```
const moment = require('moment');
const request = require('request');
const config = require('../../config');
const mcache = require('memory-cache');
const async = require('async');
const { clear, show, remove } = require('../../services/mcache');
const logger = require('../../services/logger');
const { Order } = require('../orders/model');
const responseHandler = require('../../services/response');
const { voidSACTSales, releaseWOT, voidWOTSales, voidWirecardPayment,
flagCriticalError, toggleEnv } = require('./utility');
const { confirmWOT } = require('./process_confirmations');
const { smtpMail } = require('./emailer');

exports.processWOTConfirm = (res, req, wirecard, wot_pin, reservation_id,
transaction_id, email_type, batch) => {
  const wot_query = { 'pincode.wot': wot_pin }
  Order.findOne(wot_query, { __v: 0, _id: 0 })
    .exec((err, order) => {
      if(!err)  {
        if(order !== null)  {
          let card_digits = req.body.digits
          let wot_confirm_in = {
            name: order.name,
            email: order.email,
            contact: order.contact,
            order_id: order.order_id,
            receipt_number: order.receipt_number,
            card_digits: card_digits,
            wot_pin: wot_pin,
            transaction_id: transaction_id,
            res: res,
            req: req
          }
          confirmWOT(wot_confirm_in, function(err, confirm_wot_response) {
            if(!err)  {
              // update db
              let update = {$set:{'status': 'confirmed', 'transaction_id':
req.body.transaction_id }}
              if(wirecard)  {
                update = {$set:{'status': 'confirmed', 'transaction_id':
req.body.transaction_id, 'card_digits': req.body.digits, 'bank_approval_code':
req.body.bank_approval_code, 'payment_type': req.body.payment_type }}
              }
```

```
            Order.findOneAndUpdate(wot_query, update, { __v: 0, _id: 0,
reservation_id: 0, transaction_id: 0}, function(err, updated_order) {
              let attraction = order.attractions[0]
              // retrieve rest of attraction details
              request.get({
                url: toggleEnv(config.env) + attraction.id
              }, function(err, response, attr_body) {
                if(!err)  {
                  attr_body = JSON.parse(attr_body)
                  attraction.title = attr_body.data.directory.title
                  attraction.thumbnail = attr_body.data.directory.thumbnail
                  attraction.redemption_info =
attr_body.data.directory.redemption_info
                  updated_order.attractions[0] = attraction
                  updated_order = updated_order.toObject()
                  updated_order['date_time'] =
moment(updated_order.date_time).format("MMM Do YY, h:mm a")
                  Order.findOneAndUpdate(wot_query, {$set: {'attractions':
attraction}}, { __v: 0, _id: 0, confirmed: 0}, function(err, latest_order) {
                   if(!err)  {
                    // re-cache user's orders
                    if(updated_order.user_id !== null) {
                      let key = '/ticketing/orders/user/' + updated_order.user_id
                      let cachedBody = mcache.get(key)
                      if(cachedBody)  { // exists, clear
                        remove(key)
                      }
                    }
                    logger.log('cosmodb', res.req.headers.correlation_id, null, 'info',
wot_query, null, updated_order)
                    smtpMail(latest_order.email, updated_order, latest_order,
email_type)
                    // confirmEmail(latest_order.email, updated_order, latest_order,
email_type)
                    batch ? console.log('batch (wot)') :
responseHandler.success(res, updated_order)
                   } else { // update order failed
                    var tasks = [];
                    tasks.push(async.apply(releaseWOT, order.receipt_number,
wot_pin));

                    if (wirecard) {
                      tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
                    }
```

```
                async.parallel(tasks, (err, response) => {
                  logger.log('cosmodb', res.req.headers.correlation_id, null,
'error', wot_query, null, err);
                  batch ? console.log('batch (wot)') :
responseHandler.confirmationUnsuccessful(res, { order_id: order.order_id });
                });
              }
            })
          } else { // error retriving rest of attraction details
            var tasks = [];
            tasks.push(async.apply(releaseWOT, order.receipt_number,
wot_pin));

            if (wirecard) {
              tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
            }

            async.parallel(tasks, (err, response) => {
              batch ? console.log('batch (wot)') :
responseHandler.confirmationUnsuccessful(res, { order_id: order.order_id });
            });
          }
        })
      })
    } else { // error confirming wot
      var tasks = [];
      tasks.push(async.apply(releaseWOT, order.receipt_number, wot_pin));

      if (wirecard) {
        tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
      }

      async.parallel(tasks, (err, response) => {
        batch ? console.log('batch (wot)') :
responseHandler.confirmationUnsuccessful(res, { order_id: order.order_id });
      });
      // responses handled within method
    }
  })
} else { // no order found
  flagCriticalError(res, reservation_id, transaction_id, 'order not found
error')
  batch ? console.log('batch (wot)') : responseHandler.notFound(res)
}
```

```
      } else { // cosmos db error
        var tasks = [];

        //cannot void WOT cus order does not exist TO FIX NEXT TIME
        //tasks.push(async.apply(voidWOTSales, transaction_id, wot_pin));

        if (wirecard) {
          tasks.push(async.apply(voidWirecardPayment, res,
req.body.payment_timestamp, req.body.request_id, req.body.transaction_id));
        }

        async.parallel(tasks, (err, response) => {
          flagCriticalError(res, reservation_id, transaction_id, 'db query error')
          logger.log('cosmodb', res.req.headers.correlation_id, null, 'error',
wot_query, null, err);
          batch ? console.log('batch (wot)') : responseHandler.serverError(res)
        });
      }
    })
}
```