HOSTELIX PRO

Developer Setup & Deployment Guide

Complete guide for new developers to set up, configure, build, and deploy the Hostelix Pro platform

Version 1.0 | February 2026

Flask Backend • Flutter Frontend • Cross-Platform

Table of Contents

1. Prerequisites

Ensure the following tools are installed on your machine before starting:

Required Software

Tool	Min Version	Purpose
Python	3.10+	Backend API server
pip	Latest	Python package manager
Flutter SDK	3.10+	Mobile/Desktop/Web frontend
Git	Any	Version control

Optional Software

Tool	Purpose	When Needed
Android Studio	Android emulator + SDK	Building Android APK
Xcode (macOS)	iOS build tools	Building iOS app
Visual Studio	C++ workload	Windows desktop builds
Chrome	Web browser	Flutter web debugging
Ngrok	Secure tunneling	Remote device access
PostgreSQL	Production database	Cloud deployment
Docker	Containerization	Docker deployment

Verify Your Environment

```
python --version          # Should show 3.10+
pip --version
flutter --version         # Should show 3.10+
flutter doctor            # Shows platform readiness
```

■ **IMPORTANT:** If **flutter doctor** shows issues, resolve them before proceeding. Run **flutter doctor -v** for detailed diagnostics.

2. Project Structure Overview

Directory	Contents	Description
backend/app/api/	11 modules	API route blueprints (auth, users, fees, etc.)
backend/app/models/	11 models	SQLAlchemy data models
backend/app/services/	9 services	Business logic layer
backend/scripts/	seed_db.py	Database seeding utilities
backend/migrations/	Alembic	Database migration history
hostelixpro/lib/pages/	22 pages	Flutter UI screens (6 directories by role)
hostelixpro/lib/services/	14 services	API clients and business logic
hostelixpro/lib/providers/	3 providers	State management (Provider pattern)
hostelixpro/lib/widgets/	9 directories	Reusable UI components
hostelixpro/assets/	images/	Images and icons
docs/	guides	FEATURES.md + this guide

3. Backend Setup (Flask API)

Step 1: Create Virtual Environment

```
cd project/backend
python -m venv .venv
```

Step 2: Activate Virtual Environment

Platform	Command
Windows (PowerShell)	.venv\Scripts\Activate.ps1
Windows (CMD)	.venv\Scripts\activate.bat
macOS / Linux	source .venv/bin/activate

■ **TIP:** You'll see **(.venv)** at the start of your terminal prompt when activated.

Step 3: Install Dependencies

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

This installs Flask, SQLAlchemy, Flask-Migrate, Flask-CORS, bcrypt, PyJWT, pyotp, reportlab, qrcode, openpyxl, cryptography, and all other required packages.

Step 4: Configure Environment Variables

```
copy .env.example .env     # Windows
cp .env.example .env       # macOS/Linux
```

Edit the **.env** file with your settings:

Variable	Default / Example	Purpose
FLASK_APP	app.py	Flask entry point
PORT	3000	Server port
SECRET_KEY	(generate random string)	Flask session secret
DATABASE_URL	sqlite:///hostelixpro.db	Database connection string
JWT_SECRET_KEY	(generate random string)	JWT signing key
CORS_ORIGINS	*	Allowed CORS origins (use * for dev)
MAIL_SERVER	smtp.gmail.com	SMTP server (optional for dev)
MAIL_PORT	587	SMTP port
MAIL_USERNAME	your-email@gmail.com	Email for sending OTPs
MAIL_PASSWORD	your-app-password	Gmail App Password

■ **IMPORTANT:** When SMTP is **not configured**, OTPs print to the terminal as:
[DEV MODE] OTP for user@email.com: 482917

Step 5: Initialize Database

```
python create_tables.py
```

Step 6: Start the Server

```
python app.py
```

Server runs at **http://0.0.0.0:3000**. Verify:

```
curl http://localhost:3000/api/v1/health
# Response: {"status": "healthy", "service": "Hostelix Pro API"}
```

4. Creating the Admin User

Method A: Seed Script (Recommended)

```
python scripts/seed_db.py
```

This creates a full set of test accounts:

Role	Email	Password	Details
Admin	admin@example.com	TestPass123	System Administrator
Teacher	teacher@example.com	TestPass123	Mr. Johnson
Routine Mgr	routine@example.com	TestPass123	Alice Brown
Student 1	student1@example.com	TestPass123	John Doe — Room A101
Student 2	student2@example.com	TestPass123	Jane Smith — Room A102
Student 3	student3@example.com	TestPass123	Mike Wilson — Room B201
Student 4	student4@example.com	TestPass123	Sarah Davis — Room B202
Student 5	student5@example.com	TestPass123	Tom Anderson — Room C301

■■ **WARNING:** Change all passwords before using in production!

Method B: Manual (Flask Shell)

```
python
>>> from app import create_app, db
>>> from app.models.user import User
>>> from app.services.auth_service import AuthService
>>> app = create_app()
>>> with app.app_context():
...     db.create_all()
...     admin = User(
...         email='admin@yourdomain.com',
...         password_hash=AuthService.hash_password('YourPassword'),
...         role='admin',
...         display_name='Admin Name'
...     )
...     db.session.add(admin)
...     db.session.commit()
```

5. Frontend Setup (Flutter App)

Step 1: Verify Flutter

```
flutter doctor
```

Platform	Requirements
Android	Android Studio + Android SDK + emulator or device
iOS	macOS + Xcode + CocoaPods
Web	Chrome browser
Windows	Visual Studio with 'Desktop development with C++' workload
Linux	clang, cmake, ninja-build, libgtk-3-dev
macOS	Xcode

Step 2: Install Dependencies

```
cd project/hostelixpro
flutter pub get
```

This installs all packages from **pubspec.yaml**: http, provider, go_router, shared_preferences, intl, path_provider, file_picker, permission_handler, flutter_form_builder, and more.

Step 3: Run in Development

```
flutter run                    # Default device
flutter run -d chrome          # Web
flutter run -d windows         # Windows desktop
flutter run -d      # Specific device
flutter devices                # List all devices
```

Step 4: Generate App Icon (Optional)

```
dart run flutter_launcher_icons
```

Generates platform icons from **assets/images/logo.png**.

6. Connecting Frontend to Backend

The Flutter app connects via **ApiClient** in **lib/services/api_client.dart**. Default: **http://127.0.0.1:3000/api/v1**

Platform	URL to Use	Notes
Chrome / Desktop	http://127.0.0.1:3000	Works out of the box
Android Emulator	http://10.0.2.2:3000	Emulator maps 10.0.2.2 → host localhost
iOS Simulator	http://127.0.0.1:3000	Works out of the box
Physical Device (same WiFi)	http://192.168.x.x:3000	Use computer's LAN IP
Remote / Different Network	Ngrok URL	See Section 8

Changing the URL

- **In-App (no code change):** Settings → Backend Configuration → Enter new URL → Save → Restart
- **In Code:** Edit **defaultHost** in **lib/services/api_client.dart**

■ **IMPORTANT:** Android Emulator cannot reach **127.0.0.1**. You **must** use **10.0.2.2** instead.

7. Building for Production

Platform	Build Command	Output Location
Android APK	flutter build apk --release	build/app/outputs/flutter-apk/app-release.apk
Split APKs	flutter build apk --split-per-abi	arm64, armeabi, x86_64 APKs
Play Store Bundle	flutter build appbundle --release	build/app/outputs/bundle/release/app-release.aab
iOS	flutter build ios --release	Open ios/Runner.xcworkspace in Xcode
Windows Desktop	flutter build windows --release	build/windows/x64/runner/Release/
macOS Desktop	flutter build macos --release	build/macos/Build/Products/Release/
Linux Desktop	flutter build linux --release	build/linux/x64/release/bundle/
Web App	flutter build web --release	build/web/

■ **TIP: Android:** The **arm64-v8a** APK is recommended for most modern phones.

■ **TIP: Web:** Serve with **python -m http.server 8080** from **build/web/**, or deploy to Netlify, Vercel, Firebase Hosting, etc.

■ **TIP: Windows Desktop:** Copy the entire **Release/** folder to distribute the app.

8. Remote Access with Ngrok

- **Step 1:** Install ngrok from **ngrok.com/download**
- **Step 2:** Sign up at **dashboard.ngrok.com** and get your auth token
- **Step 3:** Configure: **ngrok config add-authtoken YOUR_TOKEN**
- **Step 4:** Start backend: **python app.py** (Terminal 1)
- **Step 5:** Start tunnel: **ngrok http 3000** (Terminal 2)
- **Step 6:** Copy the **https://xxxx.ngrok-free.app** URL
- **Step 7:** In Flutter app → Settings → Backend Configuration → Paste URL

■ **TIP:** The **ngrok-skip-browser-warning** header is already added to all API requests — no extra configuration needed.

■■ **WARNING:** Free ngrok URLs change on restart. Consider a paid plan for a fixed subdomain.

9. Cloud Deployment

Docker Deployment

```
cd project/backend
docker build -t hostelixpro-api .
docker run -d -p 3000:3000 \
  -e DATABASE_URL=postgresql://... \
  -e SECRET_KEY=your-secret \
  -e JWT_SECRET_KEY=your-jwt \
  -e CORS_ORIGINS=* \
  hostelixpro-api
```

Platform-as-a-Service

Platform	Root Dir	Start Command	Database
Render	backend	gunicorn --bind 0.0.0.0:3000 --workers 4 app:app	Render PostgreSQL
Railway	backend	(auto-detected from Dockerfile)	Railway PostgreSQL
Fly.io	backend	(Dockerfile)	Fly PostgreSQL

Manual VPS Deployment

```
git clone
cd project/backend
python -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
cp .env.example .env && nano .env
python create_tables.py
python scripts/seed_db.py
```

```
gunicorn --bind 0.0.0.0:3000 --workers 4 app:app
```

■ **TIP:** For auto-restart on crashes, create a **systemd service** or use **supervisor**.

10. Email / SMTP Configuration

Development Mode (No Setup Needed)

When SMTP is not configured, OTPs print directly to the backend terminal:

```
[DEV MODE] OTP for admin@example.com: 482917
```

Production Mode (Gmail)

- 1. Enable **2-Step Verification** on your Google Account
- 2. Generate an **App Password** at myaccount.google.com/apppasswords
- 3. Add MAIL_SERVER, MAIL_PORT, MAIL_USERNAME, MAIL_PASSWORD to **.env**

Other SMTP Providers

Provider	Server	Port
Gmail	smtp.gmail.com	587
Outlook	smtp-mail.outlook.com	587
Yahoo	smtp.mail.yahoo.com	587
SendGrid	smtp.sendgrid.net	587
Mailgun	smtp.mailgun.org	587

11. API Reference Quick Start

Authentication Flow

```
# 1. Login (sends OTP)
curl -X POST http://localhost:3000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"email":"admin@example.com","password":"TestPass123"}'

# 2. Verify OTP (check terminal for code)
curl -X POST http://localhost:3000/api/v1/auth/verify-otp \
  -H "Content-Type: application/json" \
  -d '{"tx_id":"uuid-from-step1","otp":"123456"}'

# 3. Use JWT token for all requests
curl http://localhost:3000/api/v1/dashboard/stats \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"
```

All API Modules (61 Endpoints)

Module	Base Path	Endpoints
Auth	/api/v1/auth	7
Users	/api/v1/users	9
Account	/api/v1/account	6
Dashboard	/api/v1/dashboard	3
Reports	/api/v1/reports	5
Routines	/api/v1/routines	8
Fees	/api/v1/fees	10
Announcements	/api/v1/announcements	4
Notifications	/api/v1/notifications	4
Audit	/api/v1/audit	1
Backups	/api/v1/backups	4
Total		61

12. Troubleshooting

Backend Issues

Problem	Solution
ModuleNotFoundError	Run pip install -r requirements.txt
Port 3000 in use	Kill process or change PORT in .env
Database errors	Run python create_tables.py
CORS errors	Set CORS_ORIGINS=* in .env
OTP not received	Check terminal for [DEV MODE] OTP output
ImportError	Ensure virtual environment is activated

Flutter Issues

Problem	Solution
flutter pub get fails	flutter clean then flutter pub get
Can't connect to backend	Check Settings → Backend Configuration URL
Android build fails	Run flutter doctor --android-licenses
Web build blank page	Add --base-href=/ to build command
Windows build fails	Install Visual Studio C++ workload
Gradle error	Delete .gradle folder and rebuild

Connection Issues

Symptom	Solution
Connection refused	Is backend running? Is URL correct?
Android emulator fails	Use http://10.0.2.2:3000 not 127.0.0.1
Physical device fails	Use computer's LAN IP (192.168.x.x)
Ngrok tunnel not found	Restart: ngrok http 3000
401 Unauthorized	JWT token expired — log in again
403 Forbidden	User role lacks permission for endpoint

End of Document — Hostelix Pro Developer Setup & Deployment Guide v1.0