Hospital Management System - Deployment Guide

This comprehensive guide will help you set up and run the Hospital Management System on your local machine. Follow each step carefully to ensure a successful deployment.

Table of Contents

A. Environment Setup

Prerequisites

Before you begin, ensure you have the following installed on your system:

- **Python 3.8 or higher** (Python 3.9+ recommended)
- **pip** (Python package installer)
- **virtualenv** (Python virtual environment tool)

Step 1: Check Python Version

Open your terminal/command prompt and verify your Python installation:

```
python --version
```

Or try:

```
python3 --version
```

Expected Output: `Python 3.8.x` or higher (e.g., `Python 3.9.7`, `Python 3.10.5`, `Python 3.11.3`)

> **Note:** If Python is not installed or the version is below 3.8, download and install the latest version from python.org (https://www.python.org/downloads/)

Step 2: Navigate to Project Directory

Open your terminal/command prompt and navigate to the project directory:

```
cd /path/to/hospital_management_system
```

Replace `/path/to/hospital_management_system` with the actual path where your project is located.

Example:

```
cd /home/ubuntu/hospital_management_system
```

Or on Windows:

```
cd C:\Users\YourName\hospital_management_system
```

Step 3: Create a Virtual Environment

Creating a virtual environment ensures that all project dependencies are isolated from your system Python installation.

Command:

```
python -m venv venv
```

Or if `python` doesn't work:

```
python3 -m venv venv
```

This creates a folder named `venv` in your project directory containing the virtual environment.

Step 4: Activate the Virtual Environment

Activation commands differ by operating system:

Windows (Command Prompt):

```
venv\Scripts\activate
```

Windows (PowerShell):

```
venv\Scripts\Activate.ps1
```

> **Note:** If you encounter a permissions error in PowerShell, run:
> `powershell`
> `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

macOS/Linux:

```
source venv/bin/activate
```

How to verify activation:
Once activated, your terminal prompt should be prefixed with `(venv)`:

```
(venv) user@computer:~/hospital_management_system$
```

Step 5: Install Project Dependencies

With the virtual environment activated, install all required Python packages:

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

This will install:
- **Flask** (3.0.0) - Web framework
- **Flask-SQLAlchemy** (3.1.1) - Database ORM
- **Flask-Login** (0.6.3) - User authentication
- **Flask-Mail** (0.9.1) - Email notifications
- **Werkzeug** (3.0.1) - WSGI utilities
- **SQLAlchemy** (2.0.23) - Database toolkit
- **email-validator** (2.1.0) - Email validation
- **python-dotenv** (1.0.0) - Environment variable management

Expected Output:

```
Successfully installed Flask-3.0.0 Flask-SQLAlchemy-3.1.1 ...
```

> **Troubleshooting:** If you encounter errors:
> - Ensure pip is up to date: `pip install --upgrade pip`
> - Use `pip3` instead of `pip` if necessary
> - Check your internet connection

B. Database Initialization

Automatic Database Creation

The Hospital Management System uses SQLite, and the database is **created automatically** when you first run the application. You don’t need to manually create any database files.

Step 1: First Run Initialization

When you run the application for the first time (see Section C below), the system will:

1. Create a file named `hospital.db` in your project root directory
2. Set up all required tables (Users, Doctors, Patients, Appointments, Treatments, etc.)
3. Create a default **Admin** user with the following credentials:

Default Admin Credentials:
- **Username:** `admin`
- **Password:** `admin123`

> **Important:** Change the admin password immediately after your first login for security purposes!

Step 2: Verify Database Creation

After running the application once, check that the database file exists:

Windows (Command Prompt):

```
dir hospital.db
```

macOS/Linux:

```
ls -lh hospital.db
```

Expected Output:

```
-rw-r--r-- 1 user user 64K Nov 12 10:30 hospital.db
```

The file size will vary depending on the data you add.

Step 3: Database Location

The `hospital.db` file is located in your project root directory:

```
hospital_management_system/
├── app.py
├── hospital.db          ← Database file (created automatically)
├── requirements.txt
├── models/
├── routes/
├── static/
└── templates/
```

Step 4: Reset Database (If Needed)

If you need to reset the database to its initial state:

1. **Stop the application** (press `Ctrl+C` in terminal)
2. **Delete the database file:**

Windows:
`bash`
`del hospital.db`

macOS/Linux:
`bash`
`rm hospital.db`

1. **Restart the application** - A fresh database will be created automatically

> **Warning:** Deleting the database will remove ALL data (patients, doctors, appointments, etc.). Make sure to back up important data before resetting.

Database Backup (Recommended)

To create a backup of your database:

Windows:

```
copy hospital.db hospital_backup_%date:~-4,4%%date:~-10,2%%date:~-7,2%.db
```

macOS/Linux:

```
cp hospital.db hospital_backup_$(date +%Y%m%d).db
```

C. Running the Application

Step 1: Ensure Virtual Environment is Activated

Make sure your virtual environment is active (you should see `(venv)` in your terminal prompt).

If not activated, refer to Section A, Step 4.

Step 2: Start the Flask Application

Run the main application file:

```
python app.py
```

Or:

```
python3 app.py
```

Step 3: Expected Terminal Output

You should see output similar to this:

```
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 123-456-789
```

Key Information:
- The application is running at `http://127.0.0.1:5000` (localhost on port 5000)
- Debug mode is enabled for development
- Press `Ctrl+C` to stop the server

Step 4: Access the Application in Your Browser

Open your web browser and navigate to:

```
http://127.0.0.1:5000
```

Or alternatively:

```
http://localhost:5000
```

You should see the Hospital Management System homepage!

Step 5: Login with Different Roles

Admin Login:

- URL: `http://127.0.0.1:5000/admin/login`
- Username: `admin`
- Password: `admin123`

Doctor Login:

- URL: `http://127.0.0.1:5000/doctor/login`
- Use credentials created by admin (default: `dr.sharma` / `doctor123`)

Patient Login:

- URL: `http://127.0.0.1:5000/patient/login`
- Register as a new patient or use existing credentials

Step 6: Stop the Application

To stop the Flask server:

1. Go to the terminal window where the server is running
2. Press `Ctrl+C` (or `Cmd+C` on Mac)

Expected Output:

```
^C
KeyboardInterrupt
(venv) user@computer:~/hospital_management_system$
```

Step 7: Restart the Application

To restart the server, simply run:

```
python app.py
```

The database persists between restarts, so all your data remains intact.

Running on a Different Port (Optional)

If port 5000 is already in use, you can run the app on a different port:

Edit `app.py` and change the last line:

```
# Change from:
app.run(debug=True)

# To:
app.run(debug=True, port=5001)
```

Then access the application at `http://127.0.0.1:5001`

D. Email Configuration (Optional)

The Hospital Management System can send email notifications for appointments. This feature is **optional** but enhances user experience.

Step 1: Create a `.env` File

In your project root directory, create a file named `.env`:

Windows (Command Prompt):

```
echo. > .env
```

macOS/Linux:

```
touch .env
```

Step 2: Configure SMTP Settings

Open the `.env` file in a text editor and add the following configuration:

```
# Email Configuration
MAIL_SERVER=smtp.gmail.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=your-email@gmail.com
MAIL_PASSWORD=your-app-password
MAIL_DEFAULT_SENDER=your-email@gmail.com
```

Replace:
- `your-email@gmail.com` with your Gmail address
- `your-app-password` with your Gmail App Password (see below)

Step 3: Generate Gmail App Password

Gmail requires an “App Password” for third-party applications. Follow these steps:

1. **Enable 2-Step Verification:**
   - Go to Google Account Security (https://myaccount.google.com/security)
   - Enable “2-Step Verification” if not already enabled

2. **Generate App Password:**
   - Visit Google App Passwords (https://myaccount.google.com/apppasswords)
   - Select “Mail” as the app
   - Select “Other” as the device and name it “Hospital Management System”
   - Click “Generate”
   - Copy the 16-character password (format: `xxxx xxxx xxxx xxxx`)

3. **Update `.env` file:**
   `env`
   `MAIL_PASSWORD=xxxxxxxxxxxxxx  # Paste the 16-character password (remove spaces)`

Step 4: Using Other Email Providers

If you’re using a different email provider, update the SMTP settings accordingly:

Outlook/Hotmail:

```
MAIL_SERVER=smtp-mail.outlook.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=your-email@outlook.com
MAIL_PASSWORD=your-password
MAIL_DEFAULT_SENDER=your-email@outlook.com
```

Yahoo Mail:

```
MAIL_SERVER=smtp.mail.yahoo.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=your-email@yahoo.com
MAIL_PASSWORD=your-app-password
MAIL_DEFAULT_SENDER=your-email@yahoo.com
```

Step 5: Test Email Notifications

1. **Restart the application** (email configuration is loaded on startup)
2. **Log in as a patient**
3. **Book an appointment**
4. **Check your email** - You should receive a confirmation email

Step 6: Troubleshooting Email Issues

If emails are not being sent:

1. **Check `.env` file format:**
   - No extra spaces around `=`
   - Correct email and password
   - File is named exactly `.env` (not `.env.txt`)

2. **Verify Gmail settings:**
   - 2-Step Verification is enabled
   - App Password is correct (16 characters without spaces)
   - Less Secure App Access is NOT needed for App Passwords

3. **Check application logs:**
   Look for email-related errors in the terminal where the Flask app is running

4. **Test SMTP connection:**
   `python`
   `# Run this in Python console to test`
   `import smtplib`
   `server = smtplib.SMTP('smtp.gmail.com', 587)`
   `server.starttls()`
   `server.login('your-email@gmail.com', 'your-app-password')`
   `print("SMTP connection successful!")`
   `server.quit()`

Disabling Email Notifications

If you prefer to run the application without email notifications:

- Simply don't create the `.env` file
- Or leave the email fields blank in `.env`
- The application will work normally, just without sending emails

---

Quick Start Summary

For experienced users, here's a quick command summary:

```
# 1. Navigate to project directory
cd hospital_management_system

# 2. Create and activate virtual environment
python -m venv venv
source venv/bin/activate  # macOS/Linux
# OR
venv\Scripts\activate  # Windows

# 3. Install dependencies
pip install -r requirements.txt

# 4. Run the application
python app.py

# 5. Access in browser
# http://127.0.0.1:5000

# 6. Login as Admin
# Username: admin
# Password: admin123
```

Next Steps

After successful deployment:

1. ✅ **Test all features** - Use the `TESTING_CHECKLIST.md` for comprehensive testing
2. ✅ **Change default passwords** - Update admin password for security
3. ✅ **Add sample data** - Create test doctors, patients, and appointments
4. ✅ **Configure emails** - Set up email notifications for better user experience
5. ✅ **Review troubleshooting guide** - Check `TROUBLESHOOTING.md` for common issues

---

Support & Resources

- **Project Documentation:** `README.md`
- **Testing Guide:** `TESTING_CHECKLIST.md`
- **Troubleshooting:** `TROUBLESHOOTING.md`

- **Submission Guide:** `SUBMISSION_CHECKLIST.md`

Important Notes

⚠️ **Security Reminders:**
- Change default admin credentials immediately
- Never commit `.env` file to version control
- Keep your database file secure
- Use strong passwords for all accounts

⚠️ **Development Environment:**
- This setup is for **development/testing only**
- Do NOT use `debug=True` in production
- Use a proper WSGI server (Gunicorn/uWSGI) for production deployment

⚠️ **Project Submission:**
- Do NOT include `hospital.db` in your ZIP submission
- Do NOT include the `venv` folder in your submission
- Use the `create_submission_zip.py` script for proper packaging

Happy Coding! 🏥💻