I ran pydeps on the codebase and the picture matches how I set up the layers. The app package is the entry point with Flask and config, and it keeps framework code at the edges while importing the routes so the rest of the project actually shows up in the graph. Those routes call the application layer where the real decisions and orchestration live, and that layer does not reach into Flask directly. The application layer then calls the data layer in src data, which owns reading and writing and does not leak database details back up. Inside data the modules build queries with the helpers in src sql sqlsafe so identifiers are validated with psycopg identifiers, values are always bound parameters, order by uses an allow list, and the limit is clamped to a safe range. The result is a top to bottom flow with no cycles, which keeps tests straightforward and makes refactors safer because dependencies only move in one direction. External libraries like Flask and psycopg appear only where they are needed, which is the separation of concerns I was aiming for