Week - 1 Long Descriptive Questions

1. **You'll be building an application called Loc8r. Loc8r lists nearby places with Wi-Fi where people can go to get some work done. It also displays facilities, opening times, a rating, and a location map for each place. Users will be able to log in and submit ratings and reviews.**

Planning a real application

Loc8r will list nearby places with WiFi where people can go and get some work done. It will also displayfacilities, opening times, a rating, and a location map for each place.

Planning the application at a high level

The first step is to think about what screens we'll need in our application. We'll focus on the separate page views and the user journeys. We can do this at a high level, not concerning ourselves with the details of what's on each page. It also helps with organizing the screens into collections and flows, serving as a good reference point when we build it. As there's no data attached to the pages orapplication logic behind them, it's easy to add and remove parts, change what's displayed and where, and even change how many pages we want.

Planning the screens

Let's think about the screens we're going to need:

- A screen that lists nearby places
- A screen that shows details about an individual place
- A screen for adding a review about a place
- We'll probably also want to tell visitors what Loc8r is for and why it exists, and we should add
- another screen to the list:

- A screen for "about us" information

Dividing the screens into collections

Next, we want to take the list of screens and collate them where they logically belong together.

Architecting the application

On the face of it Loc8r is a simple application, with only a few screens. But we still need to think about how to architect it, because we're going to be transferring data from a database to a browser, letting users interact with the data and allowing data to be sent back to the database.

Starting with the API

Because the application is going to be using a database and passing data around, we'll start building the architecture with the piece we're going to need. Building an API to interface with our data is the base point of the architecture.

Application architecture options

At this point, we need to look at the specific requirements of our application and how we can puttogether the pieces of the MEAN stack to build the best solution

A Node.js and Express application

- A Node.js and Express application with Angular additions for interactivity
- An Angular SPA

Choosing an application architecture

No specific business requirements are pushing us to favor one architecture over another.

Wrapping everything in an Express project

Our goal is to keep things simple and contained and have everything inside a single Express project. With this approach, we have only one application to worry about hosting and deploying and one set of sourcecode to manage.

The end-product

We'll use all layers of the MEAN stack to create Loc8r.

---

2. Explain in detail about Node.js.

Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of webapplications using Node.js to a great extent.

- Node.js = Runtime Environment + JavaScript Library

Following are some of the important features that make Node.js the first choice of software architects.

Asynchronous and Event Driven

All APIs of Node.js library is asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

Very Fast

Node.js library is very fast in code execution.

Single Threaded but Highly Scalable

Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

No Buffering

Node.js applications never buffer any data. These applications simply output the data in chunks.

Example

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type':
  'text/plain'});res.end('Hello World!');
}).listen(8080);
```

- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database

To Initiate the Node.js File:

- Start the command line interface, write node myfirst.js and hit enter:
- Initiate "myfirst.js":
- C:\Users\Your Name>node myfirst.js

Node.js Modules:

- Consider modules to be the same as JavaScript libraries.
- A set of functions we want to include in your application.

3.Define Full Stack Development?

Full-stack development is developing all parts of a website or application. The full stack starts with the database and web server in the back end, contains application logic and control in the middle, and goes all the way through to the user interface at the front end.

The MEAN stack is a pure JavaScript stack comprised of four main technologies, with a cast ofsupporting technologies:

MongoDB — the database
Express — the web
framework Angular — the
front-end framework
Node.js—the web server

4.Why learn the full stack?

- We're more likely to have a better view of the bigger picture by understanding the different areasand how they fit together.
- We'll form an appreciation of what other parts of the team are doing and what they need to besuccessful.
- Like other team members, we can move around more freely.
- We can build applications end-to-end by ourselves without depending on other people.
- We can develop more skills, services, and capabilities to offer customers