ImageMedix Deployment Manual

Introduction

This deployment manual provides comprehensive instructions for setting up the ImageMedix application infrastructure from the repository at https://github.com/htmw/2025S-The-Minions/wiki. The application consists of three main components:

1. Node.js Backend (deployed on Heroku)
2. Machine Learning Backend (deployed separately)
3. Next.js Frontend (deployed on Vercel)

Table of Contents

1. Prerequisites

Before beginning deployment, ensure you have:

- Heroku account
- Vercel account
- Git installed locally
- Node.js (v16 or later) installed locally
- npm or yarn package manager
- Access to the ImageMedix repository (https://github.com/htmw/2025S-The-Minions/wiki)

2. Backend Deployment (Heroku)

2.1 Prepare the Backend for Deployment

1. Clone the repository:
2. `git clone https://github.com/htmw/2025S-The-Minions.git`
3. `cd 2025S-The-Minions`
4. Create a new `package.json` file in the root of the server directory if it doesn't exist:
5. ```
   {
   ```
6. ```
     "name": "imagemedix-backend",
   ```
7. ```
     "version": "1.0.0",
   ```
   ```

```
  "description": "ImageMedix Backend API",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "mongoose": "^7.0.0",
    "multer": "^1.4.5-lts.1",
    "axios": "^1.3.4",
    "jsonwebtoken": "^9.0.0",
    "bcrypt": "^5.1.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

Create a `Procfile` in the server directory:

```
web: npm start
```

Create a `.env` file in the server directory for local development:

```
NODE_ENV=development
PORT=8080
MONGODB_URI=mongodb://localhost:27017/imagemedix
JWT_SECRET=your_local_jwt_secret
ML_API_URL=http://localhost:5000
CORS_ORIGIN=http://localhost:3000
```

2.2 Deploy to Heroku

1. Install the Heroku CLI and log in:

```
npm install -g heroku
heroku login
```

2. Create a new Heroku app:

```
cd server
heroku create imagemedix-backend
```

3. Add the MongoDB add-on for database storage:

```
heroku addons:create mongodb:hobby-dev
```

4. Configure environment variables:

```
heroku config:set NODE_ENV=production
heroku config:set JWT_SECRET=your_production_jwt_secret
heroku config:set ML_API_URL=https://your-ml-api-url.herokuapp.com
heroku config:set CORS_ORIGIN=https://imagemedix.vercel.app
```

5. Deploy the backend to Heroku:

```
git subtree push --prefix server heroku main
```

If the above command fails, you can try:

```
git push heroku `git subtree split --prefix server main`:main --force
```

6. Ensure at least one instance is running:

```
heroku ps:scale web=1
```

18. Check the logs to verify the deployment:
19. `heroku logs --tail`

3. Machine Learning Backend Setup

3.1 Prepare the ML Service

1. Navigate to the ML service directory:
2. `cd ml-service`
3. Create a `requirements.txt` file if it doesn't exist:
4. `flask==2.2.3`
5. `gunicorn==20.1.0`
6. `numpy==1.24.2`
7. `tensorflow==2.11.0`
8. `pillow==9.4.0`
9. `scikit-learn==1.2.2`
10. `opencv-python-headless==4.7.0.72`
11. Ensure the Dockerfile exists and contains:
12. `FROM python:3.10-slim`
13.
14. `WORKDIR /app`
15.
16. `COPY requirements.txt .`
17. `RUN pip install --no-cache-dir -r requirements.txt`
18.
19. `COPY . .`
20.
21. `EXPOSE 5000`
22.
23. `CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]`

3.2 Deploy ML Service to Heroku

1. Create a new Heroku app for the ML service:
2. `heroku create imagemedix-ml-service`
3. Log in to Heroku Container Registry:
4. `heroku container:login`
5. Build and push the Docker container to Heroku:
6. `heroku container:push web --app imagemedix-ml-service`
7. Release the container:
8. `heroku container:release web --app imagemedix-ml-service`
9. Configure environment variables:
10. `heroku config:set MODEL_PATH=/app/models --app imagemedix-ml-service`
11. `heroku config:set ALLOWED_ORIGINS=https://imagemedix-backend.herokuapp.com --app imagemedix-ml-service`
12. Check the logs to verify deployment:
13. `heroku logs --tail --app imagemedix-ml-service`

4. Frontend Deployment (Vercel)

4.1 Prepare the Frontend

1. Navigate to the app directory:
2. `cd app`
3. Install dependencies:
4. `npm install`
5. Create a `.env.local` file:
6. `NEXT_PUBLIC_API_URL=https://imagemedix-backend.herokuapp.com/api`
7. `NEXT_PUBLIC_ML_API_URL=https://imagemedix-ml-service.herokuapp.com`
8. `CLERK_PUBLISHABLE_KEY=your_clerk_publishable_key`
9. `CLERK_SECRET_KEY=your_clerk_secret_key`
10. Verify the build works locally:
11. `npm run build`
12. `npm run start`

4.2 Deploy to Vercel

1. Install the Vercel CLI:
2. `npm install -g vercel`
3. Log in to Vercel:
4. `vercel login`
5. Deploy to Vercel:
6. `vercel`

   Follow the prompts to set up the project:

   - Set the root directory to `app`
   - Configure the build settings
   - Add environment variables
7. For production deployment:
8. `vercel --prod`
9. Alternatively, connect your GitHub repository to Vercel:
   - Go to https://vercel.com/new
   - Import your GitHub repository
   - Configure the project settings (set root directory to `app`)
   - Add environment variables
   - Deploy

5. Environment Configuration

5.1 Backend Environment Variables

Variable Name	Description	Example Value
NODE_ENV	Environment mode	production
PORT	Port the server runs on	8080
MONGODB_URI	MongoDB connection string	mongodb://user:pass@host:port/db

JWT_SECRET	Secret for JWT tokens	your_jwt_secret_key
ML_API_URL	URL for the ML service	https://imagemedix-ml-service.herokuapp.com
CORS_ORIGIN	Allowed origins for CORS	https://imagemedix.vercel.app

5.2 ML Service Environment Variables

Variable Name	Description	Example Value
MODEL_PATH	Path to model files	/app/models
ALLOWED_ORIGINS	Allowed origins for CORS	https://imagemedix-backend.herokuapp.com
LOG_LEVEL	Logging level	info

5.3 Frontend Environment Variables

Variable Name	Description	Example Value
NEXT_PUBLIC_API_URL	Backend API URL	https://imagemedix-backend.herokuapp.com/api
NEXT_PUBLIC_ML_API_URL	ML service URL	https://imagemedix-ml-service.herokuapp.com
CLERK_PUBLISHABLE_KEY	Clerk authentication key	pk_test_XXXXX
CLERK_SECRET_KEY	Clerk authentication secret	sk_test_XXXXX

6. Testing the Deployment

6.1 Backend Testing

1. Test the health endpoint:
2. `curl https://imagemedix-backend.herokuapp.com/api/health`
3. Test user registration:
4. `curl -X POST https://imagemedix-backend.herokuapp.com/api/auth/register \`
5. `  -H "Content-Type: application/json" \`
6. `  -d '{"name":"Test User","email":"test@example.com","password":"password123"}'`
7. Test user login:
8. `curl -X POST https://imagemedix-backend.herokuapp.com/api/auth/login \`
9. `  -H "Content-Type: application/json" \`
10. `  -d '{"email":"test@example.com","password":"password123"}'`

6.2 ML Service Testing

1. Test the health endpoint:

2. curl https://imagemedix-ml-service.herokuapp.com/health
3. Test the prediction endpoint with a sample chest X-ray:
4. curl -X POST https://imagemedix-ml-service.herokuapp.com/api/ml/analyze-chest \
5.   -F "image=@sample-xray.jpg" \
6.   -H "Content-Type: multipart/form-data"

6.3 Frontend Testing

1. Open the deployed frontend in a browser:
2. https://imagemedix.vercel.app
3. Test all main flows:
    - Registration and login
    - Uploading medical scans
    - Viewing scan analysis results
    - Accessing scan history
    - Updating settings

7. Maintenance and Monitoring

7.1 Backend Monitoring

1. View Heroku logs:
2. heroku logs --tail --app imagemedix-backend
3. Set up Heroku metrics:
4. heroku addons:create librato:development --app imagemedix-backend
5. Set up error tracking with Sentry:
6. heroku addons:create sentry:f1 --app imagemedix-backend

7.2 ML Service Monitoring

1. View ML service logs:
2. heroku logs --tail --app imagemedix-ml-service
3. Monitor ML service performance:
4. heroku addons:create newrelic:wayne --app imagemedix-ml-service

7.3 Frontend Monitoring

1. View Vercel deployment logs from the Vercel dashboard
2. Set up analytics:
    - Add Vercel Analytics by enabling it in the Vercel dashboard
    - Implement custom analytics using a service like Plausible or Umami

7.4 Scaling Considerations

1. Backend scaling on Heroku:
2. heroku ps:scale web=2 --app imagemedix-backend
3. ML service scaling:

4. `heroku ps:scale web=2 --app imagemedix-ml-service`
5. Database scaling:
    - Upgrade MongoDB plan as needed
    - Consider adding read replicas for high traffic

7.5 Backup Procedures

1. Set up automatic MongoDB backups:
2. `heroku addons:create mongolab:backup-daily --app imagemedix-backend`
3. Implement regular model backups:
    - Store ML model files in a separate storage service (AWS S3)
    - Version control your models
    - Document model versions and changes

Conclusion

Following this deployment manual will result in a fully functional ImageMedix application with:

- Node.js backend hosted on Heroku
- Custom ML model service running on a separate Heroku instance
- Next.js frontend deployed on Vercel

Once deployed, the application will be accessible at your custom domain (e.g., `imagemedix.vercel.app`), connecting to your backend (e.g., `imagemedix-backend.herokuapp.com/api`) and utilizing your ML service (e.g., `imagemedix-ml-service.herokuapp.com`) for image analysis.

After completing the deployment, be sure to test the connections between all components thoroughly before making the application available to users.