Lab 2 - Error Handling

Setup

- Install node modules by running the `npm i` command on inside both *client/angular* and *server* folders
- Start the Coinspread client and server applications by running the `npm start` command inside both *client/angular* and *server* folders. This should start the Coinspread UI application at localhost:4000 and server application at localhost:3000.
- Login with any user
- Create a wallet from the Wallets menu if none exists
- On Friends menu, if no friends exist, click on the 'Out there' menu and add a user to Friends. (If no users are shown on the Out there menu, create additional users in the application using the signup tab on the login page.)

Tasks:

Task 1: Add Express Custom Error Handler

The application doesn't include any custom error handler. As a result, any uncaught synchronous error or async errors passed to `next(err)` get caught by the Express default error handler. The Express default error handler sends an error to the client in HTML format, which is not desirable. For example here is an error received on making a payment to a friend.
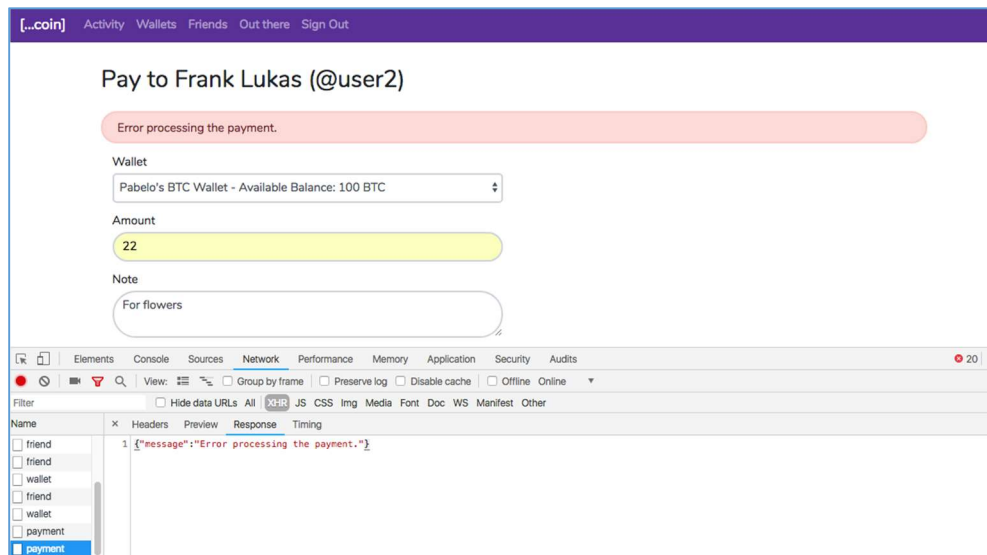

.

As part of this task, add a custom error handler (https://expressjs.com/en/guide/error-handling.html) in the `app/app.error.handler.js` that sends error message in format {"message": <Error Message>}".

As a result, the UI should receive and show errors as below.

Task 2: Use Boom API method to generate error objects

1. Click on the Activity link on the menu bar. It is expected to get a 404 response as this route is not implemented in this lab.
2. In the `app/app.error.handler.js`, locate a middleware that handles calls to any unimplemented routes, sending a 404 response.
3. Replace the error generation logic to use the notFound() (https://github.com/hapijs/boom#boomnotfoundmessage-data) method from Boom with an error message 'Resource not found!'.
4. As a result, on clicking the 'Activity' menu, you should get this error response on the UI.
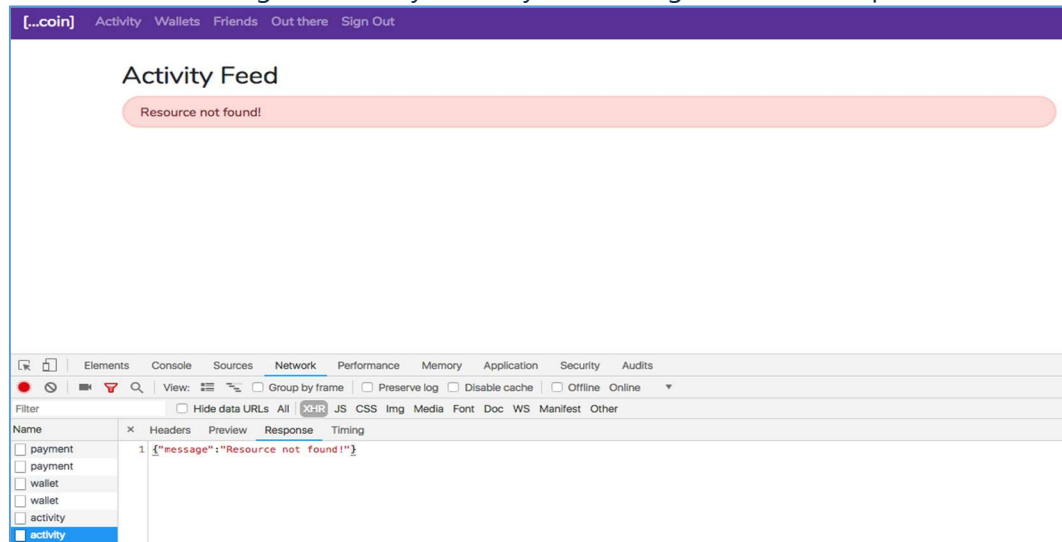
**Task 3: Handle synchronous and asynchronous errors**

In the 'Friends' menu, on making a payment to a user, the application fails with an error - *Error processing the payment.*
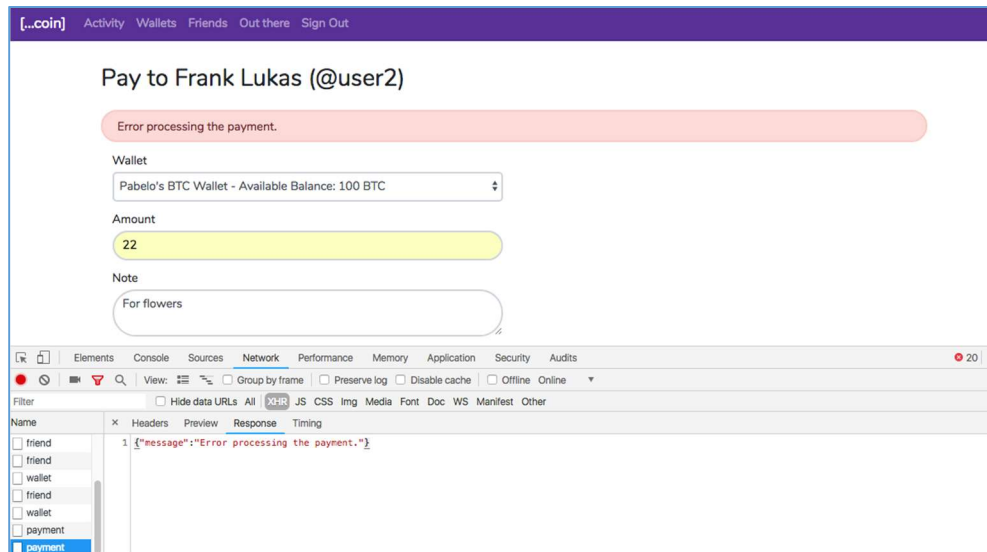
This error results from missing error handling in the `processPayment` method in `api/resources/payment/payment.controller.js` –

```
const processPayment = (paymentInfo) => {
    paymentGateway.useExchange1();
    paymentGateway.useExchange2();
    paymentGateway.useExchange3();
    paymentGateway.useExchange4();
    paymentGateway.useExchange5();
};
```

The first four `useExchangeX()` method delivers errors in different mechanisms - throws clause, callback param, event emitter, and rejected promise, respectively. As part of this task, handle errors sent from useExchangeX() methods and print the error message on console. As a result, the payment request should complete successfully, with success message as below:
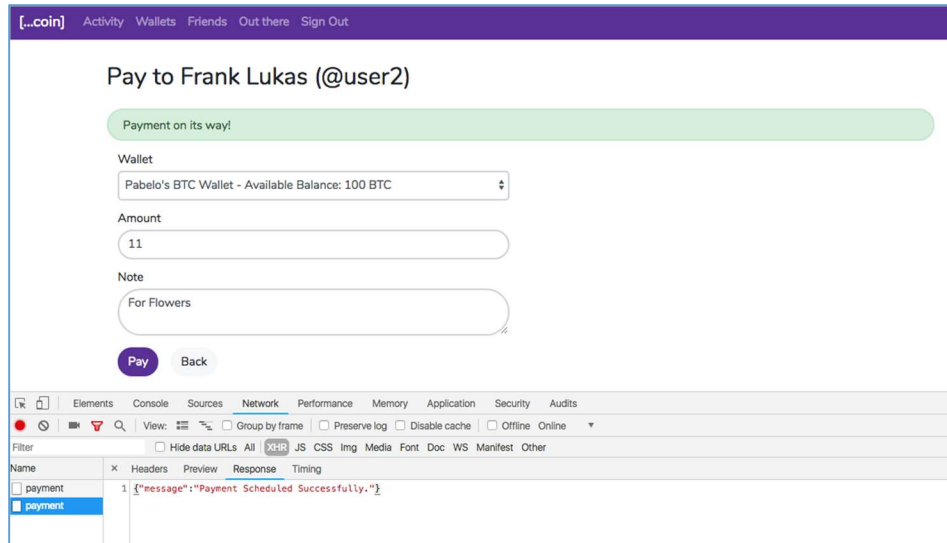
The server console, should include the statements, when error handler prints the error message on console.

Quiz

If not handled correctly, which of the error delivery mechanisms result in crashing the server?