

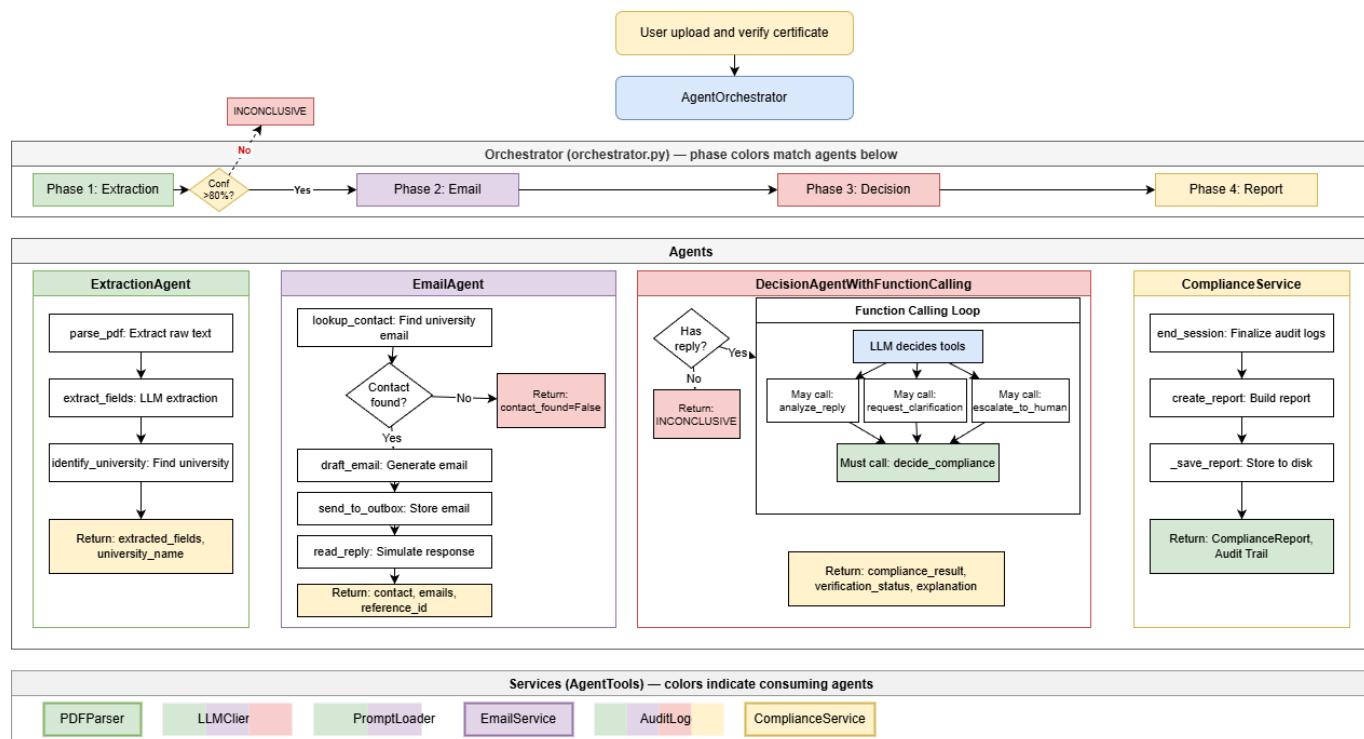
# AgentCheck - AI-Powered Certificate Verification System

[Live Demo](#) | [API Documentation](#)

python 3.11+ FastAPI 0.104+ Docker Ready

An AI agent system that automates the qualification verification workflow for RegTech compliance. The system uses a multi-agent architecture to extract certificate information, communicate with universities, and make compliance decisions with full audit trails.

## Architecture



## Project Structure

```
AgentCheck/
├── api/
│   ├── agents/
│   │   ├── orchestrator.py          # Python backend
│   │   ├── extraction_agent.py     # AI Agents
│   │   ├── email_agent.py          # Main coordinator
│   │   ├── decision_agent.py       #
│   │   └── decision_agent_fc.py    # Function calling variant
│   ├── tools/
│   │   ├── base.py
│   │   ├── definitions.py
│   │   ├── document_tools.py
│   │   ├── communication_tools.py
│   │   ├── analysis_tools.py
│   │   ├── decision_tools.py
│   │   └── tools.py                # Combined tools class
│   ├── models/
│   │   └── schemas.py             # Pydantic models
│   ├── services/
│   │   ├── pdf_parser.py
│   │   ├── email_service.py
│   │   ├── audit_logger.py
│   │   ├── compliance.py
│   │   └── task_queue.py
│   ├── utils/
│   │   ├── llm_client.py
│   │   └── prompt_loader.py
│   ├── constants.py
│   └── main.py                  # FastAPI app + CLI
├── ui/
│   ├── src/
│   │   ├── components/
│   │   ├── services/
│   │   ├── types/
│   │   └── App.tsx
│   └── public/
│       ├── package.json
│       └── vite.config.ts
└── config/
    ├── universities.json        # University contact mappings
    └── prompts/                 # Jinja2 prompt templates
└── data/
    ├── uploads/                 # Uploaded certificates
    ├── outbox/                  # Outgoing emails
    ├── inbox/                   # University replies
    ├── reports/                 # Compliance reports
    ├── queue/                   # Task queue
    └── audit_logs/              # Audit trails
└── docs/
```

```
|   └── architecture_diagram.png  
├── tests/  
└── Dockerfile  
└── docker-compose.yml  
└── nginx.conf  
└── requirements.txt  
└── RESEARCH_INSIGHT.md  
└── README.md
```

## Quick Start

### Option 1: Docker (Recommended)

Uses a single container with Nginx (serves React) + Uvicorn (Python API) for simpler prototype deployment.

```
# Copy environment file and add your Groq API key  
cp .env.example .env  
# Edit .env and set GROQ_API_KEY  
  
# Build and start with Docker Compose  
docker-compose up -d --build  
  
# Access the app at http://localhost:3000  
# API Docs at http://localhost:3000/docs
```

### Option 2: Local Development

```
# Create virtual environment  
python -m venv venv  
  
# Activate (Windows)  
venv\Scripts\activate  
  
# Activate (Linux/Mac)  
source venv/bin/activate  
  
# Install dependencies  
pip install -r requirements.txt  
  
# Copy and configure environment  
cp .env.example .env  
# Edit .env and set GROQ_API_KEY  
  
# Run API server  
uvicorn api.main:app --host 0.0.0.0 --port 8000 --reload
```

## Option 3: React Frontend

```
# Navigate to ui directory
cd ui

# Install Node.js dependencies
pnpm install

# Start development server
pnpm run dev

# Access the UI at http://localhost:3000
# Make sure the API server is running on port 8000
```

## Option 4: CLI

```
# Verify a certificate
python -m api.main verify ./data/sample_pdfs/certificate_verified.pdf

# With specific scenario
python -m api.main verify ./data/sample_pdfs/certificate_denied.pdf --scenario
not_verified

# Output as text
python -m api.main verify ./data/sample_pdfs/certificate_verified.pdf --text

# Save report to file
python -m api.main verify ./data/sample_pdfs/certificate_verified.pdf --output
report.json

# List recent reports
python -m api.main list

# Get specific report
python -m api.main report <report-id>
```

## University Contacts

University contact information is configured in `config/universities.json`. You can also add new universities via the UI (**Universities** tab) or API.

## API Endpoints

Method	Endpoint	Description
GET	/	Health check
GET	/health	Detailed health status

Method	Endpoint	Description
POST	/verify	Verify a certificate
POST	/upload	Upload a PDF file
GET	/reports	List recent reports
GET	/reports/{id}	Get specific report
GET	/reports/{id}/text	Get report as text
GET	/universities	List universities
POST	/universities	Add new university
GET	/docs	Interactive Swagger UI

## Testing

```
# Run all tests
pytest

# Run with coverage
pytest --cov=src --cov-report=html

# Run specific test file
pytest tests/test_agents.py -v
```

## Demo Scenarios

### Simulation Modes

Select different simulated university responses from the sidebar to test various agent behaviors:

Mode	Description	Expected Result
<b>Verified</b>	University confirms the certificate is authentic	COMPLIANT
<b>Not Verified</b>	University denies the certificate exists in records	NOT COMPLIANT
<b>Inconclusive</b>	University response is unclear or incomplete	Requests clarification
<b>Suspicious</b>	Reply contains fraud indicators or inconsistencies	Escalates to human reviewer
<b>Ambiguous</b>	Reply is bureaucratic and non-committal	INCONCLUSIVE
<b>Complex Case</b>	Partial matches + integrity issues + signature discrepancies	Multi-step FC: analyze → escalate

### Sample PDFs

Three sample certificates demonstrate different input quality scenarios:

Sample	Description	Use Case
<b>Generated Sample</b>	AI-generated test certificate with fictional data	Quick testing of any simulation mode
<b>Real Certificate</b>	Authentic University of Western Australia degree	Demonstrates real-world extraction accuracy
<b>Low-Quality/Damaged</b>	Degraded scan with crossed-out/altered text	Tests document quality detection and escalation