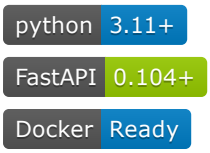
AgentCheck - AI-Powered Certificate Verification System

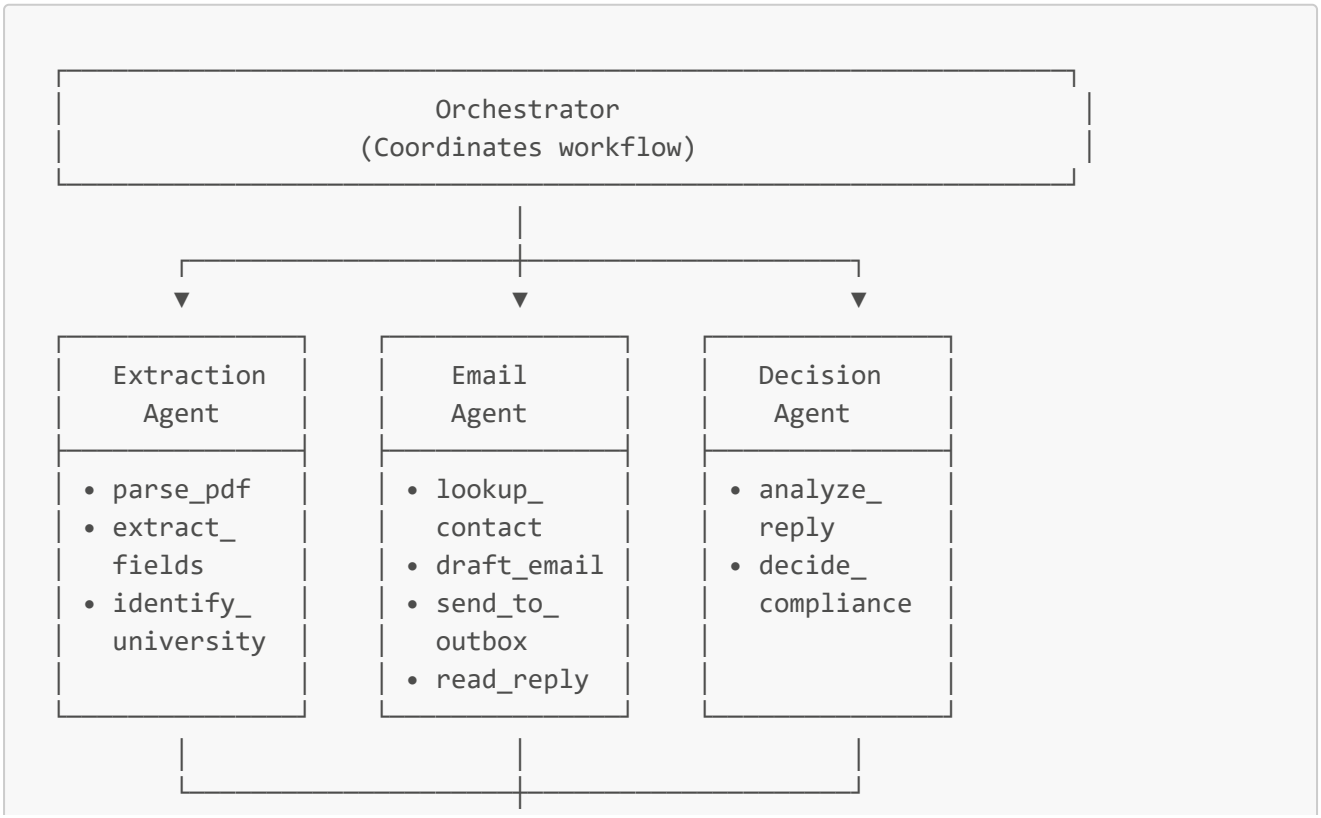


An AI agent system that automates the qualification verification workflow for RegTech compliance. The system uses a multi-agent architecture to extract certificate information, communicate with universities, and make compliance decisions with full audit trails.

Features

- **Multi-Agent Architecture**: Extraction, Email, and Decision agents working together
- **PDF Parsing with Vision API**: Extract text from both digital and scanned certificates using LLM Vision
- **AI-Powered Analysis**: LLM-based field extraction and reply interpretation
- **Simulated Email Workflow**: Complete email drafting and response simulation
- **Compliance Decisions**: Automated verification with clear explanations
- **Full Audit Trail**: Every action logged for compliance requirements
- **Web UI**: Modern React frontend with TypeScript and Tailwind CSS
- **REST API**: FastAPI backend for integration
- **Docker Support**: Production-ready containerization

Architecture



Project Structure

```
AgentCheck/
├── config/
│   ├── universities.json      # University contact mappings
│   └── prompts/               # Jinja2 prompt templates
│       ├── extract_fields.j2
│       ├── draft_email.j2
│       ├── analyze_reply.j2
│       └── identify_university.j2
├── api/                       # Python backend
│   ├── agents/                # AI Agents
│   │   ├── orchestrator.py    # Main coordinator
│   │   ├── extraction_agent.py
│   │   ├── email_agent.py
│   │   └── decision_agent.py
│   ├── tools/
│   │   └── tools.py           # Agent tools (9 tools)
│   ├── 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
│   └── main.py                # FastAPI app + CLI
├── ui/                        # React frontend
│   ├── src/
│   │   ├── components/        # React components
│   │   ├── services/          # API service layer
│   │   ├── types/             # TypeScript types
│   │   ├── App.tsx            # Main app component
│   │   └── main.tsx           # Entry point
│   ├── package.json
│   └── vite.config.ts
├── data/
│   ├── sample_pdfs/           # Sample certificates
│   ├── outbox/                # Outgoing emails
│   ├── inbox/                 # University replies
│   ├── reports/               # Compliance reports
```

```
│   └── audit_logs/            # Audit trails
├── tests/                     # pytest tests
├── Dockerfile
├── docker-compose.yml
├── requirements.txt
├── RESEARCH_INSIGHT.md        # Research & Engineering Document
└── README.md
```

🚀 Quick Start

Option 1: Docker (Recommended)

```
# Clone the repository
git clone <repo-url>
cd AgentCheck

# Copy environment file and add your OpenAI API key
cp .env.example .env
# Edit .env and set OPENAI_API_KEY

# Start with Docker Compose
docker-compose up -d

# Access the services:
# - Frontend: http://localhost:3000
# - API: http://localhost:8000
# - API Docs: http://localhost:8000/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 OPENAI_API_KEY

# Run API server (Terminal 1)
python -m api.main server
```

```
uvicorn api.main:app --host 0.0.0.0 --port 8000 --reload
```

Option 3: React Frontend

```
# Navigate to ui directory (Terminal 2)
cd ui

# Install Node.js dependencies
npm install

# Start development server
npm 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>
```

🔧 Configuration

Environment Variables

Variable	Description	Default
`OPENAI_API_KEY`	OpenAI API key	Required

Variable	Description	Default
`OPENAI_MODEL`	LLM model to use	`gpt-4o-mini`
`API_HOST`	API server host	`0.0.0.0`
`API_PORT`	API server port	`8000`
`FRONTEND_PORT`	React UI port	`3000`
`LOG_LEVEL`	Logging level	`INFO`
`DATA_DIR`	Data directory	`./data`
`CONFIG_DIR`	Config directory	`./config`

University Contacts

Edit `config/universities.json` to add or modify university contacts:

```
{
  "universities": {
    "University Name": {
      "email": "verification@university.edu",
      "country": "Country",
      "verification_department": "Registrar Office"
    }
  }
}
```

API Endpoints

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

Example API Usage

```
# Verify a certificate
curl -X POST "http://localhost:8000/verify" \
  -H "Content-Type: application/json" \
  -d '{"pdf_path": "./data/sample_pdfs/certificate_verified.pdf",
"simulation_scenario": "verified"}'

# Get reports
curl "http://localhost:8000/reports"

# Get specific report
curl "http://localhost:8000/reports/<report-id>"
```

🧪 Testing

```
# Run all tests
pytest

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

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

🐳 Docker Commands

```
# Build images
docker-compose build

# Start services
docker-compose up -d

# View logs
docker-compose logs -f

# Stop services
docker-compose down

# Development mode (with hot reload)
docker-compose --profile dev up dev

# Run tests in container
docker-compose run --rm api pytest
```

📊 Workflow Demo

Scenario 1: Verified Certificate

1. Upload `certificate_verified.pdf`
2. Agent extracts: John Smith, University of Example, BSc Computer Science
3. System finds university contact
4. Drafts verification email
5. Receives "verified" reply
6. AI analyzes: **VERIFIED** (95% confidence)
7. Final decision: **COMPLIANT** ☑

Scenario 2: Denied Certificate

1. Upload `certificate_denied.pdf`
2. Agent extracts: Jane Doe, Global Tech Institute, MBA
3. System finds university contact
4. Drafts verification email
5. Receives "not verified" reply
6. AI analyzes: **NOT_VERIFIED** (90% confidence)
7. Final decision: **NOT COMPLIANT** ❌

Scenario 3: Unknown University

1. Upload `certificate_unknown.pdf`
2. Agent extracts: Alex Johnson, Unknown Academy, Diploma
3. **No university contact found**
4. Final decision: **INCONCLUSIVE** ⚠

Sample Output

```
======================================================================
COMPLIANCE VERIFICATION REPORT
======================================================================

Report ID: 550e8400-e29b-41d4-a716-446655440000
Generated: 2024-12-03T10:30:00

----------------------------------------------------------------------
FINAL DECISION
----------------------------------------------------------------------
Compliance Result: COMPLIANT
Verification Status: VERIFIED

Explanation:
COMPLIANT: The certificate has been verified as authentic by the
issuing university. The university confirmed the certificate is
authentic. Confidence score: 95%

----------------------------------------------------------------------
CERTIFICATE INFORMATION
```

```
--------------------------------------------------------------------------
File: certificate_verified.pdf
Candidate: John Smith
University: University of Example
Degree: Bachelor of Science in Computer Science
Issue Date: 2023-05-15

--------------------------------------------------------------------------
AUDIT TRAIL
--------------------------------------------------------------------------
✓ [2024-12-03T10:30:00] 001_session_start: Started new verification session
✓ [2024-12-03T10:30:01] 002_parse_pdf: Parsing PDF file
✓ [2024-12-03T10:30:02] 003_extract_fields: Extracting structured fields
✓ [2024-12-03T10:30:03] 004_identify_university: Identifying university
✓ [2024-12-03T10:30:04] 005_lookup_contact: Looking up contact
✓ [2024-12-03T10:30:05] 006_draft_email: Generating verification email
✓ [2024-12-03T10:30:06] 007_send_to_outbox: Email stored in outbox
✓ [2024-12-03T10:30:07] 008_read_reply: Reading university reply
✓ [2024-12-03T10:30:08] 009_analyze_reply: Analyzing reply with LLM
✓ [2024-12-03T10:30:09] 010_decide_compliance: Making compliance decision

==========================================================================
END OF REPORT
==========================================================================
```

📖 Documentation

- Research & Engineering Insight Document - Detailed analysis of design decisions
- API Documentation - Interactive Swagger docs (when running)

⚠️ Limitations

- University replies are **simulated** (no real email integration)
- Limited university database (5 sample universities)
- LLM required for full functionality (mock mode available)

🔮 Future Enhancements

- [ ] Real email integration (SMTP/IMAP)
- [ ] API integration with university verification services
- [ ] Human-in-the-loop review workflow
- [ ] Batch processing support
- [ ] Advanced analytics dashboard

📄 License

MIT License - See LICENSE file for details.

Built for RegTech compliance automation by AgentCheck.