REAL-TIME CHAT APPLICATION WITH ON-THE-FLY TEXT COMPRESSION SUPPORTING EMOJI AND MULTILINGUAL MESSAGES

DONE BY :

22MID0033 – LOKESHWARI G

22MID0257 – ANU SWATHI V R

22MID0269 – SUBASHREE S

Abstract

Overview of the chat compression system

- This project introduces a real-time multilingual chat application integrated with an optimised data compression system. Traditional algorithms like Gzip and Huffman coding work well for simple text but show reduced efficiency when handling dynamic, multilingual chat data.
- To overcome this, the system uses Zstandard (Zstd) enhanced with a custom-trained dictionary tailored to multilingual chat patterns. The architecture includes:
  - A dictionary training module
  - Real-time compression engine
  - A live web interface to view compression results instantly
  - A persistent chat archive for session-level comparison
- The system evaluates compression ratios across Zstd, Gzip, and Huffman for every session.
- Experimental results show that dictionary-assisted Zstd achieves the highest compression ratio, reduced storage usage, and faster decompression, especially in multilingual environments where redundancy patterns vary across languages.

Introduction – Background

- Rapid growth in digital communication generates **massive volumes of text data** every second.
- Messaging apps and multilingual chat platforms require efficient **storage, transmission, and processing**.
- **Data compression** has become essential to overcome:
  - Storage limitations
  - Bandwidth constraints
  - Real-time communication demands

Existing Approach

Traditional algorithms:

- Huffman → symbol-based
- LZW → dynamic dictionary
- Gzip → DEFLATE-based

Limitations:

- No linguistic awareness
- Poor handling of mixed-language messages
- Inefficient for short or emoji-heavy text
- No cross-message pattern optimisation

Limitations of Traditional Compression

Common Algorithms :

- Huffman Coding
- Run-Length Encoding (RLE)
- Lempel-Ziv family (LZ77, LZW)

Issues :

- Designed primarily for monolingual or uniform datasets.
- Reduced efficiency with multilingual chat data mixing English, Hindi, Tamil, etc.
- Lack of repetitive patterns across languages leads to:
  - Lower compression ratios
  - Ineffective redundancy detection

Project Goal:
To demonstrate how the algorithms Huffman, Gzip, Zstandard wprks for real time Data Streaming and to store for the backup data.

Need for a Multilingual-Aware Solution

- Multilingual data introduces **diverse linguistic structures**.
- General compression algorithms fail to capture **cross-language patterns**.
- Requires a **customized, adaptive compression approach** that learns multilingual text behavior.

Proposed Solution – Overview

Goal: Develop a compression system optimised for multilingual chat data.

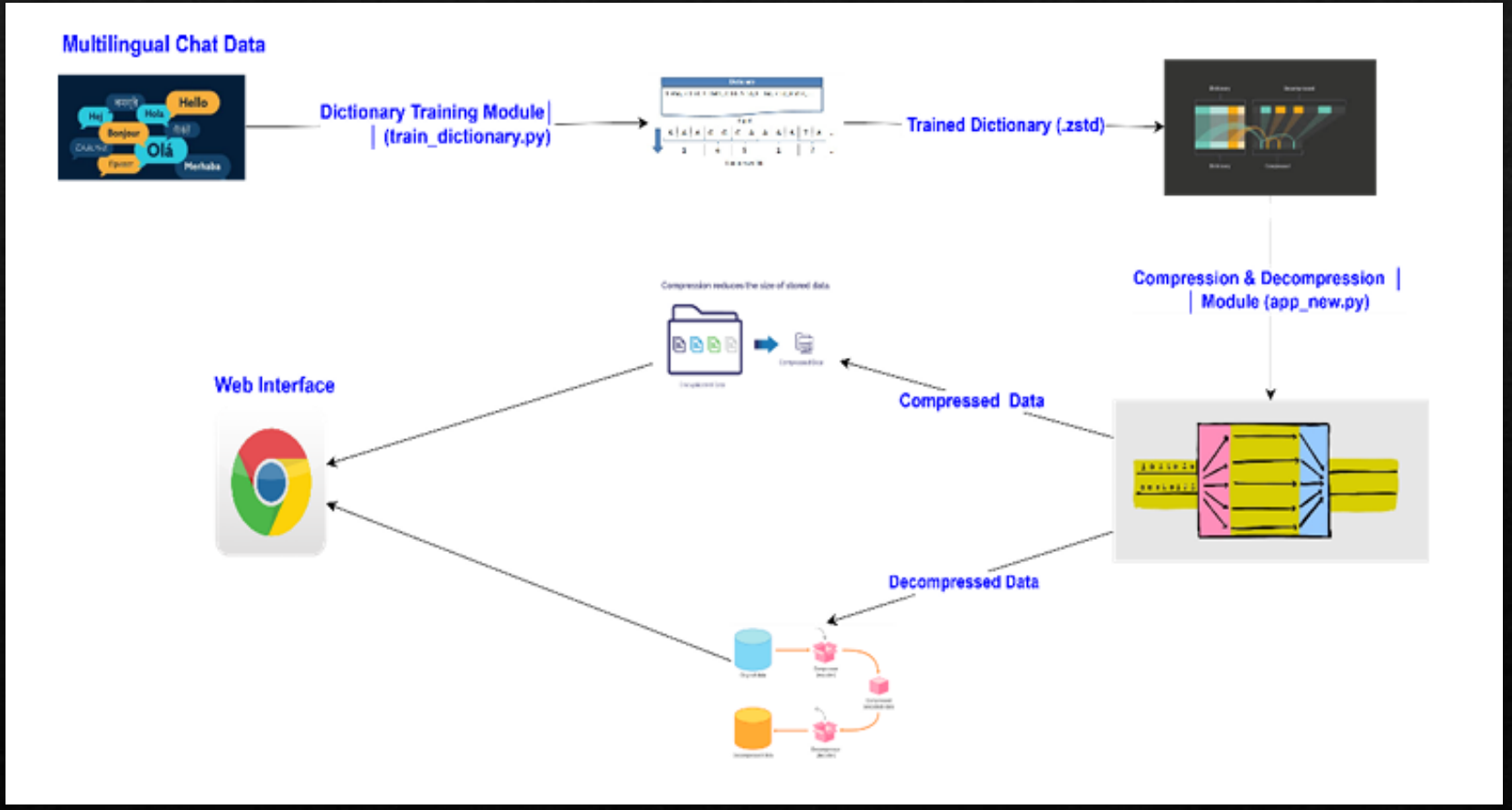
Approach:

- Use Zstandard (Zstd) for its speed and high compression capability.
- Enhance Zstd with a custom-trained dictionary built from multilingual chat samples.
- Capture common words, phrases, and patterns across languages.

System Architecture – Key Modules

1. **Dictionary Training Module**
   - Trains multilingual dictionary
   - Captures frequent words, emojis, patterns
2. **Compression/Decompression Module**
   - Performs fast, lossless compression using Zstd + Dict
3. **User Interface**
   - Web-based real-time chat
   - Shows size, ratio & comparison results

System Architecture of how the Process will be done



Overall Strategy

Combine the strength of **general-purpose compression (Zstd)** with **specialized multilingual pattern learning.**

Bridges the gap between:

- Traditional algorithms
- Real-world multilingual communication needs

Improves:

- Compression ratio
- Storage efficiency
- Transmission speed

Detailed Workflow

1. Collect multilingual chat data
2. Train Zstd dictionary (multilingual_chat_dict.zstd)
3. Integrate dictionary with Flask-SocketIO chat app
4. For each message:
   - Compress using Zstd + Dict, Gzip, Huffman
   - Calculate size & ratio
5. Display compression results live in console

Cont..

6. Store chat in:
   - chat_history.json (raw text)
   - chat_archive.zst (compressed stream)
7. On disconnect, finalise compression
8. Show final summary table
9. Automatic reloading of previous history
10. Dictionary improves over time with more data

Benefits of Proposed System

- ❑ Higher compression ratios for multilingual data
- ❑ Faster decompression
- ❑ Real-time analytics for each message
- ❑ Supports emojis & code-mixed text
- ❑ Persistent storage with reduced size
- ❑ Scalable & adaptive with continuous dictionary training
- ❑ Lossless recovery of all messages

Comparison of Algorithms

Huffman

- Best for short messages
- Very high savings for single messages

Gzip

- Moderate performance
- Good for longer messages

Zstd + Dictionary

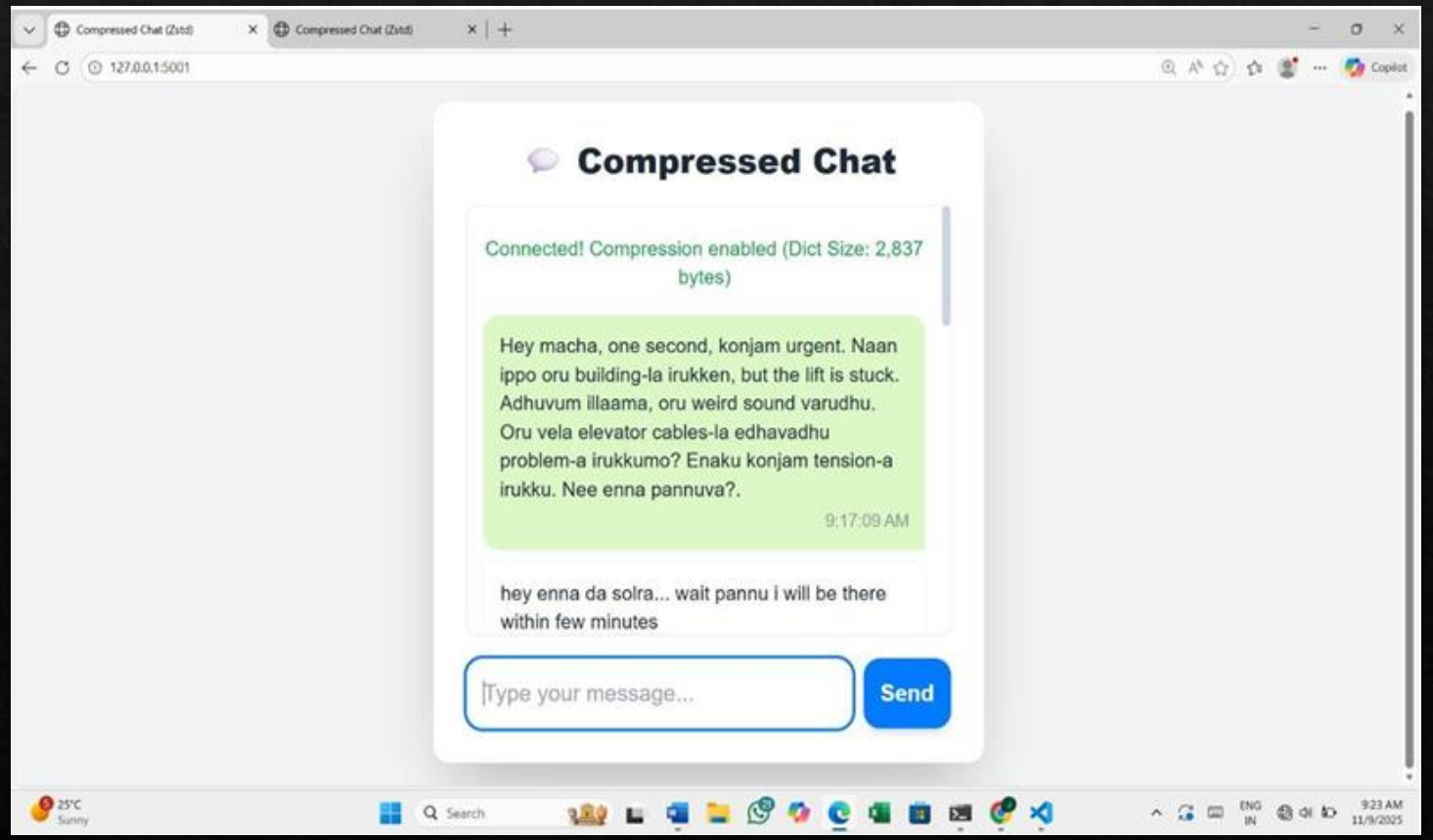
- Initially moderate for short texts
- Best for entire multilingual chat session
- Achieved 64.8% overall savings

Results & Discussion

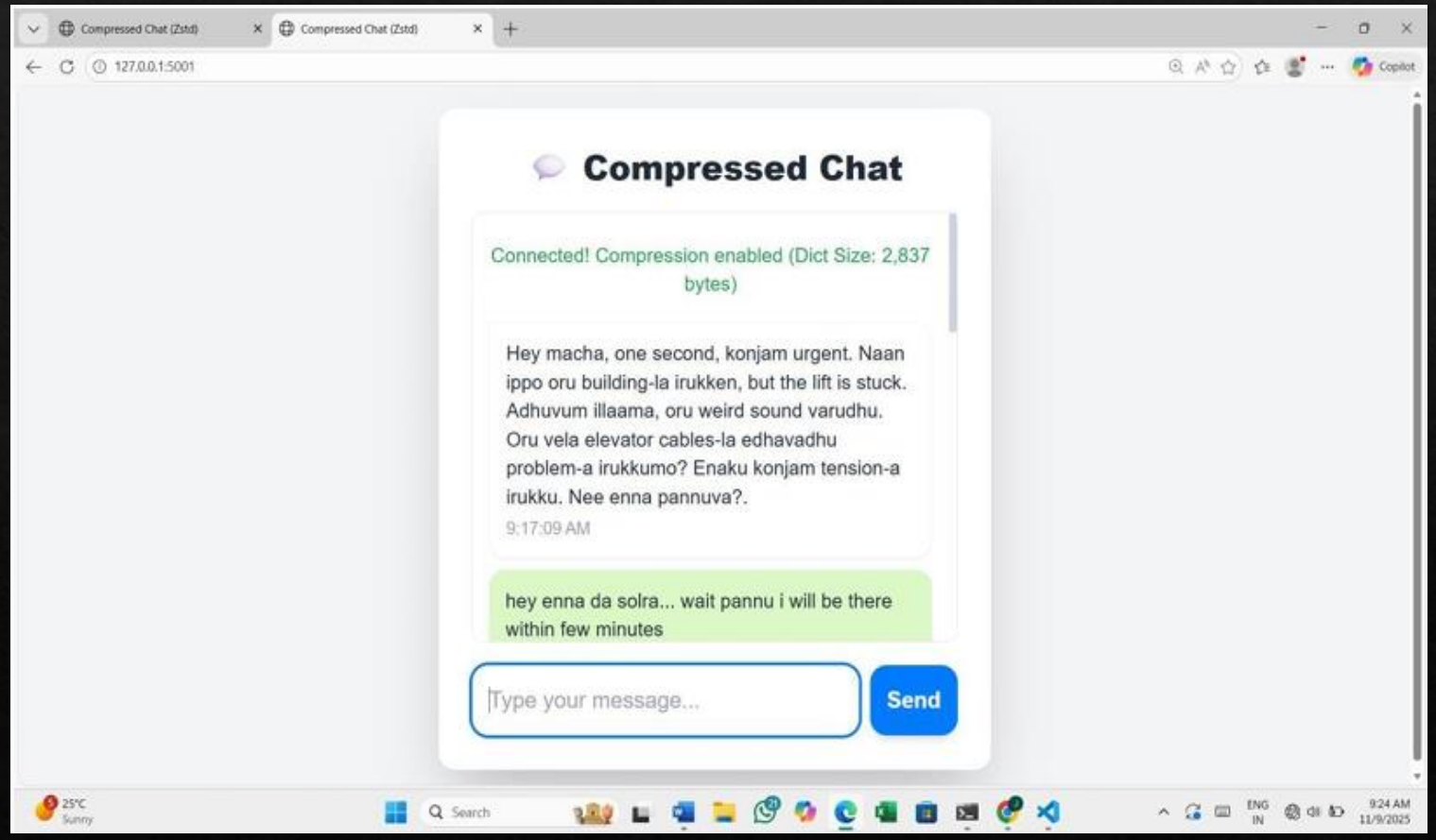
- Huffman performs best on individual short messages
- Gzip is balanced but limited
- Zstd + Dict achieves **highest overall compression**
- Captures cross-language redundancy
- Performance improves with chat length
- Ideal for real-time multilingual communication systems

SCREENSHOTS OF THE IMPLEMENTATION

Message from the sender side



Message from the Receiver side



Compression Table for the Messages

Compression Ratio for every message (in the terminal):

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	245	165	1.48x	32.7%
Gzip	245	185	1.32x	24.5%
Huffman	245	137	1.79x	44.1%

Message: Hey macha, one second, konjam urgent. Naan ippo oru building-la irukken, but the lift is stuck. Adhuvum illaama, oru weird sound varudhu. Oru vela elevator cables-la edha vadhu problem-a irukkumo? Enaku konjam tension-a irukku. Nee enna pannuva?.

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	66	57	1.16x	13.6%
Gzip	66	77	0.86x	-16.7%
Huffman	66	33	2.00x	50.0%

Message: hey enna da solra... wait pannu i will be there within few minutes

---

Compression Table for each of the Messages

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	216	154	1.40x	28.7%
Gzip	216	169	1.28x	21.8%
Huffman	216	118	1.83x	45.4%

Message: Serious-a solren, naan ninaichen andha sound namma phone vibration nu. But, when I kept my phone silent, sound konjam neram stopp aachu. Ippo back to normal. Enaku oru doubt, building-ae vibrate agura madhiri irukku.

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	19	19	1.00x	0.0%
Gzip	19	32	0.59x	-68.4%
Huffman	19	2	9.50x	89.5%

Message: ohh🥹🥹🥹

Compression Table for every message in the chat

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	202	147	1.37x	27.2%
Gzip	202	165	1.22x	18.3%
Huffman	202	111	1.82x	45.0%

Message: Hey, don't tell anyone. Andha building owner enaku message pannaru, "All is well" nu. But andha msg-ku appuram, phone-a off pannitaaru. Ennaku oru bad feeling. Inga irukkuravangala pathi enna panrathu?.

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	89	82	1.09x	7.9%
Gzip	89	95	0.94x	-6.7%
Huffman	89	41	2.17x	53.9%

Message: nee bayapadatha i am on the way...ennaku accurate location share mattum pannu😉😉😉

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	231	167	1.38x	27.7%
Gzip	231	181	1.28x	21.6%
Huffman	231	125	1.85x	45.9%

Message: Andha sound yenna-nu kandupidichen! Andha building owner-um naan-um, andha lift-a The Great Indian Laughter Challenge audtion room-a change panna plan pannom. Ippo comedian practice panra sound dha antha vibration. Semma prank ah?.

Compression Comparison Table:

Algorithm	Orig Size (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	36	36	1.00x	0.0%
Gzip	36	41	0.88x	-13.9%
Huffman	36	6	6.00x	83.3%

Message: 😂😂😂😂prank ah😁😁

Overall Summary of the Whole chat Messages

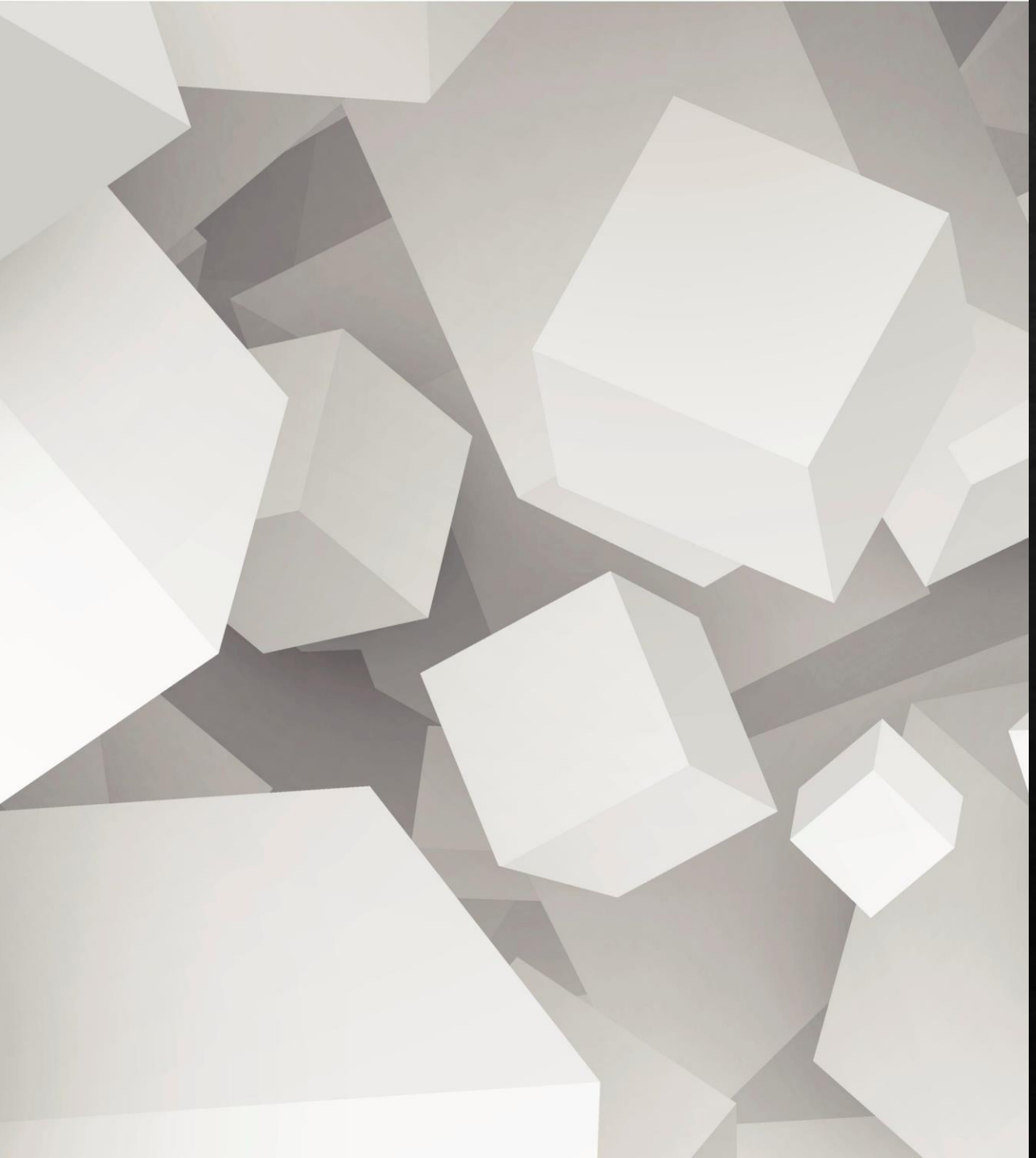
Final Compression Summary:

Final Compression Summary:

Algorithm	Orig (B)	Compressed (B)	Ratio	Savings (%)
Zstd +Dict	2627	926	2.84x	64.8%
Gzip	2627	952	2.76x	63.8%
Huffman	2627	1551	1.69x	41.0%

Conclusion

- Zstd with a trained dictionary outperforms Huffman & Gzip
- Efficient for multilingual and emoji-rich chat messages
- Integrates smoothly with real-time Flask-SocketIO application
- Ensures:
  - High compression
  - Low storage
  - Fast decompression
  - Lossless recovery
- Dictionary retraining makes system adaptive & scalable



Thank You!