# Installation Guide

## Introduction

This guide will help you set up a Flask project using SQLAlchemy with pipenv for virtual environment management. We'll cover installation for different operating systems, setting up the virtual environment, seeding the database, and running the Flask server with debugging enabled.

## Prerequisites

- Python 3.8 or higher
- Access to a command-line interface
- Basic knowledge of Python and Flask

## Installation

### Step 1: Installing Pipenv

Pipenv is a popular tool for managing virtual environments and dependencies in Python projects. Here's how to install it on different operating systems:

#### 1.1 Linux (Ubuntu)

**# Make sure Python and pip are installed**

```
sudo apt update
sudo apt install python3 python3-pip
```

**# Install pipenv**

```
pip install --user pipenv
```

### 1.2 Windows

*Download and install Python from the official website. Make sure to add Python to your system's PATH during installation.*

*Open Command Prompt and run:*

**# Install pipenv**

*pip install pipenv*

### 1.3 MacOS

**# Install pipenv using Homebrew**

*brew install pipenv*

## Step 2: Setting Up the Virtual Environment

*After installing pipenv, navigate to your project directory and create the virtual environment:*

**# Move to the project directory**

*cd /path/to/your/flask/project*

**# Create and activate the virtual environment**

*pipenv install*

*pipenv shell*

## Step 3: Seeding the Database

Once you're inside the virtual environment, you can initialize and seed the database with sample data. Here's how:

### 3.1 Create the Database

**# Create all tables in the database**

*flask dbcreate*

### 3.2 Seed the Database

**# Populate the database with sample data**

*flask dbseed*

### 3.3 Drop the Database

If you need to reset the database, you can drop all tables:

*flask dbdrop*

## Step 4: Running the Flask Server

To run your Flask server, ensure you've set the environment variables and activated the virtual environment:

### 4.1 Enable Flask Debugging

To enable debugging in Flask, set the FLASK_DEBUG environment variable to true. This allows you to see detailed error messages and auto-reload the server on code changes. If you don't do this, you won't get helpful debugging information if something goes wrong also you need to run the server every single time you made the change by using "flask run" command.

**# Enable debugging**

**#Linux/Macos**

*export FLASK_DEBUG=true*

**#Windows**

*set FLASK_DEBUG=true*

## 4.2 Start the Flask Server

Once you've set up debugging, start the Flask server with:

```
flask run
```