# Zest_Challenge-2 PubSub Team- Mumbai

**Members:** Prashant, Ankit, Shubham, Mihir

**Mentor:** Shreeishita Gupta

## Experience and Coordination

The work division was in such a way that everybody learned everything used in this project. After the learning phase, everybody sat and worked on a single system together via Teamviewer. We explored various Kafka-libraries like Segmentio and Shopify-Sarama in order to understand the variety of libraries and the different features they have. Almost complete work is also done in the Sarama, can be found on another branch of the Github Repo. Regularity and sincerity were also maintained during the meetings. And Subtask completion, before the deadline set by the mentor, is the strength of this team.

## Learning

We followed the "learn then earn (implement)" strategy, which is the key to the successful completion of the project. In this project, we came across the most powerful tool (Kafka) used in the production world, which couldn't be happened without this project. We also learned the most trending and fairly new language (Golang), which was our dream, but we couldn't get started with this earlier. We are surprised at how we can manage to learn all these in a short time, but it could be possible with such a helpful mentor. All these learnings will definitely be proved to be Plus-point for us in the future.

## Features

1. Messages can be produced based on key or partition or in a least-byte manner. If the key is empty, then partition number allocation will be in action. If both empty, then the least-byte allocation will be served by the program.
2. ConsumerGroup can be subscribed to **multiple** topics, keeping this property in mind, we can specify more than one topic for consuming, and the subscription handle happens at the service bootup.
3. As soon as the new topic is typed in the JSON object, A new topic will be created with ten partitions by default.
4. SMS and Email both clients are in action with the proper fault-tolerant system. For Email, we used **Gomail,** and for SMS **Gotwilio** library is used.
5. If the client or the consumer service itself is down, no worries, messages are safe and will be consumed again when service or client is up.
6. Made extensive documentation for running these services on any system.
7. Email and SMS client credentials are safe and stored in the environment file.
8. Millions of messages are processed in a small instance of time. Thanks to Goroutines.

## Extra/Good to Have Features

1. Design principles and standard conventions of a typical Golang project are strictly followed.
2. Testing is easy to implement for such a well-designed code. Extensive reading of articles related to UT shared by the mentor was also done. If the time would have allowed us, this could add feather in the cup.
3. All the configuration constants are stored in the JSON file, and the Viper library is used to read the same.
4. Goroutines made our service tremendously fast and accurate that our service is able to process hundreds of thousands of messages in no time.

## Resources

- Codes
- Documentation