Nova code challenge

# Intro

Welcome to my write up of the Node.js backend challenge. I would like to start out by saying that I personally love interviews like this as they allow to me to not only test my skills against a fun and challenging mini project but also provides me with an opportunity for learning and technical growth.

I want to start out by saying I have little experience implementing full stack solutions for web projects. Working in the front end isn’t something I have a lot of practice with and I find that the lack of good debugging tools as well as sometimes unpredictable behavior turned me off years ago. I have also never worked with node.js and honestly have done zero javascript coding in years so it was a lengthy process to get my development cycle moving. All that being said, I have enjoyed working on this little side project.

I’ll be splitting up the write up into sections that I feel represent my own thought process as well as the intended objective you wanted from assigning a code challenge

Contents

[Intro 1](#_Toc465179046)

[Architecture 1](#_Toc465179047)

[Front end 1](#_Toc465179048)

[Back end 1](#_Toc465179049)

[Iframe 1](#_Toc465179050)

[Server 1](#_Toc465179051)

[Logging DB 1](#_Toc465179052)

[Testing 1](#_Toc465179053)

[Final Thoughts 1](#_Toc465179054)

[Security 1](#_Toc465179055)

[Logging 1](#_Toc465179056)

[Reflection 1](#_Toc465179057)

# Architecture

## Front end

Being a backend developer by day, I tried to minimize the front end as much as possible. Here I simply created 1 static HTML page hosted on my local machine through no domain. I called this file JonnyLender.html as it is the bank side of the system. This main front end page contains an empty iframe pointing to the the node.js server, a form to include customer data, and a div to display information returned by the iframe

[INCLUDE PICTURE OF PAGE HERE]

* Iframe
* Form
* Div1

## Back end

I found this to be the most enjoyable part of the project for me. I felt more at home here as I could easily debug as well as use non browser console printing. I split the backend into 3 sections, the client page (which is the iframe used by the front end, the node.js server, and the logging DB

### Client page (Iframe)

This is the page that contains part of the magic that is the backend. It is the main point of communication between the frontend and the server to ensure any data flowing between the two is what we want and expect. It contains the following

* Window event listener: Place to receive message from the parent window. This is crucial from a cross domain communication stand point. I struggled to find a good solution but after many attempts, found that window.postmessage plus window.addeventlistener was the best way to go.
* Ajax request sender and receiver: Use this to communicate with the node.js server so that I didn’t have to send round trip call backs. This was relatively straight forward as it is a well documented and solved problem.
* Divs to display data: Use this to display customer data received from parent window and node.js server

### Server

The node.js server in this case creates 2 paths for requests to go through: forwarding request to the indexpage.html which displays the client page, and a path for ajax requests to come through.

* Indexpage route: Basically reads in the indexpage.html and writes it out to the response. I did it this way just to ease my own development and readability of the indexpage.html. Not sure if this is how it’s done other places though
* Ajax route: I check to determine whether or not a page is attempting an ajax request by checking the path of the request. If the path contains “/getCustomerData”, then I send through the request through to get a random reply from a list of ‘choices’ and add the reply to the response.

### Logging DB

# Deployment

I chose to go with the visual studio version of the node.js server add-in and it made the deployment and transfer of the project super easy. Steps to deploy

* Download project from GitHub (<https://github.com/jonnyhay/NodejsWebApp1/archive/master.zip>)
* Running with Visual studio is easiest so if you don’t have VS2015, you can download the community version of the IDE
* Install node.js visual studio plugin (https://www.visualstudio.com/vs/node-js/)
* Open .sln file with visual studio

# Testing

# Final Thoughts

## Challenges

### Cross domain communication

## Security

## Logging

## Reflection