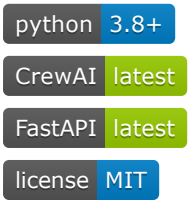
AI Code Review Crew



A multi-agent AI system for automated code review. Five specialized agents collaborate to analyze Python code for bugs, security vulnerabilities, performance issues, and documentation quality.

The Agents

1. **Code Analyst** — Identifies logical errors, edge cases, and exception handling
2. **Security Expert** — Scans for OWASP Top 10 vulnerabilities and security flaws
3. **Performance Optimizer** — Detects algorithmic bottlenecks and inefficient patterns
4. **Documentation Specialist** — Reviews docstrings and code comments
5. **Quality Assurance** — Compiles final report with recommendations

Features

- **Dark Mode UI** — Modern FastAPI interface with drag-and-drop file upload
- **GitHub Integration** — Clone and analyze public repositories directly
- **Real-time Agent Status** — See which agent is currently analyzing your code
- **Detailed Reports** — Markdown reports with severity levels, line numbers, and suggested fixes
- **Multi-Agent Workflow** — Sequential task execution with context sharing

Project Structure

```
3.Crew_AI_projects/
├── app.py                # FastAPI application
├── lambda_handler.py     # lambda_handler
├── src/
│   ├── crew.py           # CrewAI orchestration logic
│   ├── logger.py         # Logging configuration
│   ├── config/
│   │   ├── agents.yaml   # Agent definitions
│   │   ├── tasks.yaml    # Task definitions
│   │   └── settings.py   # Application settings
│   └── static/           # Frontend
│       ├── index.html    # Main UI
│       ├── app.js        # JavaScript
│       └── style.css     # Dark mode styling
├── examples/             # Sample files for testing
├── output/               # Generated reports
├── logs/                 # Application logs
└── requirements.txt      # Python dependencies
```

🚀 Getting Started

Prerequisites

- Python 3.8+
- OpenAI or Anthropic API key

Installation

1. Clone the repository

```
git clone <your-repo-url>
cd 3.Crew_AI_projects
```

2. Install dependencies

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

3. Set up environment variables

```
cp .env.example .env
# Edit .env and add your OPENAI_API_KEY or ANTHROPIC_API_KEY
```

💻 Usage

Running the Application

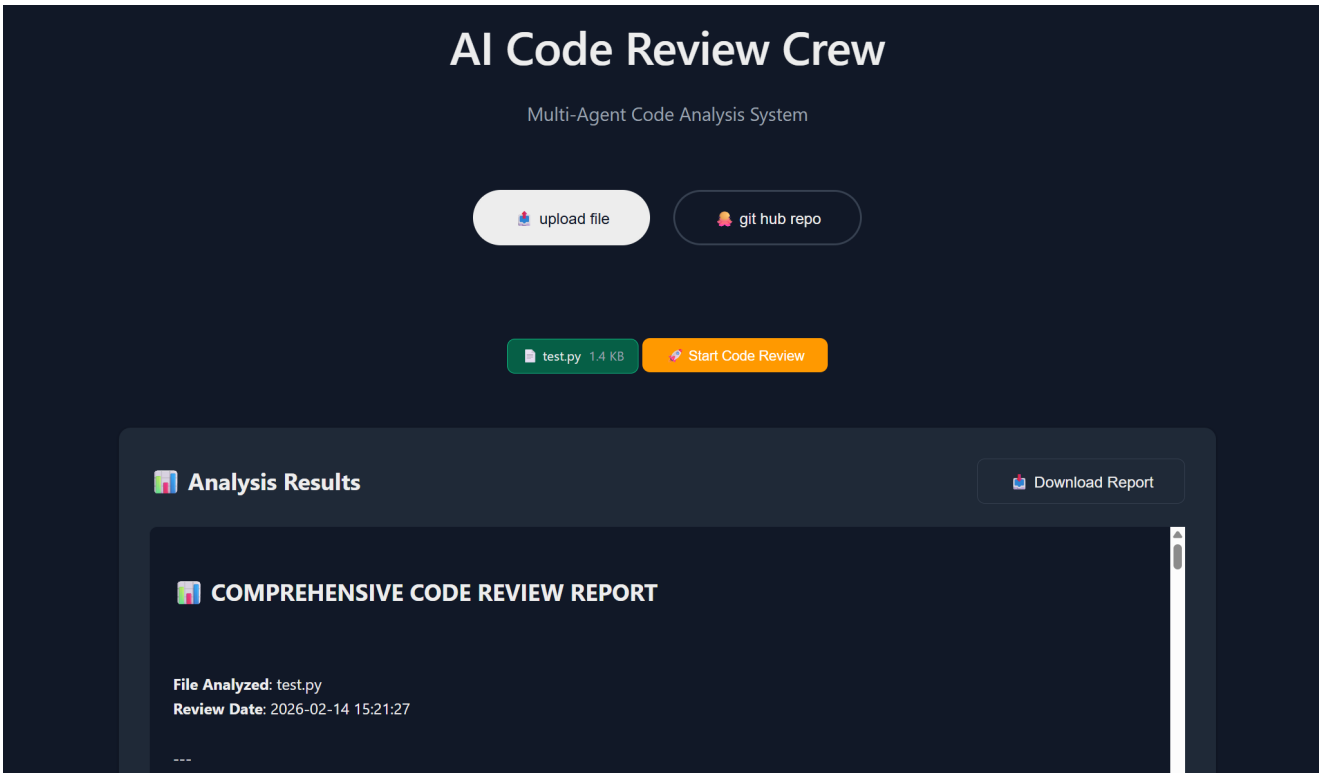
```
python app.py
```

The application will start on `http://localhost:8000`

📡 API Endpoints

- `GET /` - Serve main HTML page
- `GET /health` - Health check endpoint
- `POST /api/review/upload` - Review uploaded Python file
- `POST /api/review/github` - Review GitHub repository
- `GET /api/files/list` - List Python files in a GitHub repository

📸 Screenshots



Code Review Interface showing code analogy

Upload a File

1. Select "Upload File" mode
2. Drop your `.py` file
3. Click "Start Code Review"
4. Download the generated report

Review a GitHub Repo

1. Select "GitHub Repository" mode
2. Paste a public repo URL
3. Select files to analyze
4. Click "Analyze Selected Files"

☁ AWS Lambda Deployment

Deployed as a serverless container on **AWS Lambda + API Gateway** for cost-effective, auto-scaling code review.

Architecture

- **AWS Lambda**: Runs FastAPI app in container (10GB memory, 15min timeout for AI processing)
- **Amazon ECR**: Stores Docker image (Lambda pulls on cold start)
- **API Gateway**: HTTP endpoint routes requests to Lambda function
- **Environment Variables**: API keys (OpenAI/Anthropic) stored in Lambda configuration

Deployment Steps

1. **Build and push Docker image to ECR:**

```
# Create ECR repository
aws ecr create-repository --repository-name crew-ai-lambda --region us-east-1

# Login to ECR
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin <AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com

# Build image
docker build --platform linux/amd64 -t crew-ai-lambda .

# Tag and push
docker tag crew-ai-lambda:latest <AWS_ACCOUNT_ID>.dkr.ecr.us-east-
1.amazonaws.com/crew-ai-lambda:latest
docker push <AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com/crew-ai-
lambda:latest
```

2. **Create Lambda function:**

```
aws lambda create-function \
  --function-name crew-ai-code-review \
  --package-type Image \
  --code ImageUri=<AWS_ACCOUNT_ID>.dkr.ecr.us-east-1.amazonaws.com/crew-ai-
lambda:latest \
  --role arn:aws:iam::<AWS_ACCOUNT_ID>:role/lambda-execution-role \
  --timeout 900 \
  --memory-size 10240 \
  --environment Variables={OPENAI_API_KEY=your_key,LLM_PROVIDER=openai}
```

3. **Create API Gateway HTTP API and integrate with Lambda**

Why Lambda for This Project?

- **Cost-effective**: Pay only when code is analyzed (no idle server costs)
- **Auto-scaling**: Handles concurrent requests automatically
- **Serverless**: No infrastructure management
- **AI workloads**: Large memory (10GB) supports CrewAI agents

```
## 📄 License

MIT License
```