Project Proposal – AI Chat Desktop Application

1. Project Overview

This project is a Java-based desktop application that allows users to chat with an AI assistant, similar to ChatGPT, but with local conversation management and persistent history. Each user can log in, create multiple chat threads, and view all past conversations. AI responses are generated using the **Google Gemini API**, and all user data is stored in a PostgreSQL database.

2. Core Features

1. **User Authentication**
   - Users can sign up and log in using username + password
   - Passwords stored securely using hashing (BCrypt)
2. **AI Chat Interface**
   - Users can send messages and receive AI responses
   - AI call handled in a **background thread** so UI stays responsive
   - Uses **Gemini API** to generate replies
3. **Multi-Conversation Support**
   - Users can create, rename, and delete separate chat threads
   - Switching chats loads message history from database

4. **Persistent Data Storage**
    - All users, conversations, and messages stored in PostgreSQL
    - Accessed through Spring Boot backend using JPA
5. **Desktop GUI**
    - Built with Java Swing
    - Left sidebar: list of chats
    - Right panel: message history + input box

---

3. Java Concepts Covered (Advanced Topics Requirement)

Java Topic	How It Is Used
Java GUI (Swing)	User interface for login, chat window, and conversation list
Databases & JDBC	PostgreSQL accessed via Spring Data JPA
Spring Framework	REST backend for auth, conversation management, and AI requests
Multithreading	Background task for AI response so UI is not blocked
Networking	Swing client communicates with Spring server via HTTP; server calls Gemini API
Security	Password hashing (BCrypt); optional JWT for session handling

---

4. High-Level Architecture

```
+ Server also calls → Gemini API for AI responses
```

---

5. Development Plan

1. Set up Spring Boot backend + PostgreSQL connection
2. Implement user auth (sign up, login, hashed passwords)
3. Implement conversation/message entities + API endpoints
4. Build Swing UI (login → chat window → sidebar list)
5. Integrate Gemini API and async background reply handling
6. Final polishing of User Workflows.

---

6. Advanced Topics Covered (How Each Requirement Is Satisfied)

- **GUI (Java Swing)** – Desktop client with login screen, chat window, and conversation sidebar
- **JDBC / Database** – PostgreSQL used to store users, conversations, and messages via Spring Data JPA (built on JDBC)
- **Spring Framework** – Backend built using Spring Boot (Spring Web + Spring Data), exposes REST API to the Swing client
- **Multithreading** – AI response requests run in a background thread so the UI remains responsive
- **Networking** – Swing client communicates with Spring server using HTTP; server communicates with Gemini API over HTTPS
- **Security** – User passwords hashed using BCrypt; optional JWT token issued after login for authenticated requests