**Contents**

[1) Deployment and Setup Guide 1](#_Toc210476509)

[1.1 Running the System Locally 1](#_Toc210476510)

[1.2 Python AI Service 1](#_Toc210476511)

[1.3 .NET Backend API 2](#_Toc210476512)

[1.4 React Frontend 3](#_Toc210476513)

[1.5 Deploying the System to Azure 3](#_Toc210476514)

[1.6 AI Service Deployment 3](#_Toc210476515)

[1.7 .NET Backend Deployment 4](#_Toc210476516)

[1.8 React Frontend Deployment 4](#_Toc210476517)

[1.9 Communication Flow on Azure 4](#_Toc210476518)

# Deployment and Setup Guide

This document explains how the application operates in a local development environment and how it can be deployed to Azure. The system consists of three main components that communicate with each other through standard HTTP requests:

* **Frontend (React application)** – user interface for uploading images and displaying the processed results.
* **Backend (.NET API)** – receives the uploaded images, performs validation, and communicates with the AI service.
* **AI Service (Python Flask)** – performs the actual image analysis and returns a structured response.

## Running the System Locally

The following describes how the three parts of the solution interact when running on a local computer. Each service runs on its own port, and they communicate through their local addresses.

## Python AI Service

The AI component is a Flask application that performs basic image analysis such as brightness, contrast, and orientation detection.  
It listens on [**http://localhost:5002**](http://localhost:5002) and exposes two endpoints:

* /process – accepts image uploads and returns the analysis results.
* /health – simple health check to confirm the service is running.

To run it locally, use the following steps:

* Open a terminal in the folder where app.py is located.
* Install the required packages with:
* pip install -r requirements.txt
* Start the service:
* python app.py
* Once running, verify the endpoint with a test request:
* curl -X POST http://localhost:5002/process -F "image=@sample.png"

If successful, the service returns a JSON object containing the classification and analysis data.

## .NET Backend API

The backend application is a .NET Web API that receives image files from the React frontend. It validates the uploaded file, forwards it to the Python AI service for processing, and then returns the AI response to the frontend.

The API runs locally at [**https://localhost:5001**](https://localhost:5001).  
Its configuration specifies the AI service address in appsettings.json:

"AIService": {

"Url": "http://localhost:5002/process"

}

To start the backend API:

* Open a new terminal in the .NET project directory.
* Run the following commands:
* dotnet build
* dotnet run
* The API can now be accessed at:
* https://localhost:5001/api/process

The backend and the AI service communicate through the local network. When an image is uploaded to the .NET endpoint, it forwards the image to http://localhost:5002/process and waits for the response before sending it back to the frontend.

## React Frontend

The frontend is a React application that allows the user to upload an image and view the processing results.

It runs locally at [**http://localhost:3000**](http://localhost:3000).

To start the frontend:

* Open a terminal in the React project directory.
* Install the required dependencies:
* npm install
* Start the development server:
* npm start
* The application will open automatically in the browser.  
  When the user uploads an image, the frontend sends a POST request to:
* https://localhost:5001/api/process

The backend then forwards the image to the AI service at http://localhost:5002/process.  
After processing, the response is returned to the frontend, which displays the classification and confidence values.

In the local environment, the communication flow is as follows:

**React (http://localhost:3000)**

**↓**

**.NET API (https://localhost:5001)**

**↓**

**Python AI (http://localhost:5002)**

## Deploying the System to Azure

Once the application is running successfully on a local machine, each component can be deployed to Azure while maintaining the same interaction pattern. The local addresses are simply replaced with the public URLs assigned by Azure.

## AI Service Deployment

The Python Flask service can be deployed to **Azure App Service** or as a **containerized web app**.  
After deployment, it will receive a public address such as:

https://your-ai-service.azurewebsites.net/process

In the .NET configuration, update the AI service URL:

"AIService": {

"Url": "https://your-ai-service.azurewebsites.net/process"

}

The rest of the code remains unchanged.

## .NET Backend Deployment

The backend can be published directly to an **Azure App Service** using the dotnet publish command or Visual Studio’s publish feature.  
Once deployed, the API will have a public address such as:

https://your-backend.azurewebsites.net/api/process

Make sure the configuration in Azure points to the correct AI service URL as shown above.

## React Frontend Deployment

The frontend can be deployed to **Azure Static Web Apps** or an **Azure Storage Static Website**.  
After deployment, it will receive an address similar to:

https://your-frontend.azurestaticapps.net

Before building for production, update the API endpoint in the React application so that it points to the deployed backend:

REACT\_APP\_API\_URL=https://your-backend.azurewebsites.net/api/process

Rebuild the frontend using:

npm run build

and deploy the contents of the build folder.

## Communication Flow on Azure

Once all components are deployed, the request flow mirrors the local setup, except each component communicates using its public Azure URLs:

React (Azure Static Web App)

↓

.NET API (Azure App Service)

↓

Python AI Service (Azure App Service or Container)

When a user uploads an image through the frontend, it is sent to the backend API. The backend forwards the file to the AI service for analysis, receives the result, and then returns it to the frontend for display.

**4.** Summary

In the local environment, the services communicate through:

* React: http://localhost:3000
* .NET API: https://localhost:5001
* Python AI: http://localhost:5002

When deployed to Azure, replace these local addresses with the Azure endpoints for each service. The structure and data flow remain the same. This ensures the system behaves consistently whether running locally for development or deployed in the cloud for production use.