# DingaVision Deployment Guide

Concordia Chemistry Robotics Project

## Overview

This guide provides the steps to deploy and run the DingaVision web application on the Dell server. The application enables live camera input or file upload, generates a caption using the BLIP model, and allows for user-interactive interpretation of the image content. We will replace this with a more advanced model like Gemma 3 or Deepseek.

## Requirements

- Windows machine (Dell preferred)
- Python 3.10+
- Access to GitHub repository

## Installation Steps

### 1. Create Virtual Environment

```
python 3.13 -m venv .venv
```

### 2. Activate the Virtual Environment

- On **Windows CMD**:

  ```
  .venv\Scripts\activate
  ```

### 3. Install Dependencies

```
python 3.31 -m pip install -r requirements.txt
```

### 4. Requirements File

The `requirements.txt` file lists all the Python packages required to run the project. To install the dependencies, run:

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

This will install packages such as `fastapi`, `uvicorn`, `torch`, `transformers`, and `Pillow`, among others.

### To Create or Update the File

If you have manually installed packages and want to generate a fresh `requirements.txt`, run:

```
pip freeze > requirements.txt
```

This captures the current environment's package list. Be cautious: this will include all packages, including transient ones, so it is recommended to clean up unused dependencies first.

### Recommended Example

Below is a recommended minimal `requirements.txt` compatible with Python 3.10–3.13:

```
fastapi
uvicorn
torch
transformers
Pillow
```

Version pinning (e.g., `torch==2.1.0`) can be added for reproducibility, but it may cause incompatibility if a different Python version is used.

## Run the Server

```
uvicorn main:app --host 0.0.0.0 --port 8000
```

The application will be available at: `http://localhost:8000` or via the Dell's IP on your local network (e.g., `http://192.168.50.241:8000`).

## Allow Camera Permissions

- Use Google Chrome or Firefox.
- For **local connections** (`localhost`), camera access should work by default.
- For **network connections** (e.g., `192.168.x.x`), browsers may block camera access due to HTTP insecurity.
- In Firefox, set the following flags by navigating to `about:config`:
  - `media.devices.insecure.enabled = true`
- Still blocked? Use localhost for initial testing or configure HTTPS (not yet required for this demo).

## Directory Structure

```
DingaVision/

         static/
              index.html         % Main frontend HTML file

         main.py               % FastAPI backend script
         requirements.txt      % Python dependencies
         .venv/                % Virtual environment
```

## GitHub Hosting Notes

Ensure your repository includes:

- `main.py`
- `requirements.txt`
- `static/index.html`

## Notes on Model Caching

- First-time execution will download the AI model.
- Model will be cached under `C:\Users\<username>\.cache\huggingface\hub`.
- Warning about symlinks on Windows can be safely ignored unless disk space becomes an issue.

## Troubleshooting

- **Camera doesn't appear**: Make sure only one browser tab is open and that the camera isn't locked by another app.
- **Model not found or timeout**: Ensure internet is active on first launch; check for typos in model name.
- **CORS errors**: Ensure frontend is accessing the correct server port.
- **Permission errors**: Avoid running from folders with restricted access.