

AP-Project — How to Run Guide

This document provides a comprehensive guide for compiling, configuring, and launching the IIIT DELHI ERP desktop application, specifically tailored for the **Windows development** environment using **PowerShell** and **PostgreSQL**.

1. Prerequisites and Environment Setup

Ensure the following components are installed and accessible via your system's PATH:

Requirement	Version	Verification
Java Development Kit (JDK)	17+ (Tested with JDK 24)	Run <code>java -version</code> and <code>javac -version</code> .
PostgreSQL Server	Any Stable	Server must be running locally or remotely.
PostgreSQL Client	<code>psql</code> command line tool	Run <code>psql --version</code> .
Working Directory	Project root	Contains <code>compile.bat</code> , <code>sql/</code> , and <code>config/</code> .

2. Quick Build and Execution (PowerShell)

The recommended workflow uses the provided batch file to ensure all resources are correctly placed before runtime.

1. **Navigate to Project Root:** Open PowerShell and change the directory to the project root:

```
cd C:\Users\YourUser\AP-Project
```

2. **Compile and Stage Resources:** Execute the custom `compile.bat` file. This script compiles source files into the `bin/` directory and copies essential runtime resources (`config/` files and `lib/*.jar`) into `bin/lib/`.

```
.\compile.bat
```

3. **Run Application:** Execute the compiled application, placing the `bin` directory (for classes) and the staged libraries (`bin/lib/*`) on the classpath.

```
java -cp "bin;bin/lib/*" edu.univ.erp.Main
```

VS Code Integration Notes

To run the application directly from the **VS Code Run button**:

- The `launch.json` configuration must set the **Current Working Directory (cwd)** to the project root to correctly locate resource files (like images and `config/app.properties`).
- The classpath must explicitly include `bin` and all JAR files in `lib/` (or staged files in `bin/lib/`).

3. Database Setup and Seeding

The application requires two PostgreSQL databases to enforce the security architecture.

1. **Create Databases:** Use the `psql` command to create the two required databases. (Adjust `-U postgres` and host/port as necessary).

```
# Create the Authentication DB and the ERP DB
psql -U postgres -c "CREATE DATABASE univ_auth;"
psql -U postgres -c "CREATE DATABASE univ_erp;"
```

2. Apply Schema and Seed Data: Load the table structures and initial test data into the respective databases.

```
# Load schema and test users into the Auth DB
psql -U postgres -d univ_auth -f sql/auth_schema.sql
psql -U postgres -d univ_auth -f sql/auth_seed.sql

# Load schema, courses, and sections into the ERP DB
psql -U postgres -d univ_erp -f sql/erp_schema.sql
psql -U postgres -d univ_erp -f sql/erp_seed.sql
```

4. Configuration and Credentials

4.1. Database Configuration (`config/app.properties`)

The application requires a configuration file at `config/app.properties` relative to the project root. Ensure the host, port, user, and password fields match your PostgreSQL setup.

```
# Authentication Database Configuration (Users, Roles, Hashes)
auth.url=jdbc:postgresql://localhost:5432/univ_auth
auth.user=postgres
auth.password=postgres

# ERP Database Configuration (Academic Data)
erp.url=jdbc:postgresql://localhost:5432/univ_erp
erp.user=postgres
erp.password=postgres
```

4.2. Default Test Credentials (Seeded)

The `auth_seed.sql` script inserts these default test accounts.

Role	Username	Password	Notes
Admin	admin1	admin123	Full control and maintenance access.

Instructor	inst5	inst1234	Assigned to sample sections.
Student 1	stu1	student1	Initial enrollment records exist.
Student 2	stu3	student3	Used for new enrollment tests.

5. Security & Credential Handling (Extra Information)

The security model is based on separating authentication concerns from academic data storage.

- **Credential Storage:** All passwords are stored in the `univ_auth` database as **bcrypt/argon2 hashes** (not plaintext).
 - **Hash Separation:** The `univ_erp` database, which stores student and course records, **contains no password or hash data**, significantly reducing the security impact if the academic database is compromised.
 - **Configuration Security:** For production use, storing credentials directly in `app.properties` is discouraged. A better practice would involve environment variables or a secure vault mechanism.
-

6. Troubleshooting Common Errors

Error Message	Cause	Resolution
<code>ClassNotFoundException: org.postgresql.Driver</code>	JDBC driver not on the classpath, or <code>lib/</code> was not copied to <code>bin/lib/</code> .	Re-run <code>\compile.bat</code> and ensure the <code>java -cp</code> command includes the driver JAR (<code>bin/lib/*</code>).

Could not read config/app.properties	Wrong current working directory (cwd).	Ensure you are running java from the project root directory or set cwd correctly in VS Code.
Connection refused. Check that host and port are correct.	PostgreSQL server is not running or credentials are wrong.	Check the status of your PostgreSQL service and verify the auth.url/erp.url and credentials in config/app.properties.
Background image not loaded	Resource path error.	Ensure the resource files (like images) are copied into the bin directory or are accessible relative to the runtime environment.