# Managing Payments in Trippie

Managing payments within a mobile application is always an intimidating and challenging task for a young developer. This is due the amount of sensitive information being handled during the processing of payments as well as the security concerns. Payments processors reduce these risks and provide secure solutions for managing payments within your application. In our case we have decided to handle payments using the Stripe API with cloud functions for Firebase. The reason for using Stripe over other payment processors is because Stripe is easy to set up, use and most developers love it. It even provides a user interface that can be incorporated into our application.

The following steps outline how the payment system will be set up within our app in a safe and seamless way for both our customers as well as us the developers.

1. Create a stripe customer
   1. When a user logs in to the app for the first time, a firebase cloud function is triggered. The purpose of the function is to communicate with the Stripe server API to create a new customer.
   2. The customer ID returned from stripe is then saved into the database (Firebase Firestore), inside the document of the user who just registered. This customer ID will be used when making our next API calls to stripe.
2. Create payment method – in our app we are allowing users to post multiple trippies and therefore it would make sense to allows users to add a card that is linked to their account so that they do not have to enter the credit card information every time they make a payment. Here is a breakdown of how this functionality will be implemented:
   1. We will have the client stripe SDK create a token which will later be exchanged with stripe for a source or payment method (i.e. credit card).
   2. The token is then stored in cloud firestore which in turn will fire a cloud function.
   3. The cloud function serves the purpose of communicating with the server stripe SDK to create a source for that customer. That source is then written to cloud firestore. What is saved in the database is just a way of identifying the credit card and no private information is saved in our backend.
3. Make the payment
   1. Our client app will write a charge to cloud firestore.
   2. A cloud function is triggered that communicates with the stripe server to process the payment. Important note: cloud functions always make sure that it successfully runs at least once, this means that the same function may run more than once. This is something to keep in mind as we don’t want our customers to be charged more than once. This is where the concept of Idempotency comes to get around this issue. (clink on this [stack overflow link](https://stackoverflow.com/questions/1077412/what-is-an-idempotent-operation) for a more in depth understanding of an idempotent function)
   3. Once the payment processing is done, the result is written into the database.

# References

Goldblatt, S. (2018, October 30). Go serverless: manage payments in your apps (Firebase Summit 2018) [Video File]. Retrieved from https://www.youtube.com/watch?v=JeyxolsJ3aE

Hewgill, G. (2009, July 3). *Stack Overflow*. Retrieved from Stack Overflow questions. What is an idempotent operation?: https://stackoverflow.com/questions/1077412/what-is-an-idempotent-operation

Person, J. (2018, November 7). *Go serverless: Manage Payments in your Apps with Cloud Functions for Firebase*. Retrieved from A medium corporation: https://medium.com/firebase-developers/go-serverless-manage-payments-in-your-apps-with-cloud-functions-for-firebase-3528cfad770