Project Drishti - AI Event Safety System

Real-time crowd monitoring and safety management system for large events.

Project Structure

```
.
├── README.md
├── backend
│   ├── agent-backend
│   │   ├── Dockerfile
│   │   ├── conversation_api.py
│   │   ├── conversation_script.py
│   │   ├── db
│   │   │   ├
│   │   │   ├── llm_logger.py
│   │   │   └── trigger_manager.py
│   │   ├── deploy.txt
│   │   ├── gemini_helpers.py
│   │   ├── requirements.txt
│   │   └── tools.py
│   └── simulator-backend
│       ├── Dockerfile
│       ├── deploy.txt
│       ├── requirements.txt
│       └── server.py
├── frontend
│   ├── agent
│   │   ├── README.md
│   │   └── frontend
│   │       ├── chat-app.js
│   │       ├── chat-styles.css
│   │       ├── chat.html
│   │       ├── config.js
│   │       └── speech-services.js
│   ├── deploy.sh
│   ├── index.html
│   └── simulator
│       ├── README.md
│       └── frontend
│           ├── app.js
│           ├── config.js
│           ├── firebase.json
│           ├── index.html
│           └── styles.css
└── requirements.txt
```

Quick Start

1. Simulator Service

```
cd simulator/backend
pip install -r requirements.txt
python server.py
# Open simulator/frontend/index.html in browser
```

2. Agent Service

```
cd agent/backend
pip install -r requirements.txt
export GEMINI_API_KEY="your-api-key"
python conversation_api.py
# Open agent/frontend/chat.html in browser
```

Key Features

- **Real-time Monitoring** - Live venue state with 2-second updates
- **Event Simulation** - Trigger emergencies (Fire, Medical, Security, etc.)
- **AI Response** - Proactive agent with tool-based actions
- **Gradual Evacuation** - Realistic 10-second evacuation simulation
- **Auto-resolution** - Alerts resolve based on actions taken

Services Communication

- Simulator exposes REST API on port 3001
- Agent connects to simulator API for state/actions
- Frontend dashboards poll their respective backends