collectAI coding challenge

Thanks for trying our coding challenge!

With this document, you should have received

- A CSV file (customers.csv) which contains customer data and
- A communication service that provides a JSON API (commservice.linux / commservice.mac / commservice.windows)

If you haven't received these files, or if you think there are problems with them, please contact us immediately. To run the commservice select the binary suitable for your operating system and execute it. It listens on port 9090 by default.

The Challenge

Your task is to build an app which reads customer data from a CSV file (customers.csv) and sends out reminders of unsettled invoices based on the specified schedule. The schedule's values are offsets relative to the first message being sent.

To send a message make a POST request to the commservice's `/messages` endpoint. It expects the message to be JSON encoded in the request body and contain the customer's email and message's text, e.g. look like

```
{
  "email": "user@email.com",
  "text": "hello user"
}
```

If the communication service has been able to decode the message successfully, it responds with a 201 HTTP status code. It is possible that a customer settles an invoice. In this case, the communication service's JSON encoded response body additionally contains `{"paid": true}` and the customer shouldn't receive any additional messages.

After your service has sent out all messages terminate the commservice to get a report. (Use `SIGTERM` or `SIGINT` on Unix based operating systems, not `SIGKILL`.)

Your solution

We expect you to send us your code of a fully functional application to the proposed challenge using the Go programming language. Try to avoid using third-party libraries. If

not possible, please write a paragraph to describe the reason for using it. Your code should build and run on a recent release of Linux or Mac. Please provide a short description of how to run it.

What's important to us

We expect you to write code you would consider production-ready. This means your code should be simple, robust and follow good practices. We will look for

- If your solution fulfills the requirements and runs against the supplied communication service
- How clean your design and implementation are, how easy it is to understand and maintain your code
- How you verified your software