Phytora Deployment Guide

Table of Contents

1. Local Development Setup
   - Prerequisites
   - Frontend Setup (Next.js)
   - Backend Setup
   - MongoDB Configuration
   - Local Storage Setup
2. Machine Learning API Setup
   - Environment Setup
   - Model Configuration
   - API Deployment
3. Production Deployment
   - Vercel Deployment (Frontend)
   - Backend Deployment
   - Database Configuration
   - Environment Variables
4. Security Considerations
5. Troubleshooting Common Issues

1. Local Development Setup

Prerequisites

- Node.js 16+ and npm
- Python 3.8+
- MongoDB Community Edition 5.0+
- Git

Frontend Setup (Next.js)

1. Clone the repository:

```
git clone https://github.com/htmw/2025S-SALAAR.git
cd 2025S-SALAAR/Code/frontend
```

2. Install dependencies:

```
npm install
```

3. Create a .env.local file in the frontend directory with the following variables:

```
NEXT_PUBLIC_API_URL=http://localhost:3000/api
NEXT_PUBLIC_ML_API_URL=http://localhost:5000/detect
MONGODB_URI=mongodb://localhost:27017/phytora
```

4. Start the development server:

```
npm run dev
```

This will start the frontend application on http://localhost:3000.

Backend Setup

1. Navigate to the backend directory:

```
cd ../backend
```

2. Install dependencies:

```
npm install
```

3. Create a .env file in the backend directory:

```
PORT=4000
MONGODB_URI=mongodb://localhost:27017/phytora
JWT_SECRET=your_jwt_secret_key
CORS_ORIGIN=http://localhost:3000
```

4. Start the backend server:

```
npm run dev
```

The backend server will run on http://localhost:4000.

MongoDB Configuration

1. Install MongoDB Community Edition following the official instructions for your operating system.
2. Start the MongoDB service:

```
# Linux
sudo systemctl start mongod

# macOS
brew services start mongodb-community

# Windows (run as administrator)
net start MongoDB
```

3. Create the Phytora database:

```
mongosh
> use phytora
> db.createCollection(”scanHistory”)
> db.createCollection(”diseases”)
```

4. Import initial disease data:

```
mongoimport --db phytora --collection diseases --file ./data/diseases.json --jsonArray
```

Local Storage Setup

1. Create required directories for image storage:

```
mkdir -p public/uploads/images
chmod 755 public/uploads/images
```

2. Configure storage settings in the backend:

```
// Update the storage configuration in the backend config file
{
  "storage": {
    "type": "local",
    "basePath": "public/uploads",
    "baseUrl": "/uploads"
  }
}
```

2. Machine Learning API Setup

Environment Setup

1. Navigate to the machine learning directory:

```
cd ../machine_learning
```

2. Create a Python virtual environment:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

3. Install required dependencies:

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

Model Configuration

1. Download pre-trained model files:

```
# Create model directory
mkdir -p models

# Download model files using the provided script
python download_models.py
```

2. Configure model parameters in config.py:

```
# Adjust model parameters if needed
MODEL_CONFIG = {
  ”input_size”: (224, 224),
  ”confidence_threshold”: 0.7,
  ”model_path”: ”models/plant_disease_model.h5”
}
```

API Deployment

1. Create a .env file for the ML API:

```
PORT=5000
MODEL_PATH=models/plant_disease_model.h5
DEBUG=False
ALLOWED_ORIGINS=http://localhost:3000,http://localhost:4000
```

2. Start the ML API server:

```
python app.py
```

The machine learning API will run on http://localhost:5000.

3. Production Deployment

Vercel Deployment (Frontend)

1. Install Vercel CLI:

```
npm install -g vercel
```

2. Navigate to the frontend directory and login to Vercel:

```
cd 2025S-SALAAR/Code/frontend
vercel login
```

3. Configure environment variables for production:

```
vercel env add NEXT_PUBLIC_API_URL
vercel env add NEXT_PUBLIC_ML_API_URL
vercel env add MONGODB_URI
```

4. Deploy to production:

```
vercel --prod
```

Backend Deployment

Option 1: Digital Ocean App Platform

1. Install Digital Ocean CLI:

```
curl -sL
https://github.com/digitalocean/doctl/releases/download/v1.X.X/doctl-1.X.X-linux-amd6
4.tar.gz | tar -xzv
sudo mv doctl /usr/local/bin
```

2. Authenticate with Digital Ocean:

```
doctl auth init
```

3. Create app specification (app.yaml):

```
name: phytora-backend
region: nyc
services:
- name: api
  github:
    repo: htmw/2025S-SALAAR
    branch: main
    deploy_on_push: true
  source_dir: /Code/backend
  environment_slug: node-js
  envs:
  - key: NODE_ENV
    value: production
  - key: PORT
    value: 8080
  - key: MONGODB_URI
    scope: RUN_TIME
    value: ${mongodb_uri}
  - key: JWT_SECRET
    scope: RUN_TIME
    value: ${jwt_secret}
  - key: CORS_ORIGIN
    value: https://your-frontend-url.vercel.app
```

4. Deploy using App Platform:

```
doctl apps create --spec app.yaml
```

Option 2: AWS Elastic Beanstalk

1. Install EB CLI:

```
pip install awsebcli
```

2. Initialize EB application:

```
cd 2025S-SALAAR/Code/backend
eb init
```

3. Create environment:

```
eb create phytora-production
```

4. Set environment variables:

```
eb setenv NODE_ENV=production PORT=8080 MONGODB_URI=your_mongodb_uri
JWT_SECRET=your_jwt_secret CORS_ORIGIN=https://your-frontend-url.vercel.app
```

5. Deploy:

```
eb deploy
```

Database Configuration (MongoDB Atlas)

1. Create a MongoDB Atlas account and set up a new cluster.
2. Configure database user and network access:
    - Create a database user with appropriate permissions
    - Add your IP to the IP whitelist or allow access from anywhere (0.0.0.0/0)
3. Obtain connection string:
    - Go to "Connect" > "Connect your application"
    - Select Node.js driver and copy the connection string
    - Replace `<password>` with your database user password
4. Update connection string in environment variables for both frontend and backend applications.

ML API Deployment

Using Google Cloud Run

1. Install Google Cloud SDK and initialize:

```
gcloud init
gcloud auth configure-docker
```

2. Build Docker image:

```
cd 2025S-SALAAR/Code/machine_learning
docker build -t gcr.io/[PROJECT_ID]/phytora-ml-api .
```

3. Push to Google Container Registry:

```
docker push gcr.io/[PROJECT_ID]/phytora-ml-api
```

4. Deploy to Cloud Run:

```
gcloud run deploy phytora-ml-api \
  --image gcr.io/[PROJECT_ID]/phytora-ml-api \
  --platform managed \
  --region us-central1 \
  --allow-unauthenticated
```

5. Update the ML API URL in the frontend environment variables.

4. Security Considerations

1. **API Key Protection**:
   - Never commit API keys or secrets to version control
   - Always use environment variables for sensitive information
2. **Image Validation**:
   - Implement server-side validation for all uploaded images
   - Restrict file types to jpg, jpeg, and png
   - Limit file size to prevent DOS attacks
3. **MongoDB Security**:
   - Use strong passwords for database users
   - Restrict network access to trusted IP addresses
   - Enable MongoDB authentication
   - Configure proper access controls and roles
4. **CORS Configuration**:
   - Restrict Cross-Origin Resource Sharing to known domains
   - Update CORS settings when moving to production
5. **Regular Updates**:
   - Keep dependencies updated
   - Run security audits regularly (npm audit)
   - Update the ML model as new training data becomes available

5. Troubleshooting Common Issues

Frontend Issues

1. **API Connection Failures:**
    - Verify API URLs in environment variables
    - Check CORS configuration on the backend
    - Verify network connectivity
2. **Image Upload Problems:**
    - Check directory permissions
    - Verify file size limits
    - Inspect browser console for errors

Backend Issues

1. **MongoDB Connection Errors:**
    - Verify MongoDB is running
    - Check connection string format
    - Test connection using MongoDB Compass
2. **Performance Issues:**
    - Monitor server resource usage
    - Implement database indexes for frequently queried fields
    - Consider caching strategies for repeated requests

ML API Issues

1. **Model Loading Errors:**
    - Verify model files exist in the correct location
    - Check model version compatibility
    - Ensure sufficient memory for model loading
2. **Slow Inference Time:**
    - Consider model optimization techniques
    - Implement batch processing for multiple images
    - Monitor resource usage during inference
3. **Accuracy Problems:**

  - Review confidence thresholds
  - Consider retraining with more diverse data
  - Implement model version tracking