

Darwin Ai


Engineering Seniority Test rev. 1.1

Summary

The test involves the creation of a Telegram chatbot that facilitates the addition of expenses into a database. Users send quick short messages to the bot (e.g: "*Pizza 20 bucks*") and the bot handles the rest. This task is divided across two services:

- **Bot Service:** Developed in Python, this service is tasked with the analysis of incoming messages to identify and extract expense details before persisting these details into the database. It's the core component responsible for interpreting user inputs and converting them into structured data for storage.
- **Connector Service:** Built using Node.js, this service acts as the interface between the Telegram API and the Bot Service. It manages the reception of inbound messages from users, forwards these messages to the Bot Service for processing, and sends the appropriate responses back to the users via Telegram.

Requirements

- The bot should recognize a whitelist of Telegram users sourced from the database. Any user not listed in this database should be ignored by the bot.
- The bot should verify the content of received messages to distinguish expense-related inputs from non-expense texts, such as casual greetings or irrelevant information. Messages not pertaining to expenses should be ignored.
- The bot should automatically categorize expenses based on the content of the user's message. This categorization should classify expenses into predefined categories:
 - Housing, Transportation, Food, Utilities, Insurance, Medical/Healthcare, Savings, Debt, Education, Entertainment, and Other.
- The bot should reply "*[Category] expense added*  " upon successful addition of an expense.
- Setting up a new bot should not require any code changes. Avoid hard-coded values.

Tech requirements for Bot Service

- Python 3.11+ required.
- Must handle concurrent requests.
- Use LangChain with a [supported LLM](#).
- PostgreSQL database.

Tech requirements for Connector Service

- Node.js LTS required.
- Use ESM over CommonJS.
- TypeScript is optional.
- Other NPM dependencies are allowed.

Database DDL

```
CREATE TABLE users (  
  "id" SERIAL PRIMARY KEY,  
  "telegram_id" text UNIQUE NOT NULL  
);  
  
CREATE TABLE expenses (  
  "id" SERIAL PRIMARY KEY,  
  "user_id" integer NOT NULL REFERENCES users("id"),  
  "description" text NOT NULL,  
  "amount" money NOT NULL,  
  "category" text NOT NULL,  
  "added_at" timestamp NOT NULL  
);
```

Submission Guidelines

- Each service must include a clear README file outlining setup instructions, dependencies, and all required configuration steps.
- Your complete code must be hosted on GitHub. If your repository is private, invite [poislagarde](#) as a collaborator to provide access.
- Utilize AI tools (e.g., [ChatGPT](#), [Claude](#), [AI Studio](#), [Cursor](#), [Windsurf](#), etc.) freely to assist in development.
- Additional consideration will be given for application of best practices, thorough documentation (including inline comments), and code formatting.
- While not mandatory, you are encouraged to deploy your services to [Vercel](#), [Railway](#), or any similar PaaS for easier testing.
 - [Supabase](#) is a good option for PostgreSQL hosting with a free tier.

That's it!

If all that was easy enough for you and you want to show us more... do something unique to wow us.

Good luck!