Project Setup & Deployment Guide

This guide walks through setting up, configuring, and running the SlackBot project locally and making it accessible to Slack for testing via a public tunnel.

1) Clone the repository

- git clone https://github.com/AI-Intern2025/SlackBot.git
- cd SlackBot

2) Create and activate a Python virtual environment

- Windows (PowerShell):
  - py -3 -m venv .venv
  - ..venv\Scripts\activate
- macOS/Linux (bash/zsh):
  - python3 -m venv .venv
  - source .venv/bin/activate

Verify:

- python --version
- pip --version

3) Install dependencies

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

If any OS-specific build issues appear (e.g., psycopg2-binary or faiss-cpu), install system prerequisites or temporarily comment those lines until needed. Most workflows run fine with the listed dependencies.

4) Configure environment variables

Create a .env file in the project root with the following keys. Replace placeholder values with real credentials and connection strings.
Example .env:

- SLACK_BOT_TOKEN=xoxb-...
- SLACK_SIGNING_SECRET=xxxxxxxxxxxxxxx
- OPENAI_API_KEY=sk-...
- OPENAI_MODEL=gpt-4o
- OPENAI_TEMPERATURE=0.1
- OPENAI_MAX_TOKENS=2000
- OPENAI_TIMEOUT=60

- DATABASE_URL=postgresql+psycopg2://USER:PASSWORD@HOST:PORT/DB NAME
- SAMPLE_ROWS=2
- MAX_STRING_LENGTH=200
- AGENT_MAX_ITERATIONS=5
- AGENT_VERBOSE=false
- TESTING_MODE=true

Notes:

- Never commit .env to Git.
- DATABASE_URL must point to a reachable database with your tables.
- OPENAI_MODEL can be adjusted; defaults are already handled in code.

5) Create a Slack App

- Open https://api.slack.com/apps and click “Create New App”.
- App type: From scratch.
- Select the development workspace.

Then configure the following:

- OAuth & Permissions → Bot Token Scopes:
  - app_mentions:read
  - channels:history
  - channels:read
  - chat:write
  - chat:write.customize
  - commands
  - files:write
  - groups:history
  - groups:read
  - im:history
  - mpim:history
- Install App → Install to Workspace.
  - Copy the Bot User OAuth Token (starts with xoxb-) and put it into SLACK_BOT_TOKEN in .env.
- Basic Information → App Credentials:
  - Copy Signing Secret → put into SLACK_SIGNING_SECRET in .env.

6) Expose the local server (ngrok)

Slack must reach the local Flask/Bolt server via public HTTPS.

- Download and install ngrok (or use any HTTPS tunneling tool).
- Start a tunnel to the Flask server port (default 3000 unless changed):
  - ngrok http 3000

- Copy the generated https URL (e.g., https://abcd1234.ngrok-free.app).

Keep ngrok running while testing.

7) Configure Slack features with your public URL

Use the ngrok HTTPS URL in the following places. The project uses Slack Bolt with Flask and exposes:

- Events endpoint: /slack/events
- Interactivity endpoint: /slack/interactions
- Slash command endpoints: same base, handled by the app

In your Slack app settings:

- Event Subscriptions:
  - Enable Events: On.
  - Request URL: https://YOUR-NGROK-DOMAIN/slack/events
    - Slack should show “Verified”.
  - Subscribe to bot events:
    - app_mention
- Interactivity & Shortcuts:
  - Interactivity: On
  - Request URL: https://YOUR-NGROK-DOMAIN/slack/interactions
- Slash Commands:
  - /check-anomalies
    - Request URL: https://YOUR-NGROK-DOMAIN/slack/events
    - Short Description: Run automated checks for anomalies
    - Usage Hint: [metric] - Leave blank to run all metrics
  - /check-metric
    - Request URL: https://YOUR-NGROK-DOMAIN/slack/events
    - Short Description: Run anomaly checks for a specific business metric
    - Usage Hint: [metric] - e.g., daily_active_users
  - /create-alert
    - Request URL: https://YOUR-NGROK-DOMAIN/slack/interactions (if you plan to use interactive flows) or /slack/events if simple handling
    - Short Description: Create monitoring alerts from natural language
    - Usage Hint: [description with thresholds and priority]

Tip:

- If Slack can’t verify URLs, ensure ngrok is running, URL is HTTPS, and your app server is up and listening on the same port.

8) Start the application

From the project root (virtual environment active):

- python src/app.py

The server should start and listen on the configured port (commonly 3000). Keep both the app and ngrok windows running.

9) Test in Slack

- Mention the bot in a channel it's a member of:
    - @YourApp analyze daily deposits as a chart
- Try slash commands:
    - /check-anomalies
    - /check-metric daily_active_users
    - /create-alert Track payment errors above 50 critical realtime

The bot will:

- Analyze the request.
- Propose an SQL query and explanation.
- Ask for approval before executing.
- Return results and suggest charts for visualization when appropriate.

10) Troubleshooting

- Event URL not verifying:
    - Confirm Flask app is running and ngrok URL is correct.
    - Check logs for signature verification errors (SLACK_SIGNING_SECRET).
- Bot not responding:
    - Ensure the app is installed to the workspace.
    - Confirm required scopes were added before installation.
    - Check that the bot is a member of the channel.
- Database errors:
    - Verify DATABASE_URL connectivity and that expected tables exist.
- Dependency errors:
    - Upgrade pip (pip install --upgrade pip).
    - Reinstall failing package or comment it out temporarily if the related feature isn't needed for your test round.

11) Optional quality-of-life

- Create a .env.example with the same keys and dummy values for teammates.
- Add a proper .gitignore to exclude .venv, pycache, .env, and other artifacts.
- If using GitHub Actions or Docker later, document the environment variables and port mappings.

12) Daily monitoring (automated)

The project includes a FullyAutomatedMonitor that:

- Schedules checks based on BUSINESS_METRICS definitions.
- Sends real-time alerts to #alerts.
- Sends a daily digest at the configured time.

These are initialized in app.py when the app starts. Make sure the #alerts channel exists and the bot is a member.