

# Requirements and Installation Guide

\*\*Project\*\*: School Activity Booking System

\*\*Platform\*\*: Windows, macOS, Linux

\*\*Python Version\*\*: 3.11+

---

## PART 1: SIMPLE INSTALLATION GUIDE

### **What You Need (Prerequisites)**

Think of this like ingredients for cooking:

\*\*Required Software (Must Have):\*\*

1. \*\*Python 3.11\*\* - The main programming language (like the oven)
2. \*\*Git\*\* - Version control (like a recipe book)

3. \*\*Code Editor\*\* - VSCode recommended (like your cooking utensils)

\*\*Optional (For Production):\*\*

4. \*\*PostgreSQL\*\* - Production database (like a professional kitchen)

5. \*\*Render Account\*\* - For deployment (like a restaurant for serving)

---

### **Step-by-Step Installation (Simple)**

\*\*Step 1: Get the Code\*\*

## **Open terminal/command prompt**

## **Clone the project (download it)**

```
git clone https://github.com/sanchitmahant/School-Activity-Booking-System.git
```

## **Enter the folder**

```
cd School-Activity-Booking-System
```

\*\*Step 2: Install Python Packages\*\*

## **Install all required libraries**

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

## **This installs:**

- Flask (web framework)**
- SQLAlchemy (database)**
- Flask-Mail (email)**
- ReportLab (PDF)**
- And 8 other packages**

**\*\*Step 3: Set Up Secrets\*\***

## **Copy the example file**

`copy .env.example .env`

## **Edit .env file and add:**

- Your Gmail address**
- Your Gmail app password (not regular password!)**
- A secret key**

**\*\*Step 4: Create Database\*\***

## **Run this command**

`python populate_db.py`

## **This creates:**

- 1 admin account**
- 1 parent account (demo)**
- 2 children**
- 6 tutors**
- 8 activities**

**\*\*Step 5: Run the Website\*\***

## **Start the server**

`python app.py`

## **Open browser and go to:**

**<http://localhost:5000>**

## **Login with:**

**Email: admin@greenwood.com**

**Password: admin123**

**\*\*That's it! Website is running! ■\*\***

---

## PART 2: TECHNICAL INSTALLATION

### ***System Requirements***

**\*\*Minimum:\*\***

- CPU: Dual-core 2.0 GHz
  - RAM: 4GB
  - Storage: 500MB free space
  - OS: Windows 10+, macOS 10.15+, Ubuntu 20.04+
- \*\*Recommended (Production):\*\***
- CPU: Quad-core 2.5 GHz+
  - RAM: 8GB+
  - Storage: 2GB free space SSD
  - Network: Stable internet for SMTP

---

### ***Prerequisites Installation***

**\*\*Python 3.11 Installation:\*\***

**\*\*Windows:\*\***

## Download from [python.org](https://python.org) or use winget

```
winget install Python.Python.3.11
```

### **Verify**

```
python --version # Should show Python 3.11.x
```

**\*\*macOS:\*\***

## Using Homebrew

```
brew install python@3.11
```

### **Verify**

```
python3.11 --version
```

**\*\*Linux (Ubuntu):\*\***

## Add deadsnakes PPA

```
sudo add-apt-repository ppa:deadsnakes/ppa
```

```
sudo apt update
```

```
sudo apt install python3.11 python3.11-venv python3.11-dev
```

### **Verify**

```
python3.11 --version
```

**\*\*Git Installation:\*\***

## Windows

```
winget install Git.Git
```

## macOS

```
brew install git
```

# Linux

```
sudo apt install git
```

## Verify

```
git --version
```

```
---
```

### ***Project Setup (Detailed)***

\*\*1. Clone Repository:\*\*

```
git clone https://github.com/sanchitmahant/School-Activity-Booking-System.git  
cd School-Activity-Booking-System
```

## Verify structure

```
ls -la
```

**Should see: app.py, config.py, requirements.txt, templates/, static/**

\*\*2. Create Virtual Environment:\*\*

\*\*Why virtual environment?\*\*

- Isolates project dependencies
- Prevents conflicts with system Python
- Reproducible environment

## Create venv

```
python -m venv venv
```

## Activate

### Windows:

```
venv\Scripts\activate
```

### macOS/Linux:

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

## Verify (should show venv in prompt)

```
which python # Should point to venv/bin/python
```

\*\*3. Install Dependencies:\*\*

## Upgrade pip first

```
python -m pip install --upgrade pip
```

## Install all requirements

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

## Verify installations

```
pip list
```

**Should show 12 packages:**

**Flask==2.3.0**

**Flask-Mail==0.9.1**

**Flask-WTF==1.1.1**

**Flask-SQLAlchemy==3.0.5**

**Werkzeug==2.3.0**

**ReportLab==4.0.4**

**python-dotenv==1.0.0**

**gunicorn==21.2.0**

**psycopg2-binary==2.9.9**

**email-validator==2.0.0**

**WTForms==3.0.1**

**SQLAlchemy==2.0.20**

\*\*requirements.txt Breakdown:\*\*

## **Core Framework**

Flask==2.3.0 # Web framework

Werkzeug==2.3.0 # WSGI utilities

## **Database**

Flask-SQLAlchemy==3.0.5 # ORM integration

SQLAlchemy==2.0.20 # ORM core

psycopg2-binary==2.9.9 # PostgreSQL adapter (production)

## **Forms & Security**

Flask-WTF==1.1.1 # CSRF protection

WTForms==3.0.1 # Form validation

## **Email**

Flask-Mail==0.9.1 # SMTP integration

email-validator==2.0.0 # Email validation

## **PDF Generation**

```
ReportLab==4.0.4 # PDF library
```

## Utilities

```
python-dotenv==1.0.0 # Environment variables
```

## Production Server

```
gunicorn==21.2.0 # WSGI HTTP server
```

```
---
```

### *Environment Configuration*

\*\*Create ` `.env` File:\*\*

## Copy template

```
cp .env.example .env
```

## Edit with your values

```
nano .env # or use any text editor
```

\*\*` `.env` Contents:\*\*

## Flask Configuration

```
SECRET_KEY=your-super-secret-key-change-this-in-production-32-chars-minimum
```

```
FLASK_ENV=development
```

## Database (Development - SQLite)

```
DATABASE_URL=sqlite:///school_booking.db
```

## Database (Production - PostgreSQL)

**DATABASE\_URL=postgresql://username:  
password@localhost:5432/school\_booking**

## Email Configuration (Gmail)

```
MAIL_SERVER=smtp.gmail.com
```

```
MAIL_PORT=587
```

```
MAIL_USE_TLS=True
```

```
MAIL_USERNAME=your-email@gmail.com
```

```
MAIL_PASSWORD=your-16-character-app-password
```

## Application Settings

```
MAIL_DEFAULT_SENDER=Greenwood International School
```

\*\*Gmail App Password Setup:\*\*

1. Go to Google Account → Security
2. Enable 2-Factor Authentication
3. Go to App Passwords
4. Generate password for "Mail"
5. Copy 16-character password (no spaces)
6. Paste in ` `.env` as ` MAIL\_PASSWORD`

\*\*Generate SECRET\_KEY:\*\*

## Run in Python terminal

```
import secrets  
print(secrets.token_hex(32))
```

**Output: 64-character hex string**

**Copy to .env as SECRET\_KEY**

--

### ***Database Initialization***

\*\*Development (SQLite):\*\*

## Create database and populate with sample data

```
python populate_db.py
```

**Output:**

**Creating database...**

**Adding admin...**

**Adding parent...**

**Adding children...**

**Adding tutors...**

**Adding activities...**

**Database populated successfully!**

**Verify database created**

```
ls *.db
```

**Should see: school\_booking.db**

\*\*Production (PostgreSQL):\*\*

**Install PostgreSQL**

**Ubuntu:**

```
sudo apt install postgresql postgresql-contrib
```

# macOS:

```
brew install postgresql
```

## Create database

```
sudo -u postgres psql
CREATE DATABASE school_booking;
CREATE USER school_admin WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE school_booking TO school_admin;
\q
```

## Update .env

```
DATABASE_URL=postgresql://school_admin:secure_password@localhost:5432/school_booking
```

## Run migrations (if using Alembic)

```
alembic upgrade head
```

## Or populate directly

```
python populate_db.py
---
```

### *Running the Application*

\*\*Development Server:\*\*

## Activate venv if not active

```
source venv/bin/activate # macOS/Linux
venv\Scripts\activate # Windows
```

## Run Flask development server

```
python app.py
```

## Output:

- \* Running on http://127.0.0.1:5000
- \* Restarting with stat
- \* Debugger is active!

## Access at: http://localhost:5000

\*\*Production Server (Gunicorn):\*\*

## Run with Gunicorn

```
gunicorn --workers 4 --threads 2 --bind 0.0.0.0:5000 app:app
```

## With logging

```
gunicorn --workers 4 --threads 2 \  
--access-logfile access.log \  
--error-logfile error.log \  
--bind 0.0.0.0:5000 \  
app:app
```

## Output:

```
[INFO] Starting gunicorn 21.2.0  
[INFO] Listening at: http://0.0.0.0:5000  
[INFO] Using worker: gthread  
[INFO] Booting worker with pid: 12345
```

---

### ***Default Login Credentials***

```
**Admin Portal:**  
URL: http://localhost:5000/admin/login  
Email: admin@greenwood.com  
Password: admin123  
**Parent Portal:**  
URL: http://localhost:5000/login  
Email: john.smith@email.com  
Password: password123  
**Tutor Portal:**  
URL: http://localhost:5000/tutor/login  
Email: sarah.jenkins@greenwood.com  
Password: tutor123  
■■ ■IMPORTANT■■: Change these passwords in production!
```

---

## PART 3: TROUBLESHOOTING

### ***Common Errors and Solutions***

```
**Error 1: `ModuleNotFoundError: No module named 'flask'`**  
**Cause**: Virtual environment not activated or dependencies not installed  
**Solution**:
```

## Activate venv

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

## Install dependencies

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

---

```
**Error 2: `SMTPAuthenticationError: Username and Password not accepted`**  
**Cause**: Gmail app password not configured correctly  
**Solution**:
```

1. Use App Password (not regular password)
  2. Enable 2FA on Google Account
  3. Generate new app password
  4. Update `\*.env` with 16-character password (no spaces)
- 

\*\*Error 3: `sqlalchemy.exc.OperationalError: no such table`\*\*

\*\*Cause\*\*: Database not initialized

\*\*Solution\*\*:

## Delete existing database

```
rm school_booking.db
```

## Recreate and populate

```
python populate_db.py
```

---

\*\*Error 4: `Address already in use`\*\*

\*\*Cause\*\*: Port 5000 already occupied

\*\*Solution\*\*:

## Find process using port 5000

### Windows:

```
netstat -ano | findstr :5000
```

### macOS/Linux:

```
lsof -i :5000
```

## Kill process

### Windows:

```
taskkill /PID /F
```

### macOS/Linux:

```
kill -9
```

## Or use different port

```
python app.py --port 8080
```

---

\*\*Error 5: `CSRF token missing or invalid`\*\*

\*\*Cause\*\*: CSRF protection enabled but token not included

\*\*Solution\*\*:

```
{{ csrf_token() }}  
fetch('/api/endpoint', {  
  headers: {  
    'X-CSRFToken': document.querySelector('meta[name="csrf-token"]').content  
  }  
});
```

---

## PART 4: DEPLOYMENT (PRODUCTION)

### *Render Deployment*

\*\*Prerequisites:\*\*

- GitHub repository
- Render account (free tier available)

\*\*Steps:\*\*

\*\*1. Prepare Repository:\*\*

### Ensure files present:

- Procfile**
- requirements.txt**
- runtime.txt (optional)**

\*\*Procfile:\*\*

```
web: gunicorn --workers 4 --threads 2 --timeout 120 --bind 0.0.0.0:$PORT app:app
```

\*\*runtime.txt:\*\*

```
python-3.11.6
```

\*\*2. Create PostgreSQL Database on Render:\*\*

- Go to Render Dashboard
- New → PostgreSQL
- Name: school-booking-db
- Copy Internal Database URL

\*\*3. Create Web Service:\*\*

- New → Web Service
- Connect GitHub repository
- Name: school-booking-system
- Environment: Python
- Build Command: `pip install -r requirements.txt`
- Start Command: Auto-detected from Procfile

\*\*4. Configure Environment Variables:\*\*

```
SECRET_KEY=
DATABASE_URL=
MAIL_SERVER=smtp.gmail.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USERNAME=
MAIL_PASSWORD=
FLASK_ENV=production
```

\*\*5. Deploy:\*\*

- Click "Create Web Service"
- Render automatically builds and deploys
- Access at: <https://school-booking-system.onrender.com>

\*\*6. Initialize Production Database:\*\*

### SSH into Render shell (or use one-off job)

```
python populate_db.py
```

```
---
```

### *Performance Recommendations*

**\*\*Development:\*\***

- Use SQLite (simple, file-based)
- Debug mode enabled
- Single worker process

**\*\*Production:\*\***

- Use PostgreSQL (concurrent, robust)
- Debug mode disabled
- Multiple workers (4+ for Gunicorn)
- Connection pooling enabled
- HTTPS enforced
- Static files served via CDN

---

## PART 5: VIVA Q&A;

**\*\*Q1: Why use virtual environment?\*\***

**\*\*Simple\*\*:** "Keeps project dependencies separate from system Python. Like having separate toolboxes for different projects - prevents mixing tools!"

**\*\*Technical\*\*:** "Virtual environments provide dependency isolation, preventing version conflicts. Each project has its own site-packages directory. Enables reproducible builds and clean uninstallation. Critical for teamwork and deployment consistency."

---

**\*\*Q2: Explain the difference between development and production configuration.\*\***

**\*\*Simple\*\*:** "Development = testing on your computer (relaxed security). Production = real users on internet (strict security, faster, no debug info shown)."

**\*\*Technical\*\*:**

## Development

```
DEBUG = True
SQLALCHEMY_ECHO = True # Log SQL
SQLALCHEMY_DATABASE_URI = 'sqlite:///dev.db'
SESSION_COOKIE_SECURE = False # Allow HTTP
```

## Production

```
DEBUG = False
SQLALCHEMY_ECHO = False
SQLALCHEMY_DATABASE_URI = 'postgresql://...'
SESSION_COOKIE_SECURE = True # HTTPS only
SQLALCHEMY_POOL_SIZE = 20 # Connection pooling
---
```

**\*\*Q3: Why Gunicorn for production instead of Flask dev server?\*\***

**\*\*Simple\*\*:** "Flask's built-in server is for testing only - slow and unsafe. Gunicorn is professional-grade - handles many users at once, secure, fast!"

**\*\*Technical\*\*:**

"Flask dev server is single-threaded with no process management. Gunicorn provides:

- Multiple worker processes (parallelism)
  - Thread support within workers
  - Graceful restart without downtime
  - Request timeouts
  - Load balancing across workers
  - Production-ready security
  - Handles 1000+ concurrent connections vs Flask dev's ~10"
-

# Conclusion

The installation process is streamlined for both development and production environments. Key considerations include proper dependency management, environment configuration, and database initialization. For production deployments, additional steps ensure security, performance, and reliability.

---

**\*\*Support:\*\***

- Issues: GitHub Issues
- Email: support@greenwoodschool.com
- Documentation: /Documentation folder

University of East London  
November 2025