CAPSTONE PROJECT

CHILL SPACE: REAL-TIME CHAT APPLICATION

BY,

TAMIZHARASAN R

JAVA FULL STACK – (FACEPREP)

43731195

B.E CSE – AI

DRIVE LINK (for project demo) :

“ **CLICK HERE** ”

1. INTRODUCTION

Chill Space is a comprehensive, production-ready real-time chat application built with Spring Boot 3.2.3 and modern enterprise technologies. This placement project demonstrates advanced full-stack development practices including secure JWT authentication, real-time WebSocket communication, AI integration with Google Gemini, and cloud-ready deployment architecture. The backend showcases clean architecture principles, role-based access control (RBAC), RESTful API design, and scalable system design suitable for enterprise-level applications.

In addition, the system is designed with modular components to ensure maintainability and ease of future enhancements. Robust database modeling and efficient data access layers support reliable data persistence and performance. Overall, Chill Space reflects industry-standard practices and serves as a strong demonstration of practical backend engineering skills.

2. ABSTRACT

Chill Space is a real-time chat application backend developed using Spring Boot 3.2.3, Java 17, and MySQL 8.0. The system implements secure JWT-based authentication with role-based access control (Admin, Moderator, User) and real-time messaging via WebSocket and STOMP protocol. Key features include persistent chat history, file sharing with BLOB storage, profile management with avatar support, and an AI assistant powered by Google Gemini with function calling capabilities.

The backend follows a clean layered architecture (Controller-Service-Repository pattern) ensuring scalability and maintainability. Additional features include admin analytics, user ban/unban functionality, email verification through Supabase, and cloud-ready deployment using Docker and Aiven Cloud for MySQL hosting.

3. TARGET USERS

- **End Users**: People looking for a real-time messaging platform with AI assistance
- **Administrators**: System admins managing users, content moderation, and platform analytics
- **Developers**: Integration partners who want to utilize the REST APIs
- **Recruiters**: Evaluating the technical implementation and architecture as a portfolio project

4. TECHNOLOGY STACK

Category	Technology	Version	Purpose
Backend	Java	17	Core language
	Spring Boot	3.2.3	Framework
	Spring Security	Latest	Authentication
Database	MySQL	8.0+	Relational DB
	Hibernate/JPA	Latest	ORM
Security	JWT (JJWT)	0.11.5	Token auth
	BCrypt	Built-in	Password hashing
Real-Time	Spring WebSocket	Latest	Bidirectional communication
	STOMP	-	Message protocol
AI	Google Gemini API	Latest	AI assistance
Email	Supabase	Latest	Verification
Deployment	Docker	20.10+	Containerization
	Aiven Cloud	-	MySQL hosting
Build	Maven	3.8+	Build automation

Table 1 : Tech Stack

5. REQUIREMENTS

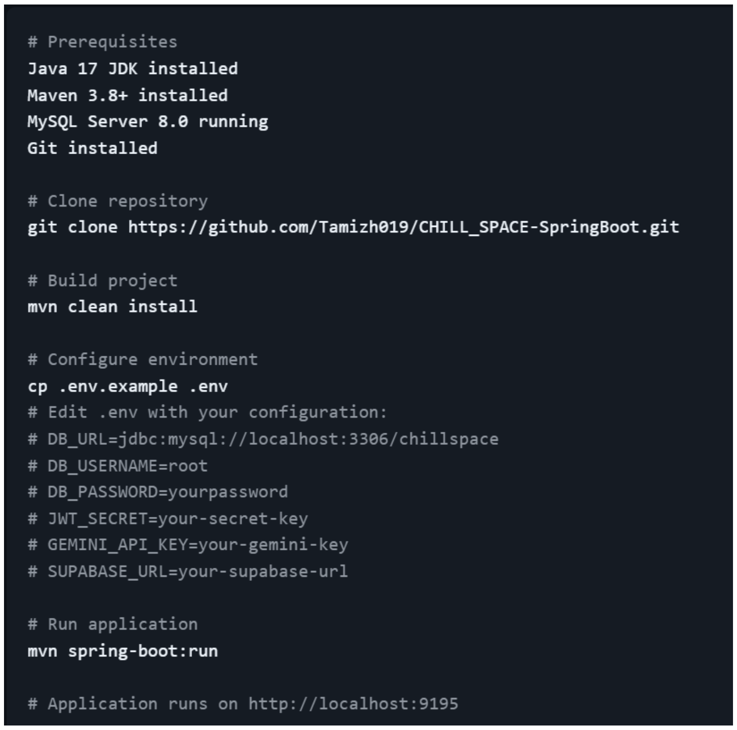
5.1 Hardware

- **RAM**: Minimum 2GB, Recommended 8GB
- **CPU**: 2+ cores
- **Storage**: 5GB+
- **Bandwidth**: 10 Mbps+

5.2 Software

- Java 17 JDK
- Maven 3.8+
- MySQL Server 8.0+
- Docker 20.10+ (for containerization)
- Git 2.30+

5.3 Development Environment Setup



```
# Prerequisites
Java 17 JDK installed
Maven 3.8+ installed
MySQL Server 8.0 running
Git installed

# Clone repository
git clone https://github.com/Tamizh019/CHILL_SPACE-SpringBoot.git

# Build project
mvn clean install

# Configure environment
cp .env.example .env
# Edit .env with your configuration:
# DB_URL=jdbc:mysql://localhost:3306/chillspace
# DB_USERNAME=root
# DB_PASSWORD=yourpassword
# JWT_SECRET=your-secret-key
# GEMINI_API_KEY=your-gemini-key
# SUPABASE_URL=your-supabase-url

# Run application
mvn spring-boot:run

# Application runs on http://localhost:9195
```

Figure 1 : Environment Setup

6. SYSTEM ARCHITECTURE

The system architecture of the Chill Space real-time chat application follows a well-structured layered design to ensure clear separation of responsibilities, scalability, and ease of maintenance. The overall flow of the application, from interaction to data persistence, is illustrated in Figure 1 on the following page, highlighting how each layer communicates with the next to deliver secure and real-time functionality.

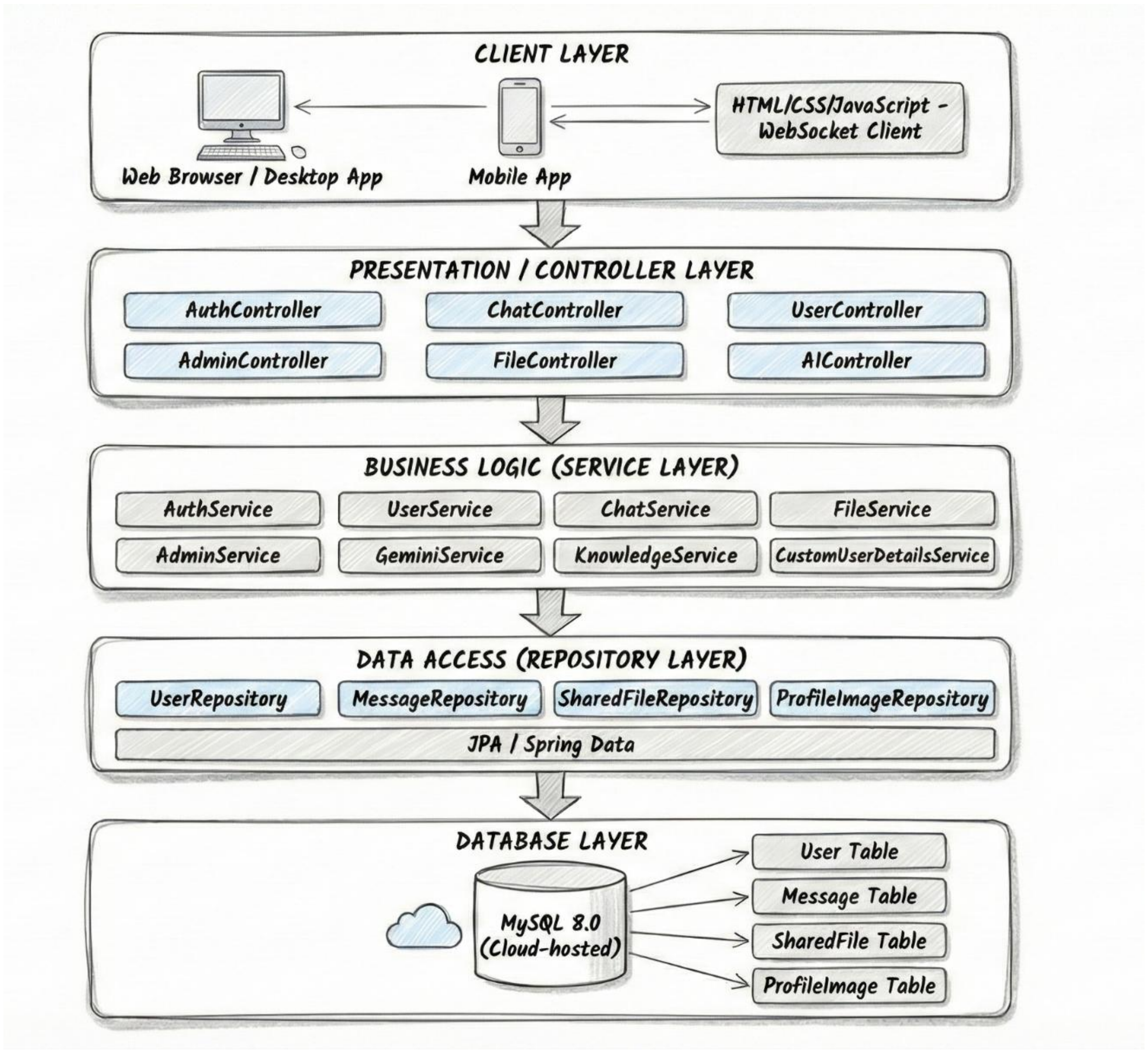


Figure 2 : System Architecture

The real-time communication in the Chill Space application is achieved using Spring Boot WebSocket to enable instant message delivery. When a client sends a message through the /app/chat.sendMessage endpoint, it is received by the MessageHandler controller and processed in the service layer, where necessary business logic is applied and the message is stored in the database. The processed message is then broadcast via the /topic/public channel, allowing all connected clients to receive the message instantly with low latency.

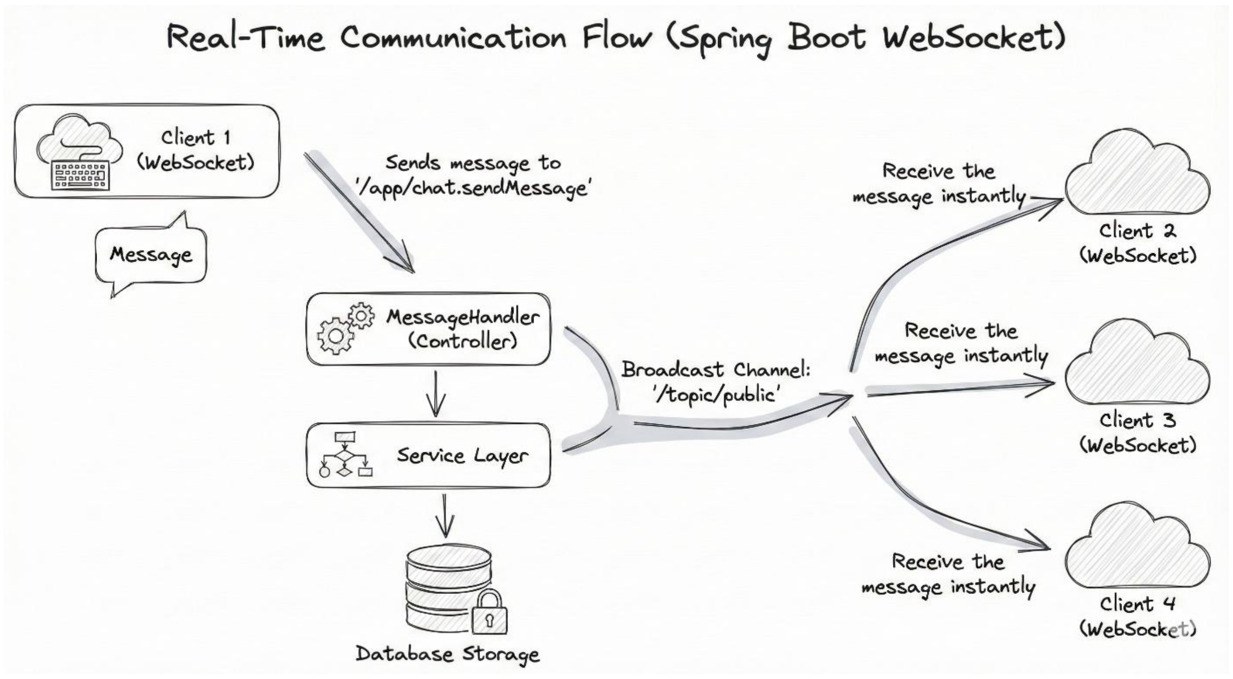
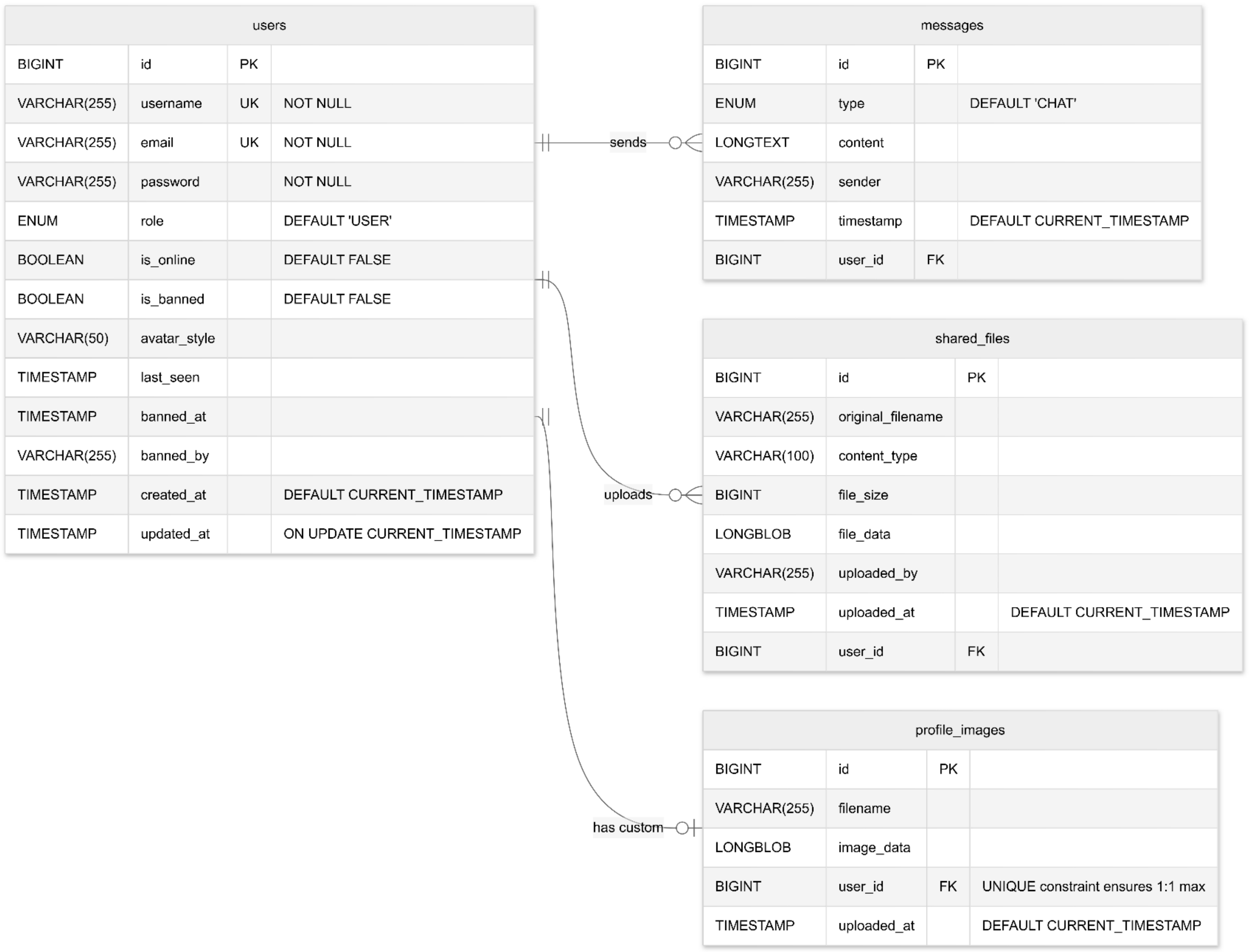

Real-Time Communication Flow (Spring Boot WebSocket)
Client 1 (WebSocket)
Message
Sends message to '/app/chat.sendMessage'
MessageHandler (Controller)
Service Layer
Database Storage
Broadcast Channel: '/topic/public'
Receive the message instantly
Client 2 (WebSocket)
Receive the message instantly
Client 3 (WebSocket)
Receive the message instantly
Client 4 (WebSocket)

Figure 3 : Communication Flow

Key Highlights:

- WebSocket enables bidirectional, low-latency communication.
- Messages are processed through controller and service layers.
- Broadcast mechanism ensures instant delivery to all connected clients

7. Entity-Relationship Diagram



users			
BIGINT	id	PK	
VARCHAR(255)	username	UK	NOT NULL
VARCHAR(255)	email	UK	NOT NULL
VARCHAR(255)	password		NOT NULL
ENUM	role		DEFAULT 'USER'
BOOLEAN	is_online		DEFAULT FALSE
BOOLEAN	is_banned		DEFAULT FALSE
VARCHAR(50)	avatar_style		
TIMESTAMP	last_seen		
TIMESTAMP	banned_at		
VARCHAR(255)	banned_by		
TIMESTAMP	created_at		DEFAULT CURRENT_TIMESTAMP
TIMESTAMP	updated_at		ON UPDATE CURRENT_TIMESTAMP

messages			
BIGINT	id	PK	
ENUM	type		DEFAULT 'CHAT'
LONGTEXT	content		
VARCHAR(255)	sender		
TIMESTAMP	timestamp		DEFAULT CURRENT_TIMESTAMP
BIGINT	user_id	FK	

shared_files			
BIGINT	id	PK	
VARCHAR(255)	original_filename		
VARCHAR(100)	content_type		
BIGINT	file_size		
LONGBLOB	file_data		
VARCHAR(255)	uploaded_by		
TIMESTAMP	uploaded_at		DEFAULT CURRENT_TIMESTAMP
BIGINT	user_id	FK	

profile_images			
BIGINT	id	PK	
VARCHAR(255)	filename		
LONGBLOB	image_data		
BIGINT	user_id	FK	UNIQUE constraint ensures 1:1 max
TIMESTAMP	uploaded_at		DEFAULT CURRENT_TIMESTAMP

Figure 4 : Database Schema (ER Diagram)

Figure 3 presents the database schema of the Chill Space application, illustrating the core entities, their attributes, and the relationships between users, messages, shared files, and profile images. The diagram highlights one-to-many relationships between users and messages, and users and shared files, as well as a one-to-one relationship between users and profile images.

8. HTTP REQUEST METHODS

HTTP request methods define the actions that a client can perform on server resources.They specify how data should be retrieved, created, updated, or deleted.Each method follows a standardized rule to ensure consistent communication between client and server.

In the Chill Space backend, these methods enable structured and secure RESTful API interactions.

HTTP Method	Endpoint	Description
GET	http://localhost:9195/api/users/	Return the list of all users
GET	http://localhost:9195/api/users/5	Return the user of id **5**
POST	http://localhost:9195/api/auth/register	Create a new user account
POST	http://localhost:9195/api/auth/login	Authenticate user and return JWT
PUT	http://localhost:9195/api/users/profile	Update user profile
PUT	http://localhost:9195/api/admin/ban/12	Ban user of id **12** (Admin only)
DELETE	http://localhost:9195/api/chat/45	Delete message of id **45**
DELETE	http://localhost:9195/api/file/8	Delete file of id **8**

Table 2 : HTTP REQUEST METHODS

WebSocket METHODS :

WebSocket Destination	**Endpoint**	**Description**
SEND	/app/chat.sendMessage	Send a chat message to all users
SUBSCRIBE	/topic/public	Subscribe to receive chat messages
SEND	/app/chat.addUser	Notify when user joins chat

Table 3 : WebSocket Methods

9. MODULES

Module 1: User Management (users table)

This module handles user account management and authentication. It stores essential user information including username, email, encrypted passwords using BCrypt hashing, and user roles (ADMIN, MODERATOR, USER). The table tracks user status such as online/offline state .It maintains timestamps for account creation and updates .

Key Fields:

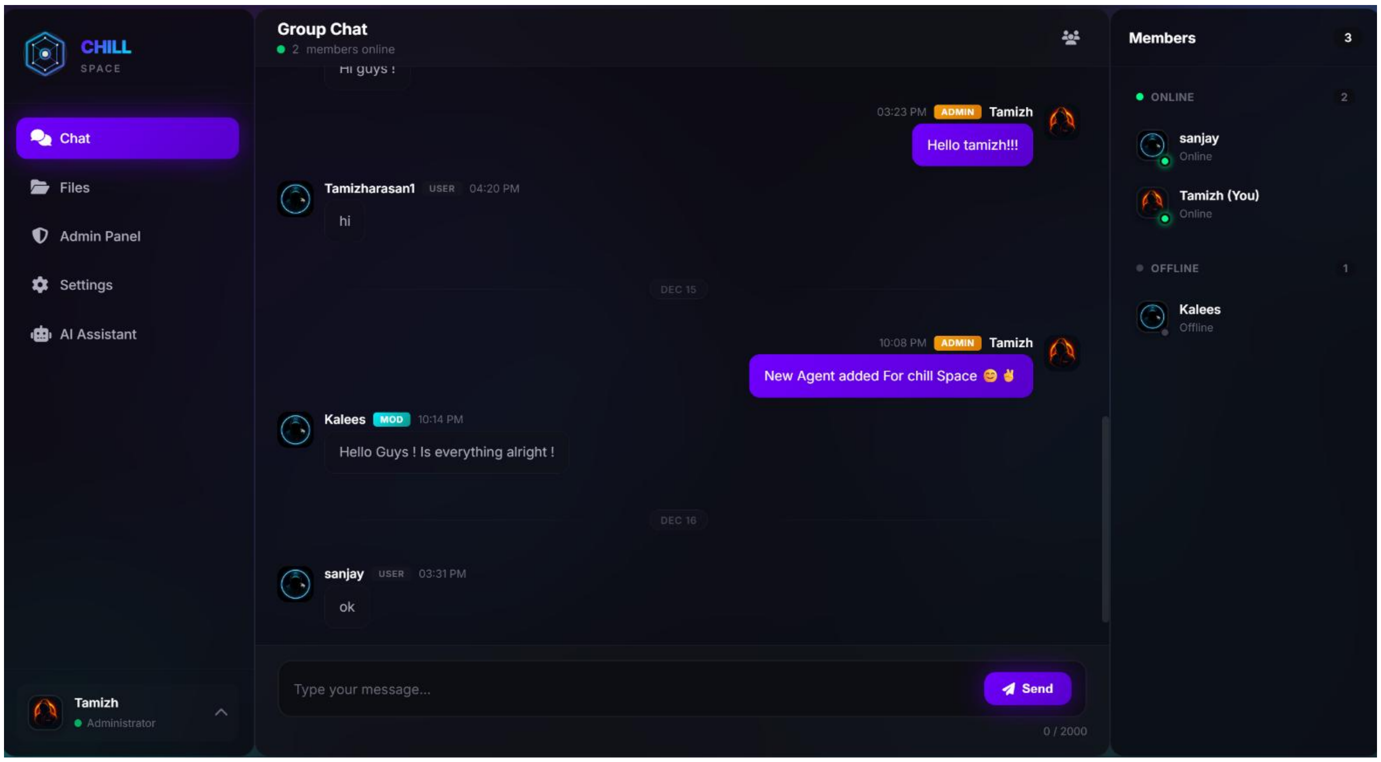
- id - Primary key
- username, email - Unique identifiers
- password - BCrypt hashed
- role - ADMIN, MODERATOR, or USER
- is_online, is_banned - User status
- created_at, updated_at - Timestamps

Module 2: Chat Management (messages table)

This module manages all chat communications in the system. It stores message content, sender information, and message types including regular chat messages, join/leave notifications, and system messages. Each message is linked to a user through a foreign key relationship and timestamped for chronological ordering. This enables real-time chat functionality and message history retrieval.

Key Fields:

- id - Primary key
- type - CHAT, JOIN, LEAVE, SYSTEM
- content - Message text
- sender - Username
- timestamp - When message was sent



Hi guys !

03:23 PM ADMIN Tamizh
Hello tamizh!!!

Tamizharasan1 USER 04:20 PM
hi

DEC 15

10:08 PM ADMIN Tamizh
New Agent added For chill Space 😊 ✌

Kalees MOD 10:14 PM
Hello Guys ! Is everything alright !

DEC 16

sanjay USER 03:31 PM
ok

Figure 5 : User Interface Of CHATS

Module 3: File Management (shared_files table)

This module handles file sharing capabilities within the application. It stores uploaded files as binary data (BLOB) along with metadata such as original filename, content type (MIME type), and file size. Each file is associated with the uploader through a user ID reference, allowing tracking of file ownership and enabling secure file download and deletion operations.

Key Fields:

- id - Primary key
- original_filename - File name
- content_type - File type (PDF, image, etc.)
- file_size - Size in bytes
- file_data - Binary file content (BLOB)

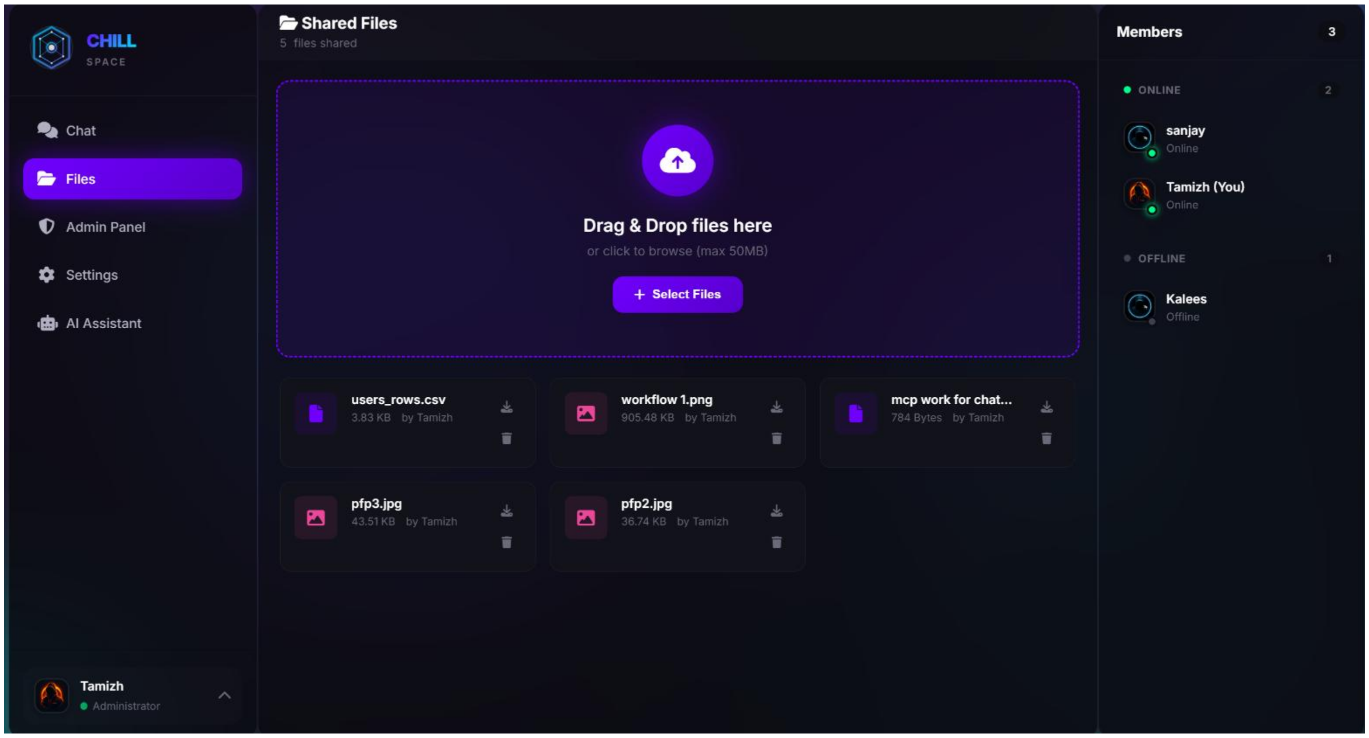


Figure 6 : File Management

Module 4: Profile Management (profile_images table)

This module manages user profile pictures and avatars. It stores custom avatar images as binary data with a unique one-to-one relationship with users, ensuring each user can have only one profile image at a time. The module supports avatar upload functionality and replaces default system-generated avatars with user-uploaded images.

Key Fields:

- id - Primary key
- filename - Image name
- image_data - Binary image (BLOB)
- user_id - Foreign key to users (UNIQUE - one avatar per user)
- uploaded_at - Upload timestamp

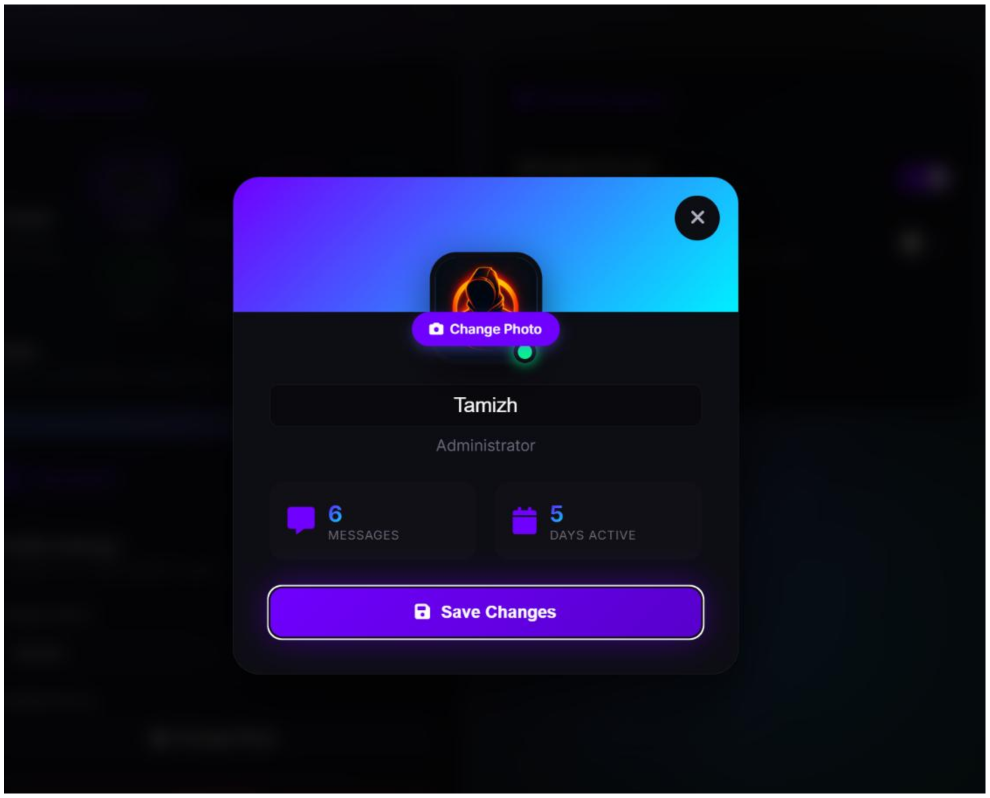


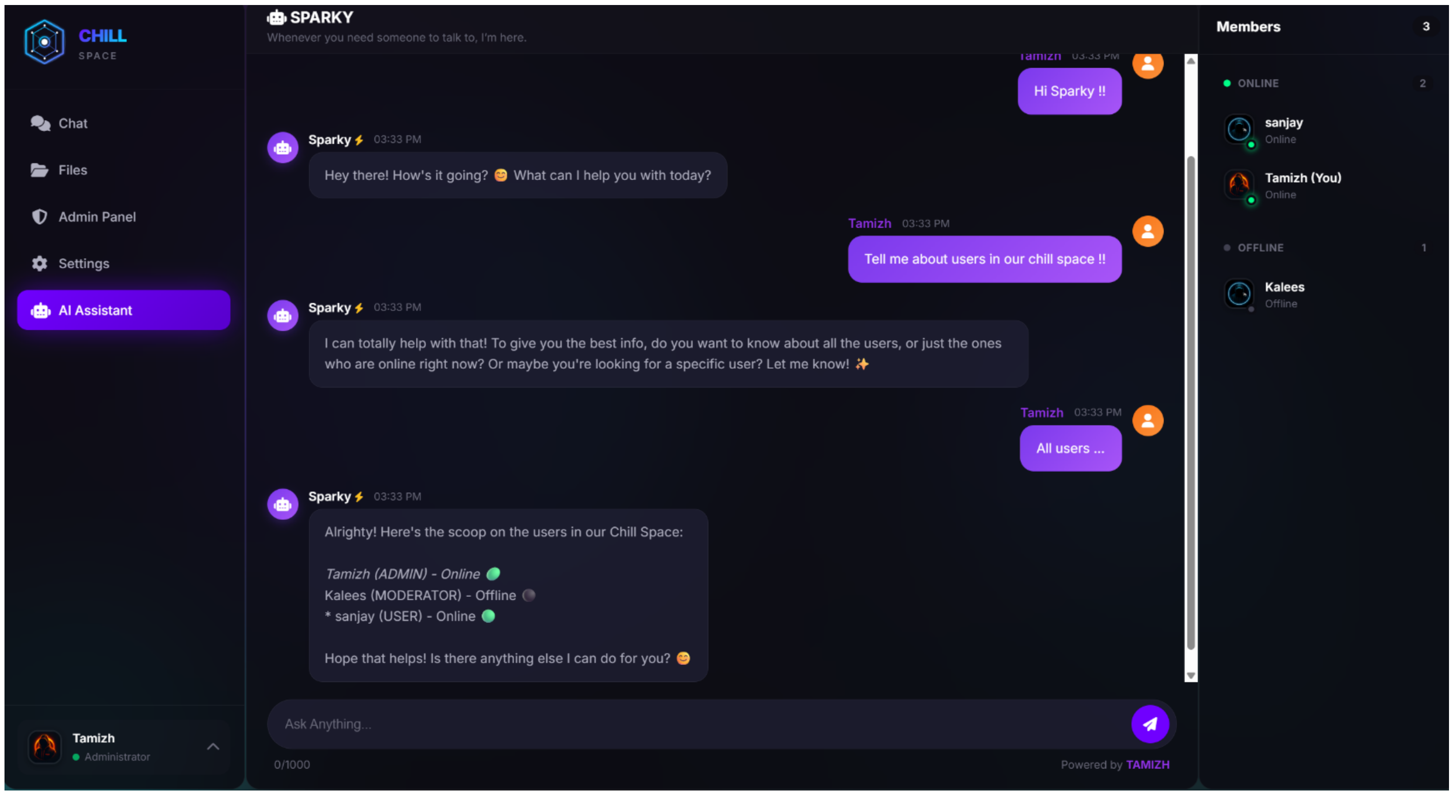
Figure 7 : Profile Management

10. SPARKY – Gemini AI–Powered Assistant

SPARKY is a friendly, context-aware AI assistant powered by Gemini AI, designed to support natural multi-turn conversations by remembering chat history.

Features:

- **Natural Language Processing**: Understand user queries
- **Function Calling (MCP)**: Invoke backend functions like:
    - get_users() - Retrieve user list
    - get_messages() - Fetch chat history
    - get_files() - List shared files
- **Knowledge Base**: PDF RAG for contextual responses.



Tamizh 03:33 PM
Hi Sparky !!

Sparky ⚡ 03:33 PM
Hey there! How's it going? 😊 What can I help you with today?

Tamizh 03:33 PM
Tell me about users in our chill space !!

Sparky ⚡ 03:33 PM
I can totally help with that! To give you the best info, do you want to know about all the users, or just the ones who are online right now? Or maybe you're looking for a specific user? Let me know! ✨

Tamizh 03:33 PM
All users ...

Sparky ⚡ 03:33 PM
Alrighty! Here's the scoop on the users in our Chill Space:

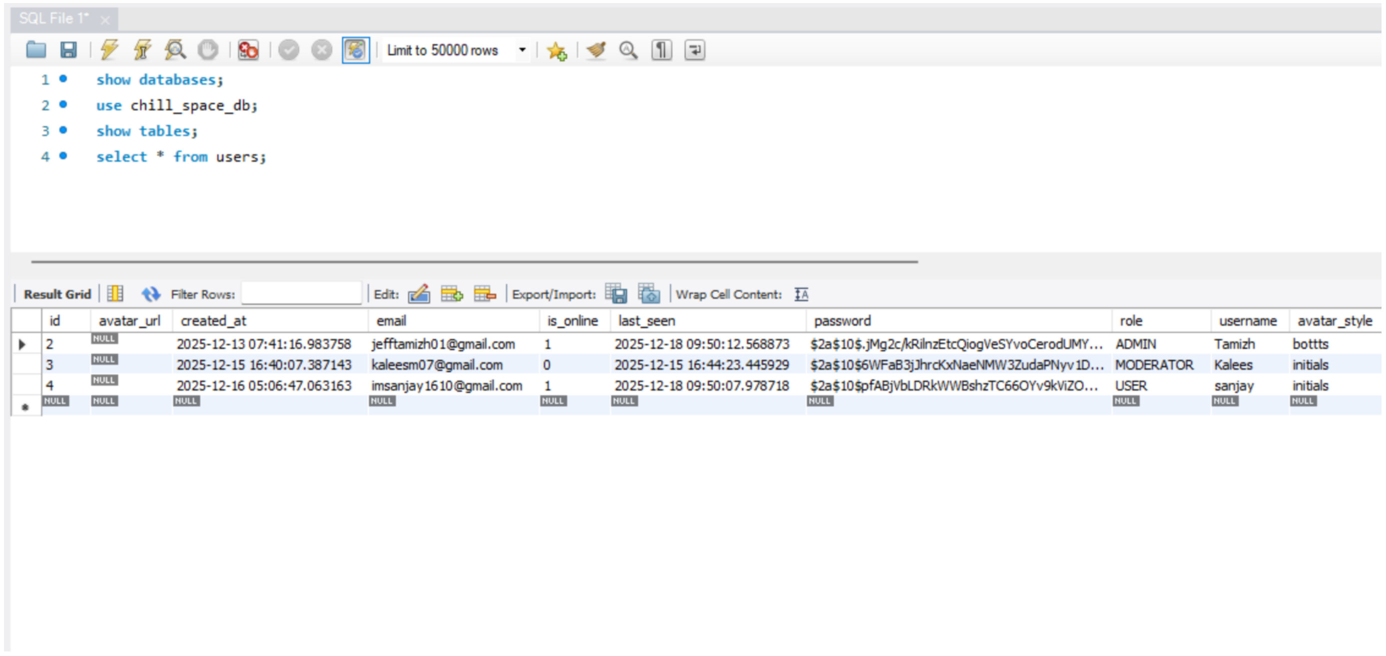
Tamizh (ADMIN) - Online 🟢
Kalees (MODERATOR) - Offline ⚫
* sanjay (USER) - Online 🟢

Hope that helps! Is there anything else I can do for you? 😊

Figure 7 : Sparky User Interface (Friendly AI)

PROJECT OUTPUT – DATABASE SCHEMA

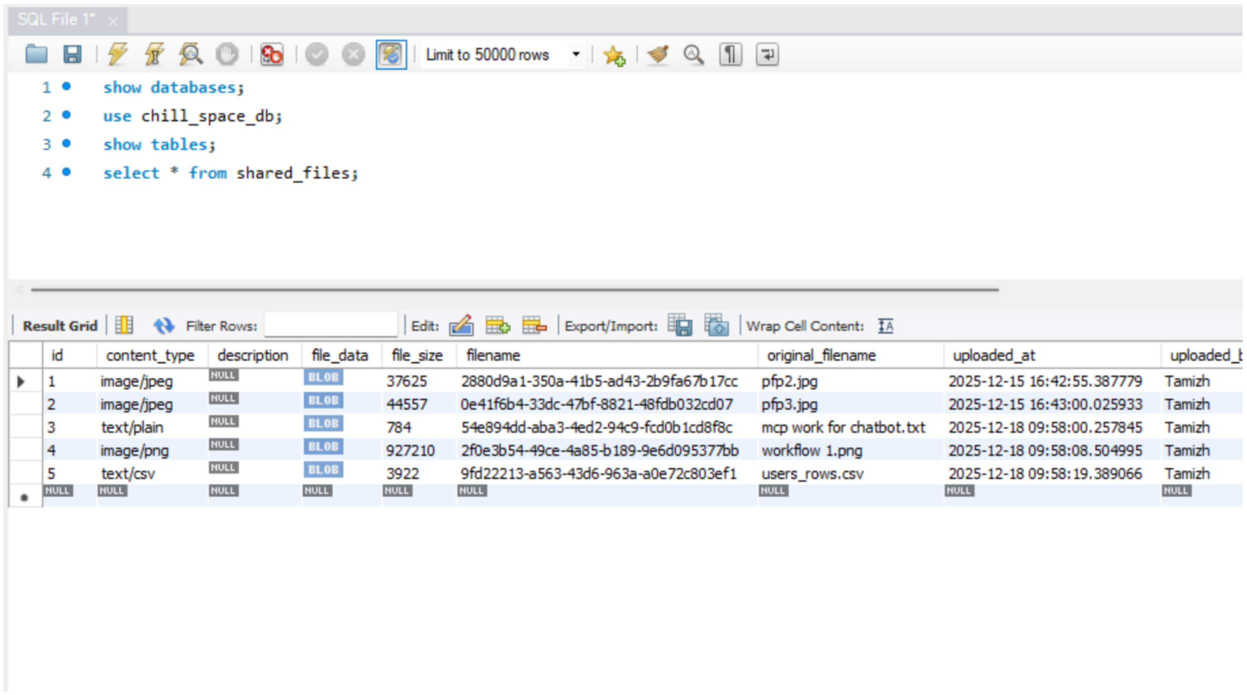
1) users TABLE :



```
show databases;
use chill_space_db;
show tables;
select * from users;
```

	id	avatar_url	created_at	email	is_online	last_seen	password	role	username	avatar_style
▶	2	NULL	2025-12-13 07:41:16.983758	jefftamizh01@gmail.com	1	2025-12-18 09:50:12.568873	$2a$10$.jMg2c/kRilnzEtcQiogVeSYvoCerodUMY...	ADMIN	Tamizh	bottts
	3	NULL	2025-12-15 16:40:07.387143	kaleesm07@gmail.com	0	2025-12-15 16:44:23.445929	$2a$10$6WFaB3jJhrcKxNaeNMW3ZudaPNyv1D...	MODERATOR	Kalees	initials
	4	NULL	2025-12-16 05:06:47.063163	imsanjay1610@gmail.com	1	2025-12-18 09:50:07.978718	$2a$10$pfABjVbLDRkWWBshzTC66OYv9kViZO...	USER	sanjay	initials
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

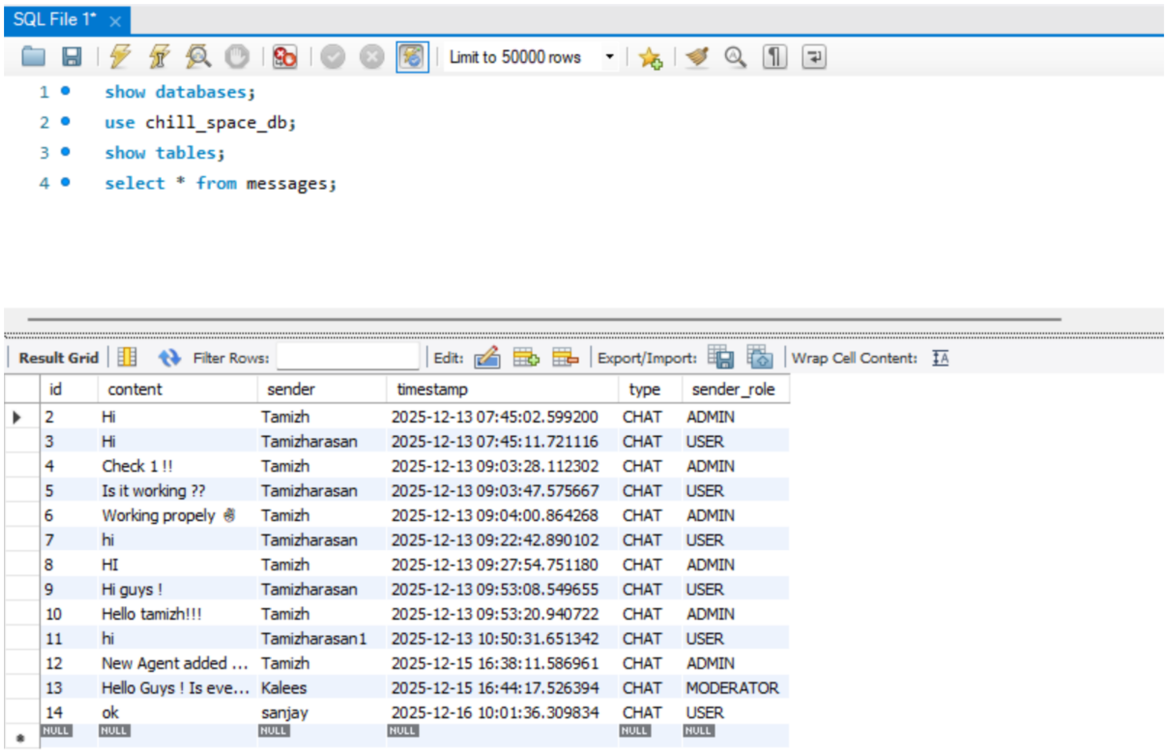
2) Shared_files TABLE :



```
show databases;
use chill_space_db;
show tables;
select * from shared_files;
```

	id	content_type	description	file_data	file_size	filename	original_filename	uploaded_at	uploaded_b
▶	1	image/jpeg	NULL	BLOB	37625	2880d9a1-350a-41b5-ad43-2b9fa67b17cc	pfp2.jpg	2025-12-15 16:42:55.387779	Tamizh
	2	image/jpeg	NULL	BLOB	44557	0e41f6b4-33dc-47bf-8821-48fdb032cd07	pfp3.jpg	2025-12-15 16:43:00.025933	Tamizh
	3	text/plain	NULL	BLOB	784	54e894dd-aba3-4ed2-94c9-fcd0b1cd8f8c	mcp work for chatbot.txt	2025-12-18 09:58:00.257845	Tamizh
	4	image/png	NULL	BLOB	927210	2f0e3b54-49ce-4a85-b189-9e6d095377bb	workflow 1.png	2025-12-18 09:58:08.504995	Tamizh
	5	text/csv	NULL	BLOB	3922	9fd22213-a563-43d6-963a-a0e72c803ef1	users_rows.csv	2025-12-18 09:58:19.389066	Tamizh
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

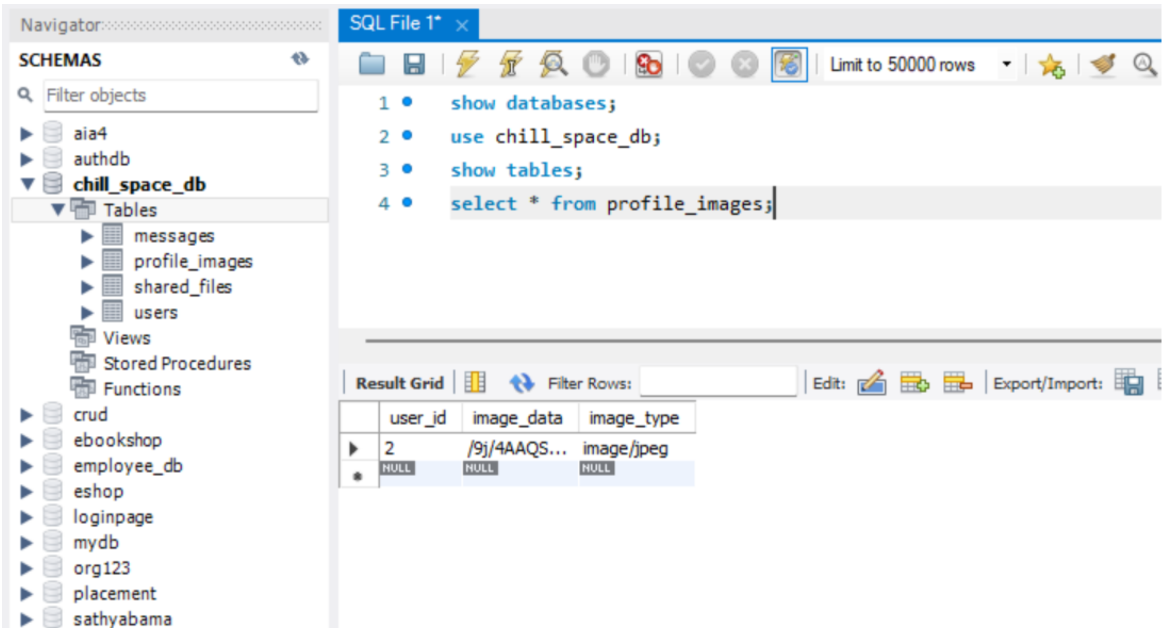
3) messages TABLE :



```
show databases;
use chill_space_db;
show tables;
select * from messages;
```

id	content	sender	timestamp	type	sender_role
2	Hi	Tamizh	2025-12-13 07:45:02.599200	CHAT	ADMIN
3	Hi	Tamizharasan	2025-12-13 07:45:11.721116	CHAT	USER
4	Check 1 !!	Tamizh	2025-12-13 09:03:28.112302	CHAT	ADMIN
5	Is it working ??	Tamizharasan	2025-12-13 09:03:47.575667	CHAT	USER
6	Working propely 👌	Tamizh	2025-12-13 09:04:00.864268	CHAT	ADMIN
7	hi	Tamizharasan	2025-12-13 09:22:42.890102	CHAT	USER
8	HI	Tamizh	2025-12-13 09:27:54.751180	CHAT	ADMIN
9	Hi guys !	Tamizharasan	2025-12-13 09:53:08.549655	CHAT	USER
10	Hello tamizh!!!	Tamizh	2025-12-13 09:53:20.940722	CHAT	ADMIN
11	hi	Tamizharasan1	2025-12-13 10:50:31.651342	CHAT	USER
12	New Agent added ...	Tamizh	2025-12-15 16:38:11.586961	CHAT	ADMIN
13	Hello Guys ! Is eve...	Kalees	2025-12-15 16:44:17.526394	CHAT	MODERATOR
14	ok	sanjay	2025-12-16 10:01:36.309834	CHAT	USER
NULL	NULL	NULL	NULL	NULL	NULL

4) Profile_images TABLE:



```
show databases;
use chill_space_db;
show tables;
select * from profile_images;
```

user_id	image_data	image_type
2	/9j/4AAQS...	image/jpeg
NULL	NULL	NULL

CONCLUSION

Chill Space successfully demonstrates a modern, scalable backend system built using Spring Boot and enterprise-level technologies. The project integrates secure authentication, real-time communication, AI-powered assistance, and clean architectural design, making it suitable for production use. Overall, it serves as a strong placement-oriented project that reflects practical backend development skills and industry best practices.

DRIVE LINK (for project demo):

https://drive.google.com/file/d/18M_Wh1dRVIOpbwxsW5WRxCUlQL03Fzp7/view?usp=sharing