Introduction

For the next 2 weeks, you will be an intern for Prosper IT Consulting, a sister company of The Tech Academy.

Prosper IT co-owns and runs the development of multiple popular web applications, including Blue Ribbons Review and a social media travel app called Bewander.

Your current project is a full-stack theme store for businesses.

With a team, you will be building a customized, database-integrated website template for purchase on the main theme store site.

On this project and as a team, you will:

- Customize the front end of an existing skeleton template as to colors and styling.
- Add specific required functionality.
- Create a database schema and implement it.
- Connect your application to the database.
- Deploy your site and database to a live server.

This project is where you will become a full-stack web developer.

#Details

On this project, you will be working in-depth with several technologies that were only touched on in the curriculum. Others you will know well. This project is designed to mirror an actual software development job, one where you won't know everything, but where you'll still be expected to do everything.

You are expected to fill the gaps in your knowledge through your own research and through collaboration with your team.

The Live Project Instructor will hold a daily stand-up with you and your team at 1pm, Monday through Friday. This person is available to answer questions and help explain concepts, but the coding and problem solving will be up to you and your team.

Your Live Project Instructor will supply you with the Github address of the skeleton template for this theme.

Project Management will be done through the web app Trello. Your link to the project:
https://trello.com/b/IywM8FxP

#Curriculum Additions

Definitions:

Web Application: A program, just like you'd find on your desktop computer, only it's hosted on a remote server and uses the browser as an interface.

A web app could be something as complex as facebook.com or as simple as an online calculator.

Web apps typically have 3 distinct parts: a database, a user interface, and some code to connect the two.

Scalable: Of an application, able to be converted easily to handle an increasing amount of users or traffic.

Robust: The capability of a program to deal with errors without breaking.

Framework: A pre-built architectural design of an application. Using a framework, a programmer can quickly make a scalable, robust web application with minimal work. This is because the flow of the application is already designed, leaving the developer free to focus on the specific coding and implementation.

Angular.js: A JavaScript framework for web applications, designed and maintained by Google.

From the Angular intro documentation:

> AngularJS is what HTML would have been, had it been designed for applications. HTML is a great declarative language for static documents. It does not contain much in the way of creating applications, and as a result building web applications is an exercise in *what do I have to do to trick the browser into doing what I want?*

Backbone.js: A JavaScript framework for developing single-page web applications.

Node.js: A server-side platform, written in C++, that can understand and execute JavaScript. Node.js uses the same high-performance JavaScript engine as the Google Chrome browser.

Until Node.js, JavaScript was used almost exclusively as a front-end language for manipulating HTML elements and content. Being able to use JavaScript to handle server requests has led to new, innovative web app frameworks and contributed to JavaScript's rise as arguably the most popular programming language in use today.

Contributing to Node's popularity is its built-in "library of libraries," called npm (Node Package Manager), which is the largest collection of open-source libraries in the world.

For example, prior to Node, using a SQL Server Database within an HTML/CSS/JavaScript site would involve advanced PHP or C# programming knowledge. Yet using the free Node.js library "mssql" from the Node Package Manager, it becomes not only possible, but relatively simple.

Express.js: A lightweight, server-side Node.js web application framework. Using Express.js, one could have a working sample web application in less than 5 minutes.

Almost all Node applications use Express.js or are Express-like.

React.js: An open-source JavaScript library (as opposed to a framework) for easy creation of robust user interfaces. React.js is built and maintained by Facebook.

#More on Node.js

Node.js is a large subject. To become expert in it involves a solid understanding of JavaScript, HTTP, and web servers. Its use becomes obvious in large applications involving real-time data.

For example, when LinkedIn switched to Node.js, it cut its servers from 30 to 3, and certain parts of its application ran 20 times faster.

Another example, when Walmart switched to Node.js in 2013, its Black Friday online traffic didn't bring its servers above 1% utilization.

Practical step:

Install Node.js

#More on Express.js

The three most important files in an Express.js file are:

Package.json: This file is written in JSON format and contains information (such as version number) regarding the libraries your app will use.

After you've added the proper libraries to your package.json file, simply run "npm install" on the command line inside your project folder, and those libraries will be automatically installed.

Inside the package.json file you can also specify a "starting point" of your app; i.e. the first file you want the application to run, the one that gets everything else going. This is typically the app.js file.

App.js: This is starting point of your app. Here is where the Express module is spun up, the server and port number are specified, and where you tell the application to look at your routes.js file.

Routes.js: This file specifies some "actions" you app can take, and what it exactly that means.

You'll mostly be using to HTTP "verbs": GET and POST.

These are "action" calls to the server and have very opposite meanings.

If you are to make an HTTP GET request, you are asking the server to GET you something, and you are asking very specifically by typically including details in the URL. Ever see a URL that ends with something like:

/test?name=jesse&lastname=johnson

That is a GET request that is asking the server to get something related to those names.

In a POST request, however, you are sending information to the server. This information isn't in the URL, but rather the message body of the server request.

In your routes.js file, you will specify specific HTTP verbs that relate to particular actions.

For example, a social media site may have a “route” defined for the “action” ‘socialmediasite.com/friend’. In this case, “friend” isn’t another file, but an action. There may be a friend.html or friend.js file, but not a “friend” file.

This section of the routes.js file for the social media file, may look something like this:

var express = require(‘express’) // the ‘require’ method is JavaScript’s syntax for importing other JavaScript files

var app = express() // calling the main method of the Express.js app, which starts up the app as “app”

(the above two statements would actually go in the app.js file, but the following would go in the routes.js:)

```
app.get(‘/friend’, function(request, response) {
        response.render(‘friend.html’)
}
```

In the above example, we are simply defining the “route” or “action” ‘/friend’ to tell the server to GET the friend.html file. This could, of course, be made more complex by adding further variables to the request.

Practical:

Do this example from the Request website:

https://expressjs.com/en/starter/hello-world.html

For Mssql you will need to download and install the package and setup the DB connection in the Routes file. This is also where you will add the email to the database. The Routes file will link all the pages as well.

Note: there is a header file that is pulled in on

Further information:

To run an application, you actually need 2 servers: a database server and a web server.

After you’ve made this project, you will host your local SQL Server database on a Gearhost.com remote server. For this size database, Gearhost is free.

For your web server, you will use a free tier with Heroku.

The end result is a autonomous app using
Start the Live Project!