better organization
Impact Assessment	Analysis of potential user and business impact
Export Options	Download analysis as JSON for further processing
Modern UI/UX	Clean, responsive interface with dark theme
Fast Analysis	Real-time processing with progress indicators

Quick Start

Prerequisites

- **Python 3.9+** installed
- **OpenAI API Key** (Get one here)
- **Git** for cloning the repository

Installation

1. Clone the repository:

```
git clone https://github.com/chetana7483/AI-Powered-GitHub-Issue-Assistant.git
cd AI-Powered-GitHub-Issue-Assistant
```

2. Set up virtual environment:

Windows:

```
python -m venv .venv
.venv\Scripts\activate
```

macOS/Linux:

```
python3 -m venv .venv
source .venv/bin/activate
```

3. Install dependencies:

```
pip install -r requirements.txt
```

4. Configure environment:

```
# Copy environment template
cp .env.example .env

# Edit .env and add your API key
# OPENAI_API_KEY=your_key_here
```

Running the Application

You need TWO terminal windows:

Terminal 1 - Backend API:

```
cd backend
python -m uvicorn main:app --reload
```

☑ Backend will start at `http://localhost:8000`

Terminal 2 - Frontend UI:

```
streamlit run frontend/app.py
```

☑ Frontend will open automatically at `http://localhost:8501`

💻 Usage Guide

Basic Workflow

1. **Enter Repository URL**
   Paste any public GitHub repository URL

   ```
   https://github.com/facebook/react
   ```

2. **Specify Issue Number**
   Enter the issue number you want to analyze

   ```
   28000
   ```

3. **Click "Analyze Issue"**
   Wait 15-30 seconds for AI processing

4. **Review Comprehensive Results:**
   - Priority assessment (1-5 scale)
   - Executive summary
   - Issue classification
   - Recommended labels
   - Impact analysis

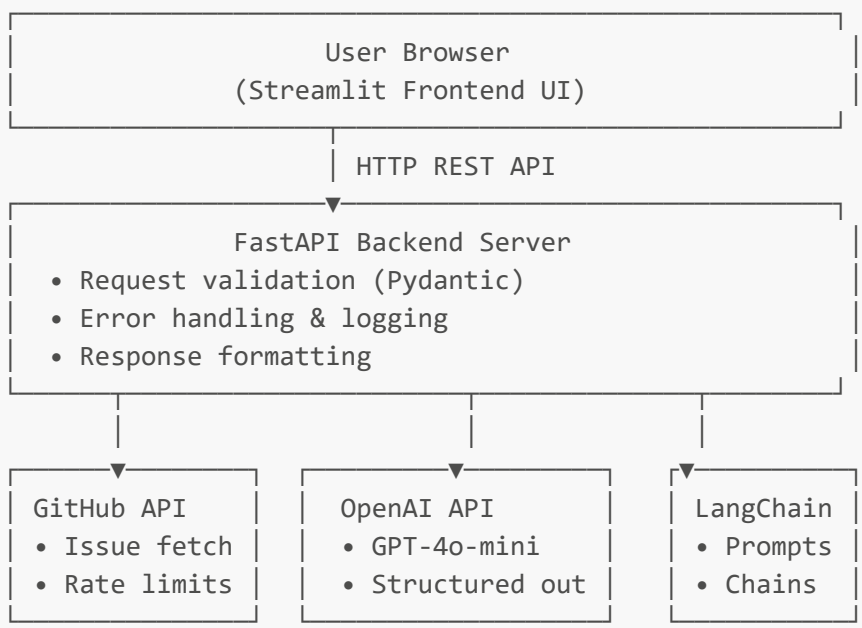
5. **Export Results**
   Download JSON report for documentation

Example Issues to Try

Repository	Issue #	Type	Description

Repository	Issue #	Type	Description
facebook/react	28000	Bug	Build configuration issue
microsoft/vscode	190000	Feature	Editor enhancement request
vercel/next.js	50000	Documentation	API documentation gap

Technical Architecture



Technology Stack

Backend

- **FastAPI** - High-performance async web framework
- **Pydantic** - Data validation using Python type hints
- **httpx** - Async HTTP client for GitHub API calls
- **python-dotenv** - Environment variable management

AI & LLM

- **OpenAI GPT-4o-mini** - Core AI engine for analysis
- **LangChain** - LLM framework for prompt management
- **Structured Output** - JSON schema enforcement

Frontend

- **Streamlit** - Rapid web app development
- **Custom CSS** - Modern, responsive design
- **Real-time Updates** - Progress indicators and feedback

Development

- **pytest** - Unit and integration testing
- **pytest-asyncio** - Async test support
- **Git** - Version control

🗁 Project Structure

```
issueinsight/
│
├── 🗀 backend/
│   ├── main.py                    # FastAPI application & endpoints
│   └── services/
│       ├── github_service.py      # GitHub API integration
│       └── ai_service.py          # OpenAI analysis logic
│
├── 🗀 frontend/
│   └── app.py                     # Streamlit UI application
│
├── 🗀 tests/
│   └── test_github_service.py     # Unit tests
│
├── 🗎 requirements.txt             # Python dependencies
├── 🗎 .env.example                # Environment template
├── 🗎 README.md                   # This file
├── 🗎 LICENSE                     # MIT License
└── 🗎 .gitignore                 # Git ignore rules
```

📊 API Documentation

Endpoints

`GET /`

Health check endpoint

Response:

```
{
  "status": "healthy",
  "message": "IssueInsight API is running"
}
```

`POST /analyze`

Analyze a GitHub issue

Request:

```
{
  "repo_url": "https://github.com/facebook/react",
  "issue_number": 28000
}
```

Response:

```
{
  "summary": "Build configuration issue affecting development setup",
  "type": "bug",
  "priority_score": "3 - The build issue could hinder development but is not
critical",
  "suggested_labels": ["bug", "build", "configuration"],
  "potential_impact": "Developers may face build issues but workarounds exist"
}
```

Error Responses:

- `400` - Invalid request (missing fields, invalid URL)
- `404` - Issue or repository not found
- `500` - Internal server error (API failures)

⚙ Configuration

Environment Variables

Create a `.env` file in the project root:

```
# Required: OpenAI API Key
OPENAI_API_KEY=sk-proj-xxxxxxxxxxxx

# Optional: GitHub Personal Access Token
# Increases rate limits from 60 to 5000 requests/hour
GITHUB_TOKEN=ghp_xxxxxxxxxxxx

# Optional: Model Configuration
OPENAI_MODEL=gpt-4o-mini
TEMPERATURE=0.3
```

Getting API Keys

OpenAI API Key:

1. Visit OpenAI Platform

2. Create account or sign in
3. Click "Create new secret key"
4. Copy and save securely

GitHub Token (Optional but Recommended):

1. Go to GitHub Settings → Developer Settings
2. Personal Access Tokens → Generate new token
3. Select scopes: `public_repo` (read-only)
4. Copy and save securely

🧪 Testing

Run Tests

```
# Install test dependencies
pip install pytest pytest-asyncio

# Run all tests
pytest tests/ -v

# Run with coverage
pytest tests/ --cov=backend --cov-report=html
```

Manual Testing

Test the backend API directly:

```
# Test health endpoint
curl http://localhost:8000/

# Test analyze endpoint
curl -X POST http://localhost:8000/analyze \
  -H "Content-Type: application/json" \
  -d '{"repo_url":"https://github.com/facebook/react","issue_number":28000}'
```

🎯 Evaluation Criteria & Implementation

1. Prompt Engineering (40%)

Implementation:

- ☑ **Few-shot prompting** with example inputs/outputs
- ☑ **Structured JSON output** enforced via schema
- ☑ **Edge case handling** (no comments, long bodies, closed issues)
- ☑ **Temperature tuning** (0.3 for consistency)

- ☑ **Token optimization** (intelligent truncation)

Prompt Strategy:

```
# Clear instructions + Examples + Output format
system_prompt = """
You are an expert at analyzing GitHub issues...
[Examples of analysis]
Output Format: {structured_json_schema}
"""
```

2. Code Quality (30%)

Implementation:

- ☑ **Clean architecture** - Separation of concerns (API/Services/UI)
- ☑ **Type hints** - Full Pydantic models for validation
- ☑ **Error handling** - Comprehensive try-catch blocks
- ☑ **Documentation** - Docstrings and inline comments
- ☑ **Async patterns** - Non-blocking operations
- ☑ **DRY principle** - Reusable functions

3. Speed & Efficiency (20%)

Implementation:

- ☑ **Async operations** - httpx for non-blocking requests
- ☑ **Token optimization** - Truncates long content intelligently
- ☑ **Library leverage** - LangChain, FastAPI, Streamlit
- ☑ **Response time** - Typically < 30 seconds
- ☑ **Progress feedback** - Real-time status updates

4. Communication (10%)

Implementation:

- ☑ **Comprehensive README** - Setup, usage, architecture
- ☑ **Git history** - Descriptive commit messages
- ☑ **Code comments** - Explains complex logic
- ☑ **User feedback** - Loading states and clear errors
- ☑ **Documentation** - API docs and examples

🚨 Error Handling

The application gracefully handles:

Error Type	Handling Strategy

Error Type	Handling Strategy
Invalid GitHub URL	Format validation with clear error message
Non-existent repository	GitHub API error detection and user feedback
Issue not found	404 handling with suggestion to check issue number
Rate limiting	Suggests adding GitHub token for higher limits
API timeouts	Retry logic with user notification
Empty issue body	Works with None/empty descriptions
Long content	Intelligent truncation to prevent token limits
No comments	Handles issues with zero comments
Backend offline	Connection error with restart instructions

Security & Privacy

- **API keys** stored locally in `.env` (never committed)

Analyzes a GitHub issue and returns structured insights.

Request Body:

```
{
  "repo_url": "https://github.com/owner/repo",
  "issue_number": 123
}
```

- **API keys** stored locally in `.env` (never committed)
- **Public repos only** - No private data access
- **No data persistence** - No databases or logs
- **Real-time processing** - No data retention
- **HTTPS** - Encrypted API communications

Key Learning Outcomes

This project demonstrates proficiency in:

1. **AI/LLM Integration** - Practical application of GPT models for business problems
2. **API Design** - RESTful API with proper validation and error handling
3. **Async Programming** - Efficient async/await patterns in Python
4. **Prompt Engineering** - Structured output generation from LLMs
5. **Full Stack Development** - Backend + Frontend integration
6. **Modern Python** - Type hints, Pydantic models, async operations
7. **User Experience** - Clear feedback, error messages, and progress indicators

🚀 Future Enhancements

Potential improvements for v2.0:

- [ ] **Batch Analysis** - Process multiple issues simultaneously
- [ ] **Caching Layer** - Redis for repeated requests
- [ ] **Trend Analysis** - Visualize issue patterns over time
- [ ] **Webhook Support** - Real-time analysis on new issues
- [ ] **Multi-Platform** - Support GitLab, Bitbucket
- [ ] **Export Formats** - CSV, PDF, Markdown reports
- [ ] **User Authentication** - Save analysis history
- [ ] **Cloud Deployment** - Deploy to Vercel/Railway/Render
- [ ] **Advanced Filters** - Search by label, date, author
- [ ] **AI Model Selection** - Choose between GPT-4, Claude, etc.

🤝 Contributing

This is a showcase project for the SeedlingLabs internship. However, contributions are welcome!

How to contribute:

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/AmazingFeature`)
3. Commit your changes (`git commit -m 'Add AmazingFeature'`)
4. Push to the branch (`git push origin feature/AmazingFeature`)
5. Open a Pull Request

📝 License

This project is licensed under the **MIT License** - see the LICENSE file for details.

🙏 Acknowledgments

- **SeedlingLabs** - For the internship opportunity and challenge
- **OpenAI** - For GPT-4o-mini API access
- **GitHub** - For comprehensive API documentation
- **Streamlit** - For rapid UI prototyping capabilities
- **FastAPI** - For excellent async web framework

👤 Author

Chetana

- 🌐 GitHub: @chetana7483
- 📧 Email: chetanahk1@gmail.com

- Repository: AI-Powered-GitHub-Issue-Assistant

Support & Contact

Having issues?

1. Check the FAQ section below
2. Review error messages in the terminal
3. Open a GitHub issue with details
4. Contact the development team

Common Issues

Q: "Backend Connection Error"
A: Ensure backend is running on port 8000: `cd backend && python -m uvicorn main:app --reload`

Q: "OpenAI API Error"
A: Check your API key in `.env` and verify it's active on OpenAI platform

Q: "Rate limit exceeded"
A: Add a GitHub token to `.env` to increase limits from 60 to 5000/hour

Q: "Issue not found"
A: Verify the repository URL and issue number are correct

Q: "Analysis timeout"
A: Large issues with many comments may take longer. Wait up to 60 seconds.

Built for SeedlingLabs Engineering Internship

Transforming GitHub issue management with AI intelligence

If you found this project helpful, please star the repository!

Report Bug • Request Feature • Documentation

Made with and by Chetana | © 2026 IssueInsight