React Stripe Document

Client Configuration

First store the Stripe publishable key at .env of client app

```
REACT_APP_STRIPE_API_KEY=pk_test_asdfasd
```

Install “react-stripe-checkout” for client

npm install react-stripe-checkout

```
<StripeCheckout token={handleToken}
                billingAddress
                stripeKey={stripeAPIKey}
                name="Stripe Payment"
                amount={CartUtil.calcGrandTotal(cartItems) * 100}
                description="Payments with Stripe"
                currency="INR"
                zipCode
                image={stripeImage}
                panelLabel="Pay for {{amount}}">
</StripeCheckout>
```

Handle the payment with the function

```
let handleToken = async (token?, addresses?) => {
    const body = {
        product : {
            name : `${cartItems[0].name} & Others`,
            amount : CartUtil.calcGrandTotal(cartItems) * 100,
            currency : 'INR',
        },
        customer : {
            name : addresses.billing_name,
            address : {
                line1: addresses.billing_address_line1,
                postal_code: addresses.billing_address_zip,
                city: addresses.billing_address_city,
                state: addresses.billing_address_state,
                country: addresses.billing_address_country,
            }
        },
        description : "Shopping from BrainsKart",
        email : token.email,
        source : token.id,
        stripeTokenType : token.type
```

```
    };

    // dispatch action
    dispatch(orderActions.makeStripePayment(body,history));
}
```

Dispatch an action for payment

```
export const makeStripePayment = (body, history) => {
    return async (dispatch:any) => {
        dispatch({type : MAKE_STRIPE_PAYMENT_REQUEST});
        try {
            // set token
            if(AuthUtil.isLoggedIn()) {
                let token = AuthUtil.getToken();
                TokenUtil.setTokenHeader(token);
                TokenUtil.setStripeKey();
                let dataURL:string =
`${process.env.REACT_APP_SERVER_URL}/api/payments/checkout`;
                let response = await axios.post(dataURL, body);
                dispatch({type : MAKE_STRIPE_PAYMENT_SUCCESS, payload
: response.data});
                dispatch(alertActions.setAlert('Payment is Success' ,
'success'));
                history.push('/');
                // place an order
            }
        }
        catch (error) {
            console.error(error?.response?.data);
            dispatch({type : MAKE_STRIPE_PAYMENT_FAILURE, payload :
error?.response?.data});
        }
    };
};
```

Set the Authorization in the Axios request header

```
public static setStripeKey(){
    axios.defaults.headers.common['Authorization'] = `Bearer
${process.env.REACT_APP_STRIPE_API_KEY}`;
}
```

Server Configuration

First store the Stripe Secret key in the .env file

```
STRIPE_SECRET_KEY=sk_test_asfasf
```

Install packages below

npm install stripe @types/stripe uuid @types/uuid

```
const Stripe = require('stripe');
const apiKeySecret = 'sk_test_stripe key';
const stripe = Stripe(apiKeySecret);
```

use the below interface to accept the payment body

```
interface PaymentBody{
    product : {
        name : string;
        amount : number;
        currency : string;
    },
    customer : {
        name : string;
        address : {
            line1: string;
            postal_code: string;
            city: string;
            state: string;
            country: string;
        }
    },
    description : string;
    email : string;
    source : string;
    stripeTokenType : string;
}
```

use the below router to accept the payments

```
paymentRouter.post('/checkout', verifyToken, cors(), async (request ,
response) => {
    console.log(request.body);
    try {
        let userId = request.header['user'].id;
```

```
        const paymentBody:PaymentBody = request.body;
        const customer = await stripe.customers.create({
            name : paymentBody.customer.name,
            address : {
                line1: paymentBody.customer.address.line1,
                postal_code: paymentBody.customer.address.postal_code,
                city: paymentBody.customer.address.city,
                state: paymentBody.customer.address.state,
                country: paymentBody.customer.address.country,
            },
            description : paymentBody.description,
            email: paymentBody.email,
            source: paymentBody.source,
            metadata: {
                userId: userId,
            },
        });

        if(paymentBody.stripeTokenType === 'card'){
            const idempotencyKey = v4();
            const charge = await stripe.charges.create(
                {
                    amount : paymentBody.product.amount,
                    currency: paymentBody.product.currency,
                    customer: customer.id,
                    description: paymentBody.description,
                },
                {
                    idempotencyKey,
                }
            );
        }
        else{
            throw Error(`Unrecognized Stripe token type:
"${paymentBody.stripeTokenType}"`);
        }
        response.status(200).json({msg : 'payment is success'});
    }
    catch (error) {
        console.error(error);
        response.status(200).json({errors : [{msg : error.message}]});
    }
});
export default paymentRouter;
```