

Chat Application Project: Structure, Database, UI, and Responsiveness

This presentation covers the chat app's architecture, database design, user interface, and responsiveness. We'll explore multithreading for smooth message handling and discuss key technologies like Python, JAVA and PostgreSQL.



Project Structure: Modular Design

Backend

Handles server logic and database operations using Java.

Frontend

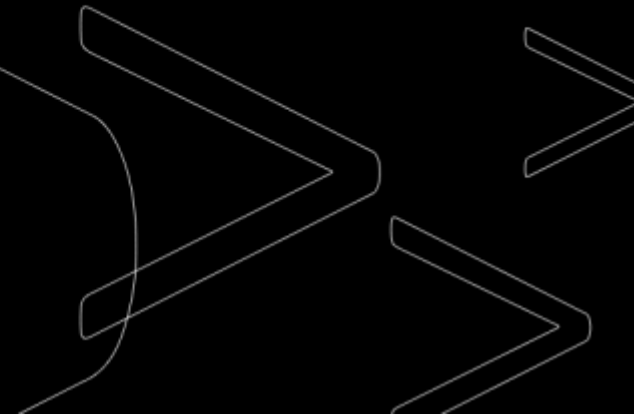
Chat interface built with React or Angular for smooth user interaction.

API Layer

Stockserver for secure communication between frontend and backend.

Modules

Client server for User authentication, messaging, and group chat functionality as separate units.



Project Structure: Multithreading Implementation



```
graph LR; 1[1 Thread Pool] --> 2[2 Message Queue]; 2 --> 3[3 Thread Safety]; 3 --> 4[4 Client Port];
```

1 Thread Pool

Enables concurrent handling of multiple user messages efficiently.

2 Message Queue

Manages incoming messages orderly before processing.

3 Thread Safety

Server Port to prevent data conflicts and race conditions

4 Client Port

A Client Server for multithreading operations within multiple users.

Database Design: Data Relationships

Relationships

- One-to-many: Users to Messages (users send multiple messages)
- Many-to-many: Users and Groups via Group_Members

Optimizations

- Indexes on foreign keys for faster queries
- JAVA chosen for reliability and concurrency

Database Connectivity: API Endpoints

To start server

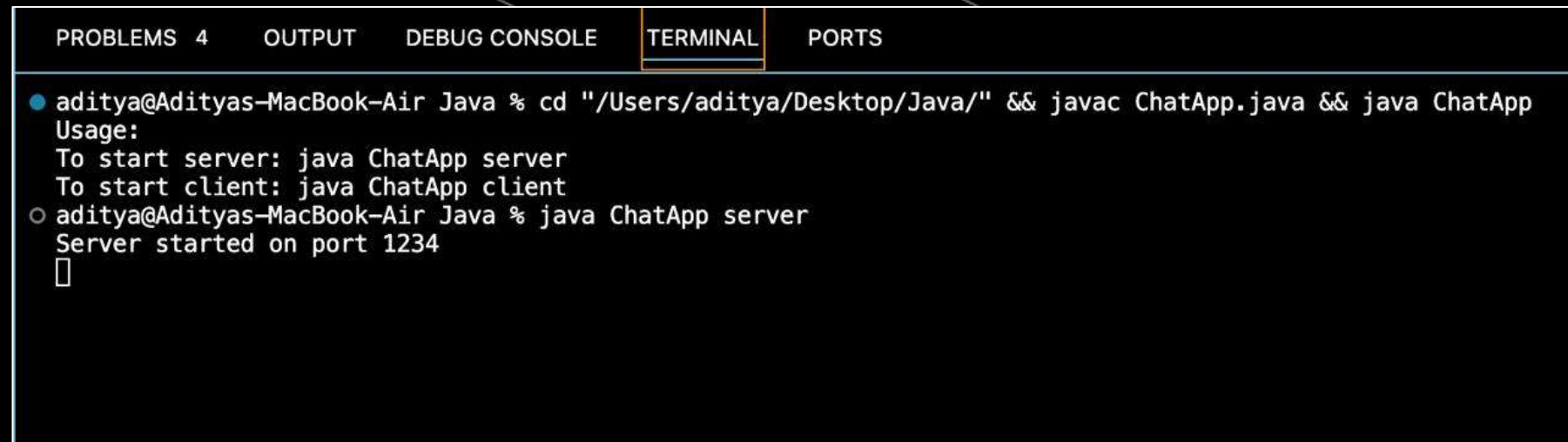
Use;- java ChatApp server

- Server starts on Port 1234,

Which we created using private Socket.

To start client:

Use :- java ChatApp client



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
● aditya@Adityas-MacBook-Air Java % cd "/Users/aditya/Desktop/Java/" && javac ChatApp.java && java ChatApp
Usage:
To start server: java ChatApp server
To start client: java ChatApp client
○ aditya@Adityas-MacBook-Air Java % java ChatApp server
Server started on port 1234
□
```

User Interface: Key Components



The screenshot shows an IDE terminal window with three tabs: PROBLEMS, OUTPUT, and DEBUG CONSOLE. The TERMINAL tab is active, displaying the execution of a Java chat application. The left pane shows the command prompt with the following commands and output:

```
cd "/Users/aditya/Desktop/Java/"
&& javac ChatProgrammee.java &&
java ChatProgrammee
aditya@Adityas-MacBook-Air ~ % c
d "/Users/aditya/Desktop/Java/"
&& javac ChatProgrammee.java &&
java ChatProgrammee
Usage:
To start server: java ChatApp se
rver
To start client: java ChatApp cl
ient
aditya@Adityas-MacBook-Air Java
% java ChatApp server
Server started on port 1234
Received: hii
Received: hoo
[]
```

The right pane shows the output of the client application:

```
aditya@Adityas-MacBook-Air Java
% java ChatApp client
Connected to chat server.
hii
hoo
how are you
[]
```

Chat Window

Displays real-time message streams smoothly.

Input Field

Area for composing and sending messages easily.

User List

Multiple user at a same time within a Chat Panel .

Group Chat Panel

Lists user's groups and quick access to chats.

User Interface: Design Principles



Clean & Intuitive

Simple navigation between chats and groups.

Multiple user on same panel.



Real-time Updates

Instant message display for active conversations.

Messages encrypted within client servers

Aesthetics and Styling



Color Palette

Modern, with light and dark theme options.

Typography

Consistent fonts for clarity and readability.

Visual Cues

Icons enhance usability and guide user actions.



Conclusion: Key Takeaways

Modular Structure

Promotes maintainability and scalability.

Robust Database

Efficient data storage with relational integrity.

User-Friendly Interface

Engaging experience with real-time responsiveness.

Responsive Design

Supports diverse devices and platforms seamlessly.