joyagent-jdgenie项目分为三部分 java 、vue、python 分别对应genie-backend 、ui、genie-tool文件件，数据库在./genie-tool/autobots.db中

我要改造这个项目，先写一个项目的 prd：

1 改造https://github.com/jd-opensource/joyagent-jdgenie 尽量对修改关闭 对扩展开放，尽量减少对源码的入侵更改；

2 把界面加上类似chatgpt一样的聊天记录，可以查看历史，历史记录在./genie-tool/autobots.db数据库中，数据库结构和数据在./genie-tool/sql/main.sql ；

3 数据库中建一个表user\_request表，表列有 id（自增），username，request\_id

4 不同用户不同的网址参数，比如：访问[http://localhost:3000/api/username/admin](http://localhost:3000/api/user/admin) 就是username为admin的用户，不同用户历史记录是不同的，user\_request表中找到username对应的所有request\_id，再到fileinfo表中找到该用户的所有request\_id的历史记录，加载显示在页面上，类似chatgpt的页面。

joyagent 改造方案

1. 项目背景

joyagent项目由三部分组成：

Java → genie-backend （后端服务）

Vue → ui （前端界面）

Python → genie-tool （工具层，包含数据库）

数据库 → ./genie-tool/autobots.db（SQLite），结构定义在 ./genie-tool/sql/main.sql

目前项目缺少 ChatGPT 式的对话历史功能，用户在前端无法查看自己的历史交互内容。

2. 改造目标

对原有代码尽量少入侵修改：

以扩展为主，减少对核心源码的直接修改。

通过新增 API、Vue 组件和数据库表实现功能扩展。

增加类似 ChatGPT 的聊天历史功能：

用户可在前端 UI 中查看自己的历史聊天记录。

界面风格参考 ChatGPT 对话流布局。

多用户历史隔离：

不同 username 的历史记录互不干扰。

用户访问 http://localhost:3000/api/username/{username} 时，加载该用户的历史聊天记录。

3. 功能设计

3.1 用户请求与历史存储

新建数据表：user\_request

CREATE TABLE user\_request (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT NOT NULL,

request\_id TEXT NOT NULL

);

功能说明：

记录每个用户与 request\_id 的对应关系。

一个用户可有多个 request\_id。

与已有 fileinfo 表通过 request\_id 关联，获取详细历史。

3.2 历史记录展示逻辑

前端访问 URL：

http://localhost:3000/api/username/{username}

后端处理逻辑：

从 user\_request 表查询 username 对应的所有 request\_id

根据这些 request\_id，从 fileinfo 表获取用户的历史交互内容

将结果返回给前端

前端展示：

使用 Vue 新增 ChatGPT 风格的历史对话 UI

左侧显示历史会话列表（按时间排序）

右侧显示具体对话详情（类似 ChatGPT 界面）

4. 系统架构改造点

4.1 后端（Java / genie-backend）

新增 API：

@GetMapping("/api/username/{username}")

public List<HistoryRecord> getUserHistory(@PathVariable String username);

功能：

调用 autobots.db，查询 user\_request + fileinfo

组装用户的历史对话数据返回

4.2 前端（Vue / ui）

新增页面/组件：

ChatHistory.vue：显示用户历史记录

ChatView.vue：展示单个会话对话内容（类似 ChatGPT）

新增路由：

/username/:username

自动加载该用户的历史

4.3 工具层（Python / genie-tool）

负责初始化数据库、迁移：

在 main.sql 中增加 user\_request 表结构

提供迁移脚本更新已有数据库

5. 数据流

用户访问 http://localhost:3000/api/username/admin

后端查询：

user\_request 表获取所有 request\_id

fileinfo 表获取对应交互数据

返回 JSON：

{

"username": "admin",

"history": [

{"request\_id": "req1", "content": "..."},

{"request\_id": "req2", "content": "..."}

]

}

前端渲染：

左侧：会话索引

右侧：对话详情

6. 非功能性需求

扩展优先：避免直接改动核心业务逻辑

性能：查询时加索引（username、request\_id）保证多用户并发效率

兼容性：保证原有功能可正常使用