Postnatal Stories - Installation Guide

Table of contents

System Requirements

Minimum Hardware Requirements

- **CPU**: 2 cores, 2.0 GHz
- **RAM**: 4 GB minimum, 8 GB recommended
- **Storage**: 10 GB free space
- **Network**: Stable internet connection for external services

Recommended Hardware (Production)

- **CPU**: 4+ cores, 3.0 GHz
- **RAM**: 16 GB or more

- **Storage**: 50 GB+ SSD storage
- **Network**: High-bandwidth connection

Operating System Support

- **Linux**: Ubuntu 20.04+, CentOS 8+, Debian 10+
- **Windows**: Windows 10/11, Windows Server 2019+
- **macOS**: macOS 10.15+

Prerequisites

Required Software

1. **Python 3.11+**
2. **Node.js 16+** (for development tools)
3. **MongoDB 5.0+** or MongoDB Atlas account
4. **Docker & Docker Compose** (for containerized deployment)
5. **Git** (for source code management)

External Services Required

1. **MongoDB Database** (local or MongoDB Atlas)
2. **OpenAI API** (for story generation)
3. **SMTP Email Service** (Gmail, SendGrid, or similar)
4. **Domain Name** (for production deployment)

Development Tools (Optional)

- **Visual Studio Code** or preferred IDE
- **Postman** for API testing
- **MongoDB Compass** for database management

Environment Setup

1. Install Python 3.11+

Linux (Ubuntu/Debian)

```
bash
sudo apt update
sudo apt install python3.11 python3.11-pip python3.11-venv
```

Windows

1. Download Python from https://python.org
2. Run installer with "Add to PATH" checked
3. Verify installation:

```
cmd
python --version
pip --version
```

macOS

```
bash
# Using Homebrew
brew install python@3.11
```

2. Create MongoDB Atlas Database (Cloud)

Step 1: Create MongoDB Atlas Account

1. Go to https://cloud.mongodb.com
2. Click **"Try Free"** to create a free account
3. Fill out the registration form:

- Email address
- Password (make it strong)
- First and Last name

4. Click **"Create your Atlas account"**
5. Verify your email address when prompted

Step 2: Create a Database Cluster

1. **Choose deployment option**: Select **"Shared"** (free tier)
2. **Choose cloud provider**: Select **"AWS"**, **"Google Cloud"**, or **"Azure"**
3. **Choose region**: Select the region closest to your users
4. **Cluster name**: Leave as default or change to "PostnatalStories"
5. Click **"Create Cluster"** (this takes 1-3 minutes)

Step 3: Create Database User

1. In the left sidebar, click **"Database Access"**
2. Click **"Add New Database User"**
3. **Authentication Method**: Select **"Password"**
4. **Username**: Enter `postnatal_user` (or your preferred username)
5. **Password**: Click **"Autogenerate Secure Password"** and **copy the password** somewhere safe
6. **Database User Privileges**: Select **"Read and write to any database"**
7. Click **"Add User"**

Step 4: Configure Network Access

1. In the left sidebar, click **"Network Access"**
2. Click **"Add IP Address"**
3. **For development**: Click **"Allow Access from Anywhere"** (0.0.0.0/0)
4. **For production**: Click **"Add Current IP Address"** and add your server's IP
5. Click **"Confirm"**

Step 5: Get Connection String

1. In the left sidebar, click **"Database"**
2. Click **"Connect"** on your cluster
3. Select **"Connect your application"**
4. **Driver**: Select **"Python"** and **"3.6 or later"**
5. **Copy the connection string** - it will look like:

```
mongodb+srv://postnatal_user:<password>@cluster0.xxxxx.mongodb.net/?retryWrites=true&w=majority
```

6. **Replace** `<password>` with the password you copied in Step 3
7. **Add database name** by changing the connection string to:

```
mongodb+srv://postnatal_user:your_password@cluster0.xxxxx.mongodb.net/postnatal_stories?retryWrites=true&w=majority
```

Step 6: Test Connection (Optional)

1. Download **MongoDB Compass** from https://www.mongodb.com/products/compass
2. Install and open MongoDB Compass
3. Paste your connection string and click **"Connect"**
4. You should see your `postnatal_stories` database (it will be empty initially)

3. Install Docker

Windows/macOS

- Download Docker Desktop from https://docker.com
- Follow installation wizard

Linux

```
bash
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER
```

Database Configuration

MongoDB Atlas Setup Complete

The database setup is complete after following the MongoDB Atlas creation steps above. The application will automatically:

1. **Connect to your Atlas cluster** using the connection string
2. **Create required collections** on first run:
   - `users` - User accounts and authentication
   - `pending_stories` - Stories awaiting moderation
   - `approved_stories` - Published stories
   - `rejected_stories` - Stories that didn't meet guidelines
   - `saved_stories` - User's saved story collections
3. **Create database indexes** automatically for optimal performance

Verify Database Connection

You can verify your database is properly configured by:

1. **Using MongoDB Compass** (if installed):
   - Connect using your connection string
   - You should see the `postnatal_stories` database
   - Collections will appear after running the application
2. **Checking application logs**:
   - When you start the application, you should see:

- `Connected to MongoDB successfully!`
- `Database indexes created successfully`

Important Notes

- **Connection String Security**: Keep your MongoDB connection string private and secure
- **Free Tier Limits**: MongoDB Atlas free tier includes 512MB storage (sufficient for development)
- **Automatic Backups**: Atlas provides automatic backups on paid tiers
- **Scaling**: You can upgrade to paid tiers for more storage and features as needed

---

Backend Installation

1. Extract Project Files

1. **Locate the project files** you received via email
2. **Extract the ZIP archive** to your desired location:
   - Right-click the ZIP file
   - Select "Extract All..." or "Extract Here"
   - Choose destination folder (e.g., `C:\postnatal-stories\`)
3. **Navigate to the extracted folder**
4. **Verify the project structure**:

```
postnatal-stories/
├── backend/
│   ├── main.py
│   ├── requirements.txt
│   ├── .env.example
│   └── ...
├── frontend/
```

```
│   ├── index.html
│   ├── css/
│   ├── js/
│   └── ...
├── docker-compose.yml
├── Dockerfile
└── nginx.conf
```

2. Create Virtual Environment

```
bash
cd backend
python -m venv venv

# Activate virtual environment# Linux/macOS:
source venv/bin/activate
# Windows:
venv\Scripts\activate
```

3. Install Dependencies

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

4. Create Environment Configuration

```
bash
# Create .env file in backend directory
copy .env.example .env
```

Edit the `.env` file with your configuration:

```
env
# Environment
ENVIRONMENT=development
DEBUG=true

# Database - Use your MongoDB Atlas connection string
MONGODB_URI=mongodb+srv://postnatal_user:your_password@cluster0.
xxxxx.mongodb.net/postnatal_stories?retryWrites=true&w=majority

# Authentication
JWT_SECRET_KEY=your-super-secret-jwt-key-minimum-32-characters-lon
g
JWT_ALGORITHM=HS256
ACCESS_TOKEN_EXPIRE_MINUTES=10080

# CORS
ALLOWED_ORIGINS=http://localhost:8080,http://localhost:3000,http://local
host:5500

# OpenAI API
OPENAI_API_KEY=your-openai-api-key-here

# Email Configuration (Choose one)
# Option 1: Gmail SMTP
SMTP_SERVER=smtp.gmail.com
SMTP_PORT=587
SMTP_USERNAME=your-email@gmail.com
SMTP_PASSWORD=your-app-password
EMAIL_FROM=your-email@gmail.com
EMAIL_FROM_NAME=Postnatal Stories

# Option 2: SendGrid (Alternative)
# EMAIL_PROVIDER=sendgrid
# SENDGRID_API_KEY=your-sendgrid-api-key
```

```
# EMAIL_FROM=noreply@yourdomain.com

# Story Matching
STORY_MATCHER_MODEL=all-distilroberta-v1
MODELS_CACHE_DIR=./ai_models
SIMILARITY_THRESHOLD=0.1
MAX_SIMILAR_STORIES=9

# Logging
LOG_LEVEL=INFO
LOG_FILE=logs/app.log
```

Important: Replace the `MONGODB_URI` with your actual connection string from MongoDB Atlas (Step 5 in the database setup).

5. Create Required Directories

```
bash
mkdir -p logs ai_models backups
```

6. Initialize Database

```
bash
# Run the application to auto-create indexes
python main.py
```

7. Create Admin User (Optional)

You can create an admin user through the MongoDB Atlas web interface:

Method 1: Using MongoDB Compass (Recommended)

1. **Open MongoDB Compass** and connect using your connection string
2. **Navigate** to the `postnatal_stories` database

3. **Click** on the `users` collection (it will be created after first app run)
4. **Click "Insert Document"**
5. **Add the following document** (replace the email and generate a proper password hash):

```
json
{
  "email": "admin@postnatalstories.com",
  "password_hash": "$2b$12$LQv3c1yqBwEHxaKuNJkfpefY4QPOGzs8Z8/
WEhHhCZLxtJ7xJTjHK",
  "display_name": "Administrator",
  "created_at": {"$date": "2025-01-01T00:00:00.000Z"},
  "last_login": null,
  "is_active": true,
  "role": "admin"
}
```

Note: The password hash above is for the password "admin123" - you should generate a proper hash for your desired password.

Method 2: Create via Application

1. **Start the application** (see next step)
2. **Register normally** through the web interface
3. **Manually update** the user role in MongoDB Atlas:
   - Find your user in the `users` collection
   - Edit the document
   - Change `"role": "user"` to `"role": "admin"`

Frontend Setup

1. Navigate to Frontend Directory

```
bash
cd ../frontend
```

2. Configure API Endpoint

Edit `frontend/js/main.js` and update the API_BASE_URL:

```
javascript
// For development
const API_BASE_URL = 'http://localhost:8000';

// For production
const API_BASE_URL = '/api';
```

3. Test Frontend

You can serve the frontend using:

Python HTTP Server

```
bash
python -m http.server 8080
```

Node.js HTTP Server

```
bash
npx http-server -p 8080
```

Live Server (VS Code Extension)

1. Install Live Server extension
2. Right-click `index.html`
3. Select "Open with Live Server"

Docker Deployment

1. Extract Project Files and Verify Docker

Extract the Project Files

1. **Check your email** for the Postnatal Stories project files (ZIP attachment)
2. **Save the ZIP file** to a convenient location (e.g., Downloads folder)
3. **Extract the files**:
   - Right-click the ZIP file
   - Select "Extract All..."
   - Choose destination: `C:\postnatal-stories\`
   - Click "Extract"

Verify Project Structure

Navigate to the extracted folder and ensure you have:

```
C:\postnatal-stories\
├── backend/
│   ├── main.py
│   ├── requirements.txt
│   ├── .env.example
│   └── ...
├── frontend/
│   ├── index.html
│   ├── css/
│   ├── js/
│   └── ...
```

```
├── docker-compose.yml
├── Dockerfile
└── nginx.conf
```

Verify Docker Installation

```
cmd
# Check Docker is running
docker --version
docker-compose --version
```

Expected output:

```
Docker version 24.0.x
Docker Compose version v2.20.x
```

2. Create Environment File

```
bash
# Copy environment template
cp backend/.env.example .env
# Edit .env with your production values
```

3. Build and Run with Docker Compose

```
bash
# Build and start all services
docker-compose up --build -d
```

```
# View logs
docker-compose logs -f

# Stop services
docker-compose down
```

4. Verify Docker Deployment

```
bash
# Check running containers
docker-compose ps

# Check backend health
curl http://localhost:8000/health

# Check frontend
curl http://localhost:8080
```

Production Deployment (Windows Server)

1. Windows Server Setup

System Requirements for Production

- **Windows Server 2019/2022** (recommended)
- **Windows 10/11 Pro** (for smaller deployments)
- **8GB+ RAM** for production workloads
- **50GB+ SSD storage**
- **Static IP address** or domain name

Install Required Software

1. Install Docker Desktop for Windows

1. Download Docker Desktop from https://docker.com/products/docker-desktop
2. Run installer as Administrator
3. Enable WSL 2 backend when prompted
4. Restart computer when installation completes
5. Start Docker Desktop and verify it's running

2. Install Text Editor (Recommended)

- Install Notepad++ (https://notepad-plus-plus.org) or Visual Studio Code for editing configuration files
- Built-in Windows Notepad can also be used

2. Clone and Configure Project

Extract Project Files

1. **Locate the email** containing the Postnatal Stories project files
2. **Download the ZIP attachment** to your desired location (e.g., `C:\Production\`)
3. **Extract the ZIP file**:
    - Right-click the ZIP file
    - Select "Extract All..."
    - Choose `C:\Production\` as the destination
    - This will create `C:\Production\postnatal-stories\`

Open Command Prompt or PowerShell as Administrator

```
cmd
# Navigate to the extracted project directory
cd C:\Production\postnatal-stories

# Verify project structure
dir
```

You should see these folders and files:

```
backend/
frontend/
docker-compose.yml
Dockerfile
nginx.conf
README.md (if included)
```

Create Production Environment File

```
cmd
# Create production environment file
copy backend\.env.example .env
```

Edit Environment Configuration

1. Open `.env` file in text editor (Notepad++, VS Code, or notepad)
2. Configure for production:

```
env
# Production settings
ENVIRONMENT=production
DEBUG=false

# Database - Use MongoDB Atlas connection string from your setup
MONGODB_URI=mongodb+srv://username:password@cluster.mongodb.ne
t/postnatal_stories

# Security - Generate a strong 32+ character secret
JWT_SECRET_KEY=your-production-super-secret-jwt-key-minimum-32-ch
```

```
aracters-long
ALLOWED_ORIGINS=https://yourdomain.com,https://www.yourdomain.com

# Email (SendGrid recommended for production)
EMAIL_PROVIDER=sendgrid
SENDGRID_API_KEY=your-production-sendgrid-api-key
EMAIL_FROM=noreply@yourdomain.com
EMAIL_FROM_NAME=Postnatal Stories

# OpenAI API
OPENAI_API_KEY=your-production-openai-api-key

# Logging
LOG_LEVEL=WARNING
LOG_FILE=logs/app.log
```

3. SSL Certificate Setup (Windows)

Option A: Using IIS with Let's Encrypt (Recommended)

Install IIS and URL Rewrite Module:

1. Open "Turn Windows features on or off"
2. Enable "Internet Information Services"
3. Enable "World Wide Web Services"
4. Download and install URL Rewrite Module from Microsoft

Install win-acme for Let's Encrypt:

1. Download win-acme from https://github.com/win-acme/win-acme
2. Extract to `C:\win-acme\`
3. Run as Administrator:

```
cmd
cd C:\win-acme
wacs.exe
```

1. Follow wizard to create certificate for your domain

Configure IIS:

1. Open IIS Manager
2. Add website with your domain name
3. Point to your application directory
4. Bind SSL certificate created by win-acme

Option B: Using Cloudflare (Easier for beginners)

1. Sign up for Cloudflare
2. Add your domain to Cloudflare
3. Update nameservers at your domain registrar
4. Enable "Full (strict)" SSL mode
5. Cloudflare will handle SSL termination

4. Windows Firewall Configuration

Configure Windows Firewall

```
cmd
# Open ports for HTTP and HTTPS
netsh advfirewall firewall add rule name="HTTP" dir=in action=allow protoc
ol=TCP localport=80
netsh advfirewall firewall add rule name="HTTPS" dir=in action=allow prot
ocol=TCP localport=443
netsh advfirewall firewall add rule name="Docker API" dir=in action=allow
protocol=TCP localport=8000
```

5. Deploy with Docker

Start Docker Desktop

1. Open Docker Desktop
2. Ensure it shows "Docker Desktop is running"

3. Open Command Prompt as Administrator in project directory

Build and Deploy

```
cmd
# Navigate to project directory
cd C:\Production\postnatal-stories

# Build and start services
docker-compose up --build -d

# Verify containers are running
docker-compose ps
```

Expected Output:

```
NAME                    IMAGE                     COMMAND                  STATUS
postnatal-stories-backend-1   postnatal-stories-backend   "uvicorn main:
app --..."   Up
postnatal-stories-frontend-1   nginx:alpine              "/docker-entrypoin
t...."   Up
```

6. Configure Reverse Proxy (Production)

Update nginx.conf for Windows Production

Create `nginx-production.conf`:

```
nginx
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
```

```
# Redirect all HTTP to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;
    root /usr/share/nginx/html;
    index index.html;

# SSL Configuration (if using direct SSL)
    ssl_certificate /etc/ssl/certs/yourdomain.crt;
    ssl_certificate_key /etc/ssl/private/yourdomain.key;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

# Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header Strict-Transport-Security "max-age=31536000; includeSub
Domains" always;

# Main site
    location / {
        try_files $uri $uri/ /index.html;
    }

# API proxy to backend
    location /api/ {
        proxy_pass http://backend:8000/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

# CORS headers
```

```
        add_header Access-Control-Allow-Origin $http_origin;
        add_header Access-Control-Allow-Methods "GET, POST, PUT, DELET
E, OPTIONS";
        add_header Access-Control-Allow-Headers "Authorization, Content-T
ype, X-Request-ID";
        add_header Access-Control-Allow-Credentials true;
    }

# Cache static assets
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
```

7. Windows Service Setup (Optional)

Create Windows Service for Docker Compose

Create start-service.bat:

```
batch
@echo off
cd /d C:\Production\postnatal-stories
docker-compose up -d
```

Create stop-service.bat:

```
batch
@echo off
cd /d C:\Production\postnatal-stories
docker-compose down
```

Install as Windows Service using NSSM:

1. Download NSSM from https://nssm.cc/download
2. Extract to `C:\nssm\`
3. Open Command Prompt as Administrator:

```
cmd
cd C:\nssm\win64
nssm install PostnatalStories C:\Production\postnatal-stories\start-service.bat
nssm set PostnatalStories DisplayName "Postnatal Stories Application"
nssm set PostnatalStories Description "Postnatal Stories web application"
nssm start PostnatalStories
```

8. Monitoring and Maintenance (Windows)

Set Up Task Scheduler for Monitoring

1. Open Task Scheduler
2. Create Basic Task
3. Name: "Check Postnatal Stories Health"
4. Trigger: Daily
5. Action: Start a program
6. Program: `powershell.exe`
7. Arguments: `File C:\Production\postnatal-stories\scripts\health-check.ps1`

Create health-check.ps1:

```
powershell
# Health check script
$response = Invoke-WebRequest -Uri "http://localhost:8000/health" -UseBasicParsing
if ($response.StatusCode -eq 200) {
    Write-Host "Application is healthy"
```

```
} else {
    Write-Host "Application health check failed"
# Restart services
    cd C:\Production\postnatal-stories
    docker-compose restart
}
```

Windows Event Log Monitoring

1. Open Event Viewer
2. Navigate to Windows Logs > Application
3. Filter for Docker and application events
4. Set up alerts for critical errors

Performance Monitoring

1. Open Performance Monitor (perfmon)
2. Add counters for:
    - CPU usage
    - Memory usage
    - Disk I/O
    - Network usage
3. Create baseline measurements for normal operation

9. Backup Strategy (Windows)

Automated Database Backup

Create backup-database.ps1:

```
powershell
# MongoDB backup script for Windows
$backupPath = "C:\Backups\postnatal-stories"
$date = Get-Date -Format "yyyyMMdd_HHmmss"
$backupDir = "$backupPath\$date"
```

```
# Create backup directory
New-Item -ItemType Directory -Path $backupDir -Force

# Backup database (requires MongoDB tools)
mongodump --uri="$env:MONGODB_URI" --out="$backupDir"

# Compress backup
Compress-Archive -Path $backupDir -DestinationPath "$backupDir.zip"
Remove-Item -Recurse -Path $backupDir

# Clean old backups (keep 30 days)
Get-ChildItem $backupPath -Filter "*.zip" | Where-Object {$_.CreationTime
-lt (Get-Date).AddDays(-30)} | Remove-Item
```

Schedule with Task Scheduler:

1. Create task to run backup-database.ps1 daily at 2 AM
2. Set to run with highest privileges
3. Configure email notifications on failure

10. Windows Security Considerations

Windows Defender Configuration

1. Add exclusions for Docker directories
2. Add exclusions for application log directories
3. Configure real-time protection settings

User Account Security

1. Create dedicated service account for running application
2. Grant minimum required permissions
3. Enable account lockout policies
4. Configure strong password policies

Network Security

1. Configure Windows Firewall with Advanced Security

2. Block unnecessary ports
3. Enable connection logging
4. Consider VPN access for administration

Configuration

Backend Configuration Details

JWT Configuration

```
python
# Strong secret key generation
import secrets
secret_key = secrets.token_urlsafe(32)
```

Email Configuration Options

Gmail Setup:

1. Enable 2-factor authentication
2. Generate app password
3. Use app password in SMTP_PASSWORD

SendGrid Setup:

1. Create SendGrid account
2. Verify sender email
3. Generate API key
4. Set EMAIL_PROVIDER=sendgrid

OpenAI API Setup

1. Create account at https://platform.openai.com
2. Generate API key

3. Add to OPENAI_API_KEY environment variable

Frontend Configuration

API Endpoint Configuration

```
javascript
// Development
const API_BASE_URL = window.location.port === '8080' ? '/api' : 'http://loc
alhost:8000';

// Production
const API_BASE_URL = '/api';
```

CORS Configuration

Ensure your backend ALLOWED_ORIGINS includes your frontend domain.

Testing the Installation

1. Backend Health Check

```
bash
# Test backend API
curl http://localhost:8000/health

# Expected response:
{
  "status": "healthy",
  "database_connected": true,
  "features_enabled": [
    "recovery_story_generation",
    "symptom_extraction",
    "database_storage",
```

```
    "authentication",
    "moderation"
  ]
}
```

2. Database Connection Test

```
bash
# Test database stats
curl http://localhost:8000/stats

# Expected response:
{
  "database_stats": {
    "total_stories": 0,
    "approved_stories": 0,
    "pending_stories": 0,
    "total_users": 0,
    "database_connected": true
  }
}
```

3. Frontend Test

1. Open browser to http://localhost:8080
2. Verify page loads with Postnatal Stories title
3. Test navigation links
4. Try creating an account
5. Test story sharing (requires account)

4. Integration Tests

Test User Registration

```
bash
curl -X POST http://localhost:8000/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "password": "testpass123",
    "display_name": "Test User",
    "age_verified": true,
    "agrees_to_terms": true
  }'
```

Test Story Submission

```
bash
# First login to get token
curl -X POST http://localhost:8000/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "password": "testpass123"
  }'

# Use token to submit story
curl -X POST http://localhost:8000/stories/submit \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_TOKEN_HERE" \
  -d '{
    "challenge": "Test challenge",
    "experience": "Test experience description",
    "solution": "Test solution that helped",
    "advice": "Test advice for others"
  }'
```

Support and Documentation

Getting Help

1. **Check logs** first for error messages
2. **Review this manual** for common solutions
3. **Search community forums** for similar issues
4. **Contact system administrator** for organization-specific issues

Additional Resources

- **MongoDB Documentation**: https://docs.mongodb.com
- **FastAPI Documentation**: https://fastapi.tiangolo.com
- **Docker Documentation**: https://docs.docker.com
- **OpenAI API Documentation**: https://platform.openai.com/docs

Version Information

- **Application Version**: 3.0.0
- **Python Requirements**: 3.11+
- **MongoDB Version**: 5.0+
- **Docker Version**: 20.10+

Last Updated: January 2025Installation Manual Version: 1.0