School of Computer Engineering & Technology
Class: Third Year B.Tech CSE (Trimester VIII)
Course: Web Technology Laboratory (WTL)

WT Laboratory 7

Laboratory Continuous Assessment (LCA)[*Refer Rubric table in WTL manual]

Understanding of the Objective (5)	Performance (5)	Journal Submission and Ethics (Neatness, Handwriting, Timely submission) (5)	Orals (5)	Total (20)	Remarks	Instructor's Sign

Aim:

a) **Creating HTTP server in Node.js**
b) **Creating simple static file server**
c) **Using Express framework develop a website using Node JS and MySQL**

Objectives:

1. To understand difference between Node.js and PHP
2. To understanding how to choose correct java script framework for web development.
3. Create a simple web application using Node.js and MySQL.

Theory:

1. What is Node.js? Why it popular?
2. How to set up environment for Node.js?
3. Using Node as HTTP server and static File Server
4. Create simple CRUD application using Node.js and database

FAQ:

1. What is Full Stack Development? What are different technologies related to full stack development front end and back end? What are popular stacks?
2. How to choose Technology Stack for Web Application Development?
3. Compare Java script frameworks available for Web Development.

(Minimum 3 handwritten pages)

Note: Student is expected to attach this page as a title page of an assignment.

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Definition

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. 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 web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

Features of Node.js

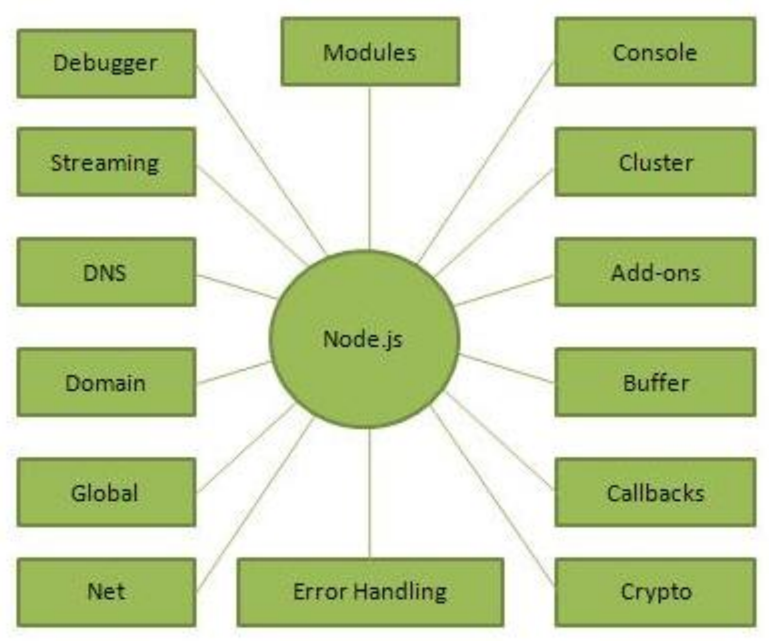
Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** − All APIs of Node.js library are 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** − Being built on Google Chrome's V8 JavaScript Engine, 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.
- **License** − Node.js is released under the MIT license

Who Uses Node.js?

Following is the link on github wiki containing an exhaustive list of projects, application and companies which are using Node.js. This list includes eBay, General Electric, GoDaddy, Microsoft, PayPal, Uber, Wikipins, Yahoo!, and Yammer to name a few.

Concepts



Why it is popular???

In a sequential language such as PHP, in order to get the HTML content of a page you would do the following:

```php
$response = file_get_contents("http://example.com");
print_r($response);
```

In Node, you register some callbacks instead:

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

http.request({ hostname: 'example.com' }, function(res) {
  res.setEncoding('utf8');
  res.on('data', function(chunk) {
    console.log(chunk);
  });
}).end();
```

There are two big differences between the two implementations:

- Node allows you to perform other tasks while waiting to be notified when the response is available.
- The Node application is not buffering data into memory, but instead it's outputting it chunk-by-chunk.

Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Where Not to Use Node.js?

It is not advisable to use Node.js for CPU intensive applications.

For more information explore: https://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js

Node.js application examples: https://magnetoitsolutions.com/blog/node-js-application-examples

Environment Setup:

- Node JS installed on your PC.
- Basic understanding of Node JS and Express JS.
- Knowledge of SQL, you should know and understand how to query a database.
- phpmyadmin installed on your PC. I recommend installing xampp as it already contains phpmyadmin in it.
- Understand how to use templating engines -- we are going to be using ejs in this tutorial).
- A text editor or IDE of your choice.

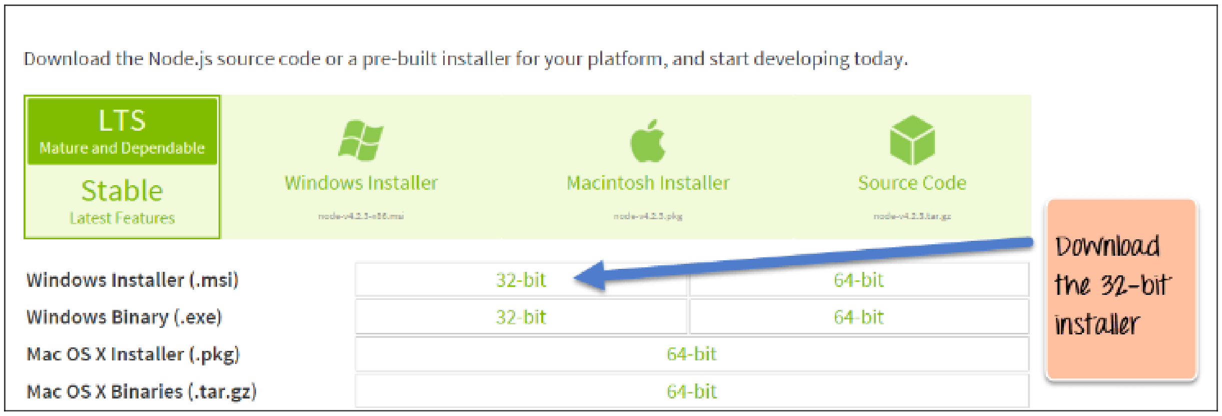
To start building your Node.js applications, the first step is the installation of the node.js framework. The Node.js framework is available for a variety of operating systems right from Windows to Ubuntu and OS X. Once the Node.js framework is installed, you can start building your first Node.js applications.

Node.js also has the ability to embedded external functionality or extended functionality by making use of custom modules. These modules have to be installed separately. An example of a module is the MongoDB module which allows you to work with MongoDB databases from your Node.js application.

How to install Node.js on Windows

The first steps in using Node.js is the installation of the Node.js libraries on the client system. To perform the installation of Node.js, perform the below steps;

Go to the site https://nodejs.org/en/download/ and download the necessary binary files. In our example, we are going to download the 32-bit / 64-bit setup files for Node.js.


Download the Node.js source code or a pre-built installer for your platform, and start developing today.

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.exe)	32-bit	64-bit
Mac OS X Installer (.pkg)	64-bit	
Mac OS X Binaries (.tar.gz)	64-bit	

Follow defaults steps and finish installation.

Installing NPM (Node Package Manager) on Windows

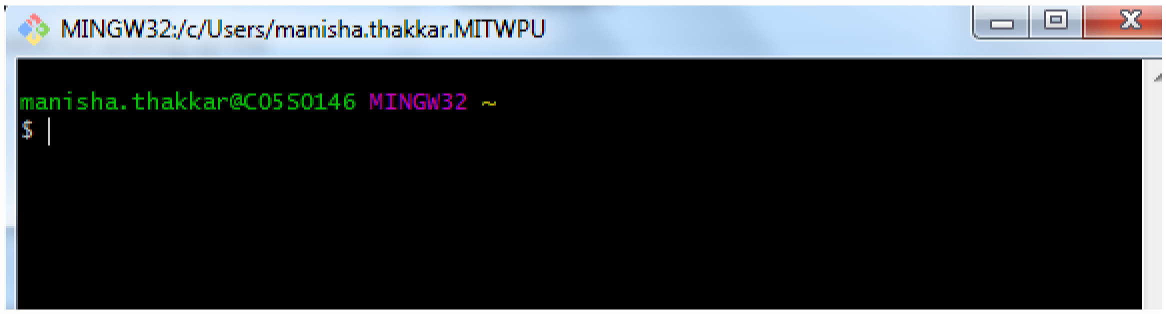
NPM is a "package manager" that makes installing Node "packages" fast and easy. A package is just a code library that extends Node by adding useful features. For example, the "request" package simplifies the process of making HTTP requests so you can easily get web resources from other sites.

NPM is installed when you install Node.js®

Prerequisites

You should have some familiarity with an application that lets you issue command line instructions. For example, the Windows Command Prompt, PowerShell, Cygwin, or the Git shell (which you get when you install Github for Windows).

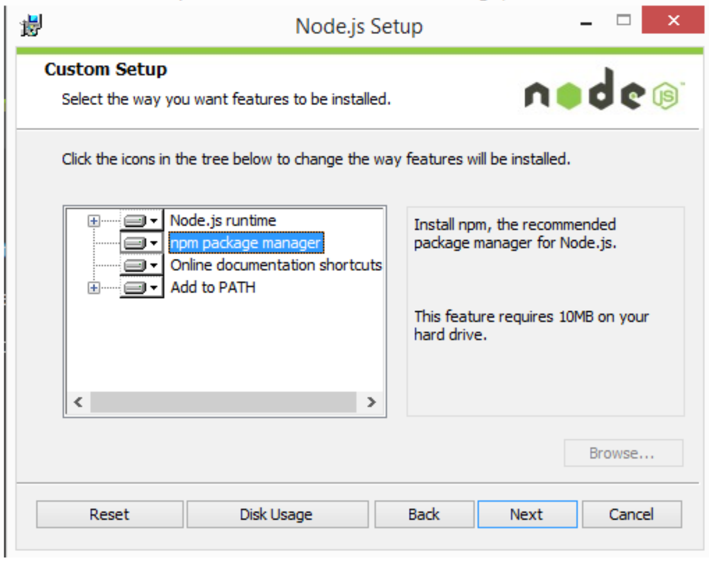
- Follow **the prompts in the installer**
  1. First of all download the Git Bash


MINGW32:/c/Users/manisha.thakkar.MITWPU

```
manisha.thakkar@C05S0146 MINGW32 ~
$ |
```

Main difference between command prompt and the git bash is:
In command prompt we use back slashes (\) but in GitBash we use forward slashes (/)

Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).

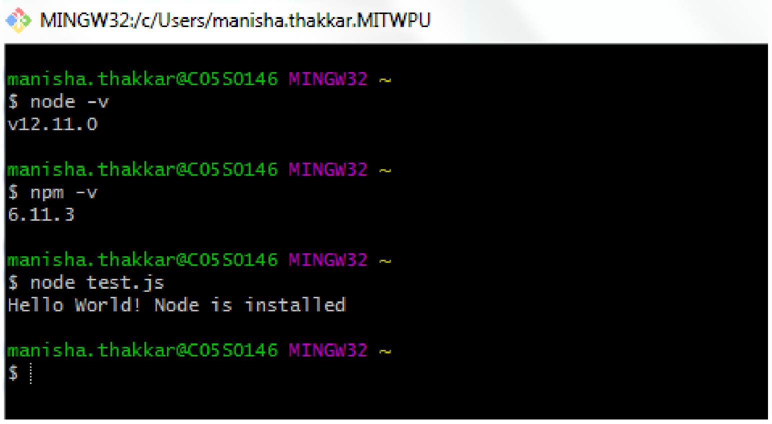

Node.js Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Node.js runtime
npm package manager
Online documentation shortcuts
Add to PATH
Install npm, the recommended package manager for Node.js.
This feature requires 10MB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel

- Restart **your computer.** You won't be able to run Node.js until you restart your computer.

Test it!

Make sure you have Node and NPM installed by running simple commands to see what version of each is installed:

- **Test Node.** To see if Node is installed, open the Windows Command Prompt, Powershell or a similar command line tool, and type `node -v`. This should print the version number so you'll see something like this `v0.10.35`.
- **Test NPM.** To see if NPM is installed, type `npm -v` in Terminal. This should print the version number so you'll see something like this `1.4.28`
- **Create a test file and run it.** A simple way to test that node.js works is to create a simple JavaScript file: name it hello.js, and just add the code `console.log('Node is installed!');`. To run the code simply open your command line program, navigate to the folder where you save the file and type `node hello.js`. This will start Node.js and run the code in the `hello.js` file. You should see the output `Node is installed!`.



MINGW32:/c/Users/manisha.thakkar.MITWPU

```
manisha.thakkar@C05S0146 MINGW32 ~
$ node -v
v12.11.0

manisha.thakkar@C05S0146 MINGW32 ~
$ npm -v
6.11.3

manisha.thakkar@C05S0146 MINGW32 ~
$ node test.js
Hello World! Node is installed

manisha.thakkar@C05S0146 MINGW32 ~
$
```

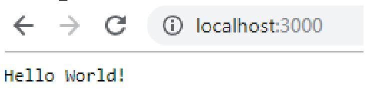
Creating HTTP server in Node.js

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Use the `createServer()` method to create an HTTP server:

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

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(3000); //the server object listens on port 8080
```

Output:



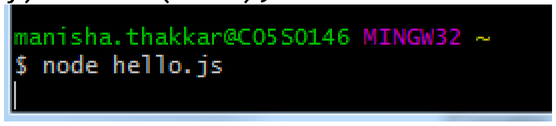
```
localhost:3000

Hello World!
```

Creating simple static file server (hello.html)

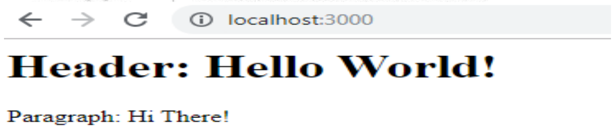
```
<html>
<body>
<h1>Header: Hello World!</h1>
<p>Paragraph: Hi There!</p>
</body>
</html>
```

Create a Node.js file that reads the HTML file, and return the content: (hello.js)

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('hello.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    res.end();
  });
}).listen(3000);
```



```
manisha.thakkar@C0550146 MINGW32 ~
$ node hello.js
```



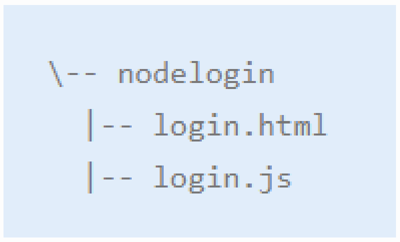
```
localhost:3000
```

Header: Hello World!

Paragraph: Hi There!

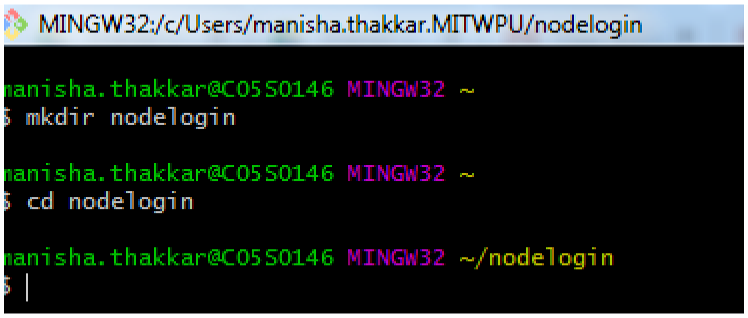
Using Express framework develop a website using Node JS and MySQL

Folder Structure: **(* Students are required to create other application using node.js and complete CRUD operations**)



```
\-- nodelogin
  |-- login.html
  |-- login.js
```

Creating the directory and change path to Project directory



```
MINGW32:/c/Users/manisha.thakkar.MITWPU/nodelogin

manisha.thakkar@C05S0146 MINGW32 ~
$ mkdir nodelogin

manisha.thakkar@C05S0146 MINGW32 ~
$ cd nodelogin

manisha.thakkar@C05S0146 MINGW32 ~/nodelogin
$ |
```

Initialize the Project

Run the command: **`npm init`** from inside the directory, it will prompt us to enter a package name, enter: **`login`**.

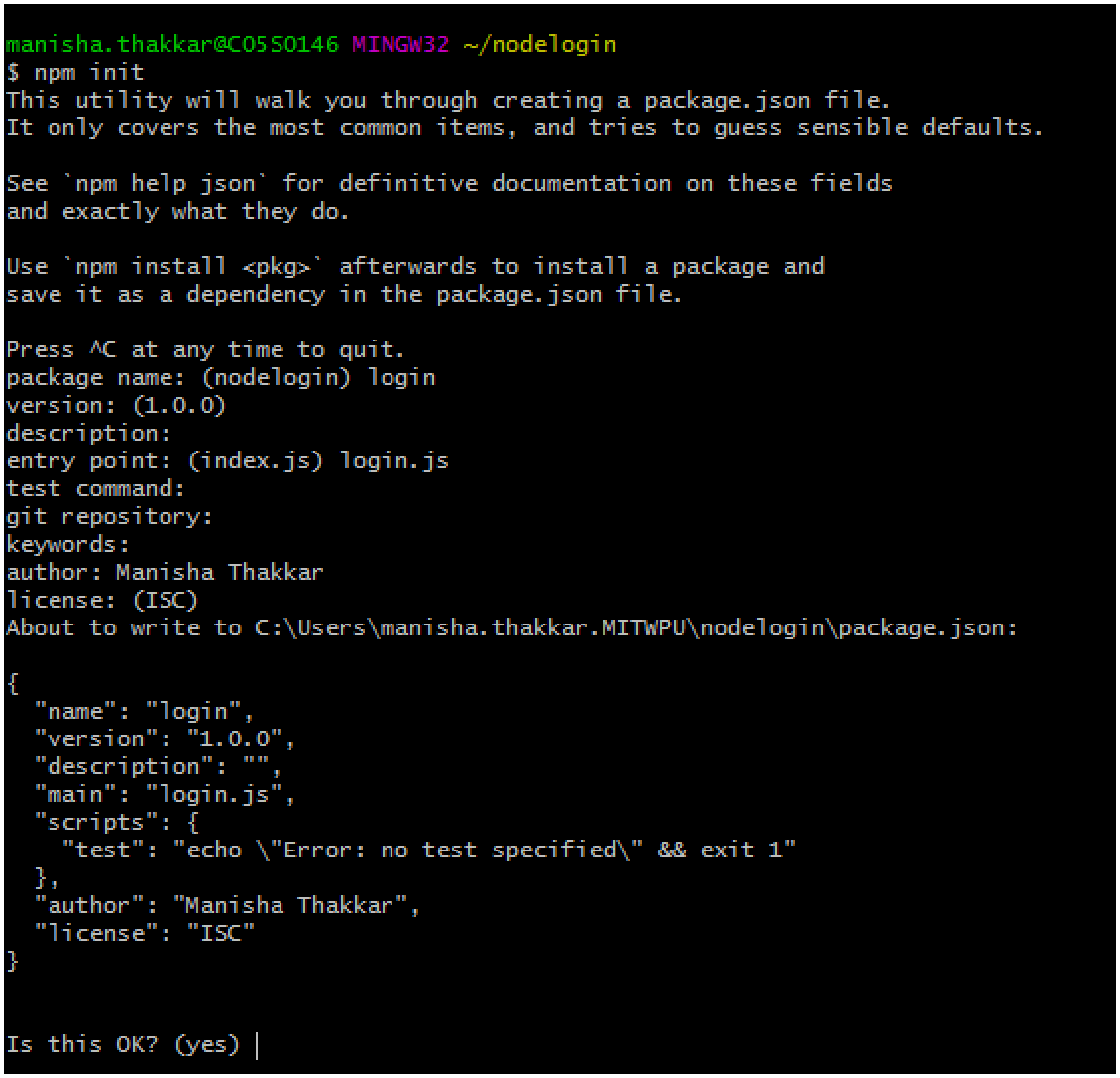
When it prompts to enter the entry point enter **`login.js`**.

Now we need to install the packages listed in the requirements, while still in the command line run the commands listed in the requirements above.

We should now have a new directory called: **`node_modules`** with all the modules installed

Run the command: `npm init` from inside the directory, it will prompt us to enter a package name, enter: `login`.

$ npm init



```
manisha.thakkar@C05S0146 MINGW32 ~/nodelogin
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (nodelogin) login
version: (1.0.0)
description:
entry point: (index.js) login.js
test command:
git repository:
keywords:
author: Manisha Thakkar
license: (ISC)
About to write to C:\Users\manisha.thakkar.MITWPU\nodelogin\package.json:

{
  "name": "login",
  "version": "1.0.0",
  "description": "",
  "main": "login.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Manisha Thakkar",
  "license": "ISC"
}


Is this OK? (yes)
```

Install required modules

The following modules are going to be needed to successfully build the app.

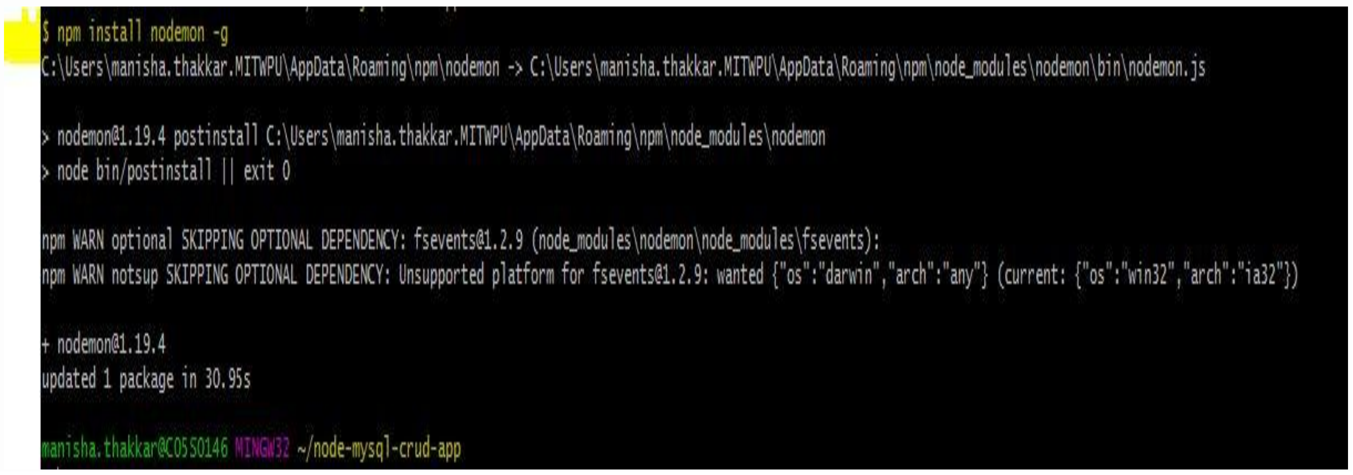
- MySQL Server
- Node.js
- Express - Install with command: `npm install express`.
- Express Sessions - Install with command: `npm install express-session`.
- MySQL for Node.js - Install with command: `npm install mysql`.

Then type the following command to install the last module globally on your PC.

What is Nodemon?

You started your server and it's finally up and running. You make a change or two, save the file, and open up your browser again. Something is wrong. It didn't reload and now you find yourself manually restarting the server every time. This is where Nodemon comes in.

Nodemon is a development dependency that **monitors for any changes** in your Node.js application and **automatically restarts the server**, saving time and tedious work.

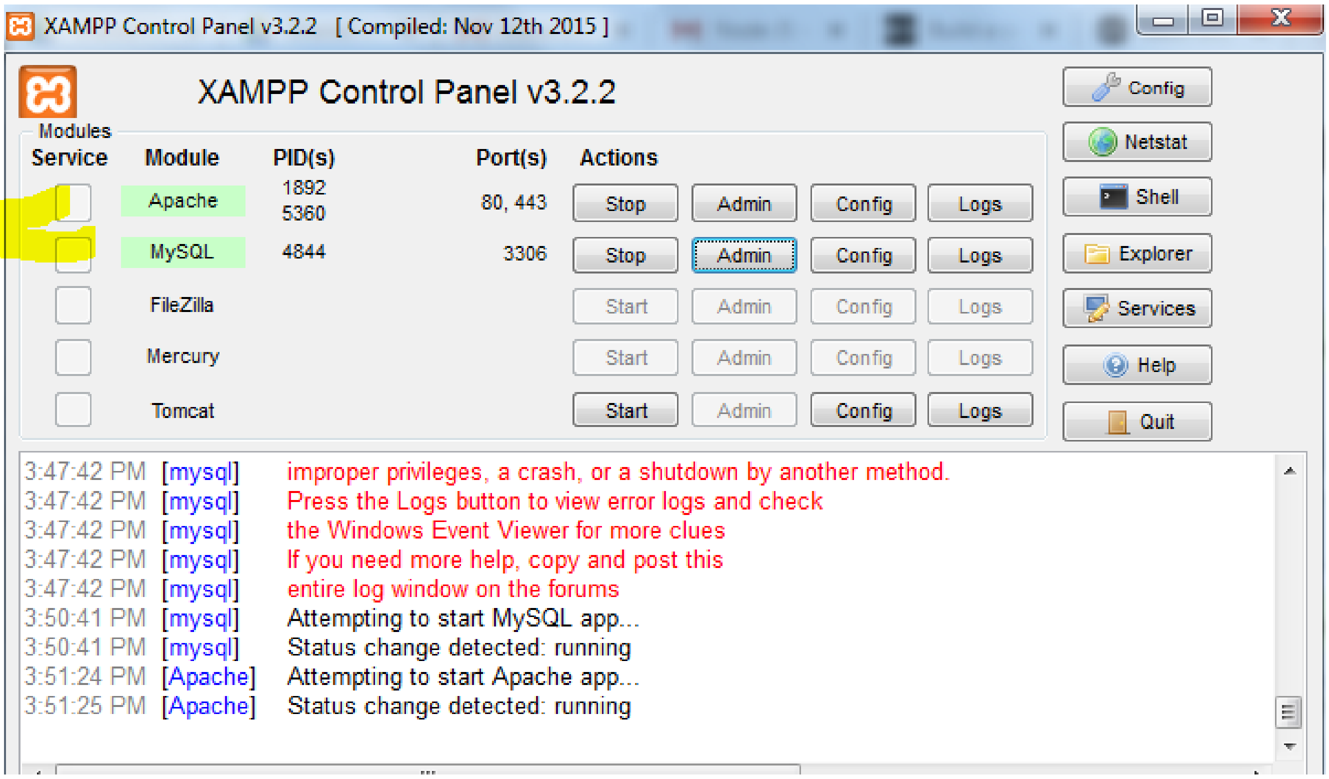


```
$ npm install nodemon -g
C:\Users\manisha.thakkar.MITWPU\AppData\Roaming\npm\nodemon -> C:\Users\manisha.thakkar.MITWPU\AppData\Roaming\npm\node_modules\nodemon\bin\nodemon.js

> nodemon@1.19.4 postinstall C:\Users\manisha.thakkar.MITWPU\AppData\Roaming\npm\node_modules\nodemon
> node bin/postinstall || exit 0

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\nodemon\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"ia32"})

+ nodemon@1.19.4
updated 1 package in 30.95s

manisha.thakkar@C05S0146 MINGW32 ~/node-mysql-crud-app
```

Tutorial is available here : https://codeshack.io/basic-login-system-nodejs-express-mysql/

Creating the database for the app

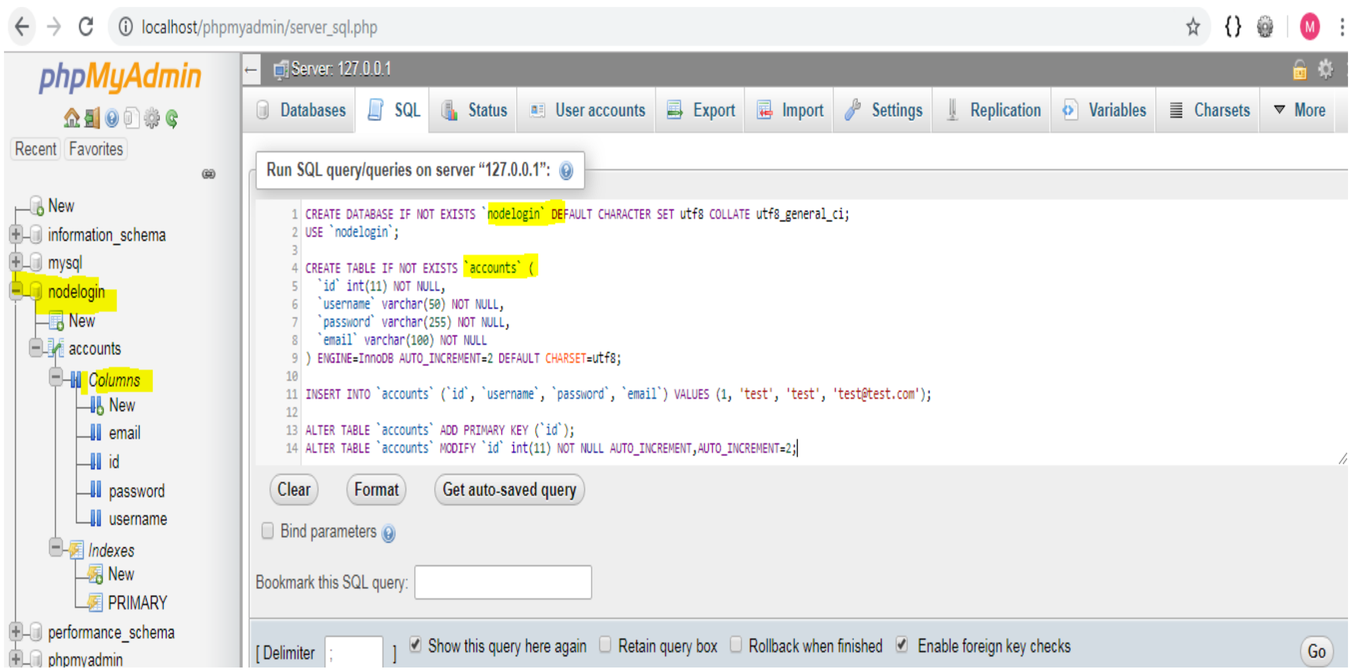
Copy the command below and navigate to your phpmyadmin dashboard and execute the following query in the console (usually found at the bottom of the page) in order to create database and table for the app.



```
3:47:42 PM  [mysql]   improper privileges, a crash, or a shutdown by another method.
3:47:42 PM  [mysql]   Press the Logs button to view error logs and check
3:47:42 PM  [mysql]   the Windows Event Viewer for more clues
3:47:42 PM  [mysql]   If you need more help, copy and post this
3:47:42 PM  [mysql]   entire log window on the forums
3:50:41 PM  [mysql]   Attempting to start MySQL app...
3:50:41 PM  [mysql]   Status change detected: running
3:51:24 PM  [Apache]  Attempting to start Apache app...
3:51:25 PM  [Apache]  Status change detected: running
```

Open MySQL admin tab

```sql
CREATE DATABASE IF NOT EXISTS `nodelogin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `nodelogin`;

CREATE TABLE IF NOT EXISTS `accounts` (
  `id` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;


INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES (1, 'test', 'test',
'test@test.com');


ALTER TABLE `accounts` ADD PRIMARY KEY (`id`);
ALTER TABLE `accounts` MODIFY `id` int(11) NOT NULL
AUTO_INCREMENT,AUTO_INCREMENT=2;
```



```
CREATE DATABASE IF NOT EXISTS `nodelogin` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `nodelogin`;

CREATE TABLE IF NOT EXISTS `accounts` (
  `id` int(11) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

INSERT INTO `accounts` (`id`, `username`, `password`, `email`) VALUES (1, 'test', 'test', 'test@test.com');

ALTER TABLE `accounts` ADD PRIMARY KEY (`id`);
ALTER TABLE `accounts` MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=2;
```

Adding the views

Creating the Login System
login.html

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Login Form Tutorial</title>
		<style>
		.login-form {
			width: 300px;
			margin: 0 auto;
			font-family: Tahoma, Geneva, sans-serif;
		}
		.login-form h1 {
			text-align: center;
			color: #4d4d4d;
			font-size: 24px;
			padding: 20px 0 20px 0;
		}
		.login-form input[type="password"],
		.login-form input[type="text"] {
			width: 100%;
			padding: 15px;
			border: 1px solid #dddddd;
			margin-bottom: 15px;
			box-sizing:border-box;
		}
		.login-form input[type="submit"] {
			width: 100%;
			padding: 15px;
			background-color: #535b63;
```

```
                        border: 0;
                        box-sizing: border-box;
                        cursor: pointer;
                        font-weight: bold;
                        color: #ffffff;
                }
                </style>
        </head>
        <body>
                <div class="login-form">
                        <h1>Login Form</h1>
                        <form action="auth" method="POST">
                                <input type="text" name="username" placeholder="Username" required>
                                <input type="password" name="password" placeholder="Password" required>
                                <input type="submit">
                        </form>
                </div>
        </body>
</html>
```

login.js Source

```
var mysql = require('mysql');
var express = require('express');
var session = require('express-session');
var bodyParser = require('body-parser');
var path = require('path');

var connection = mysql.createConnection({
        host     : 'localhost',
        user     : 'root',
        password : '',
		database : 'nodelogin'
});

var app = express();
app.use(session({
		secret: 'secret',
		resave: true,
		saveUninitialized: true
}));
app.use(bodyParser.urlencoded({extended : true}));
app.use(bodyParser.json());

app.get('/', function(request, response) {
		response.sendFile(path.join(__dirname + '/login.html'));
});

app.post('/auth', function(request, response) {
		var username = request.body.username;
		var password = request.body.password;
		if (username && password) {
				connection.query('SELECT * FROM accounts WHERE username = ? AND
password = ?', [username, password], function(error, results, fields) {
						if (results.length > 0) {
								request.session.loggedin = true;
								request.session.username = username;
								response.redirect('/home');
						} else {
								response.send('Incorrect Username and/or Password!');
						}
						response.end();
				});
		} else {
				response.send('Please enter Username and Password!');
				response.end();
```

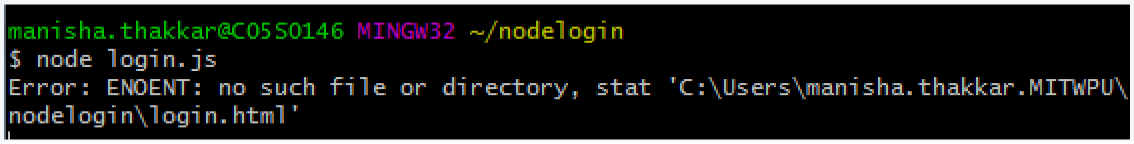
```
	}
});

app.get('/home', function(request, response) {
	if (request.session.loggedin) {
		response.send('Welcome back, ' + request.session.username + '!');
	} else {
		response.send('Please login to view this page!');
	}
	response.end();
});

app.listen(3000);
```

Our web application needs to listen on a port, for testing purposes we'll use port 3000:

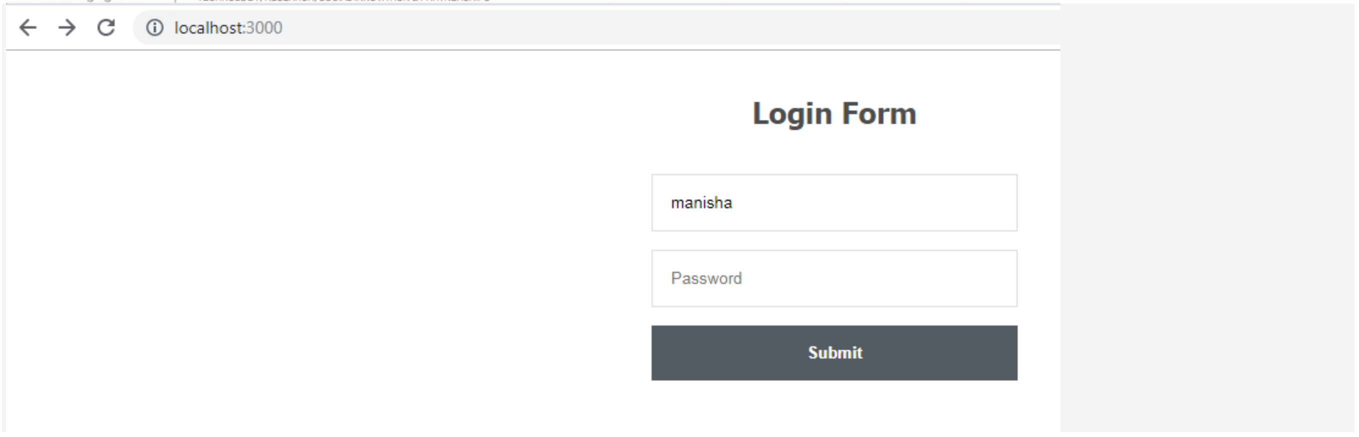
```
app.listen(3000);
```

To run our new web application we can run the following command:

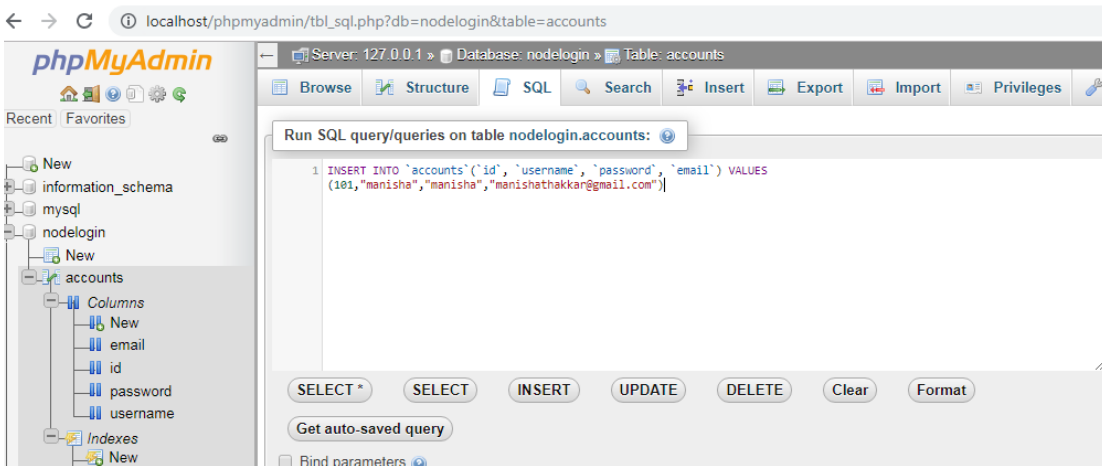
`node login.js` in command prompt/console, this will start the server, if we enter the address: `http://localhost:3000/` it should display our login form.



```
manisha.thakkar@C05S0146 MINGW32 ~/nodelogin
$ node login.js
Error: ENOENT: no such file or directory, stat 'C:\Users\manisha.thakkar.MITWPU\
nodelogin\login.html'
```


localhost:3000
Login Form
manisha
Password
Submit

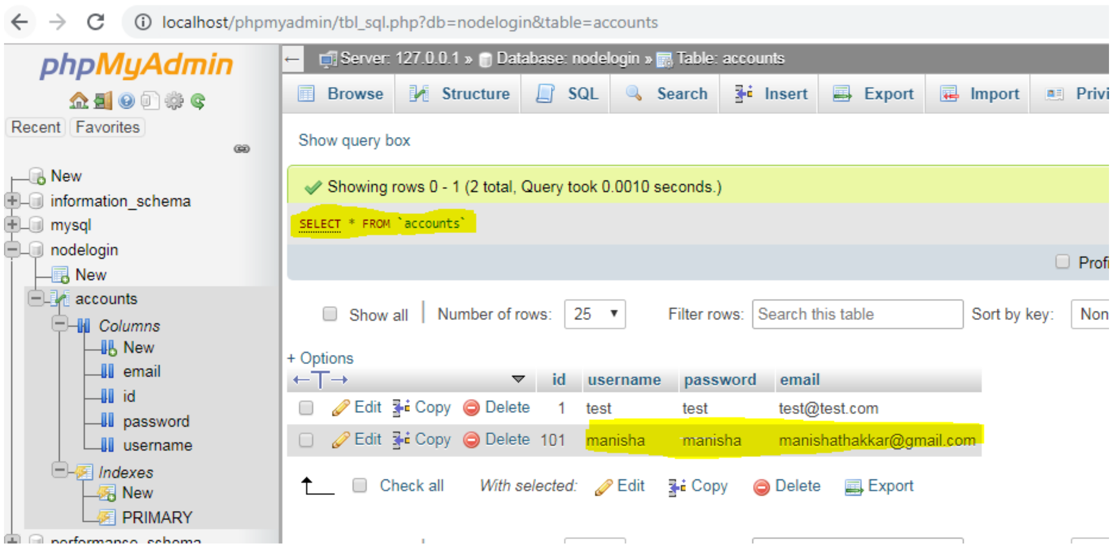
Insert row in table accounts



localhost/phpmyadmin/tbl_sql.php?db=nodelogin&table=accounts

Run SQL query/queries on table nodelogin.accounts:

```
INSERT INTO `accounts`(`id`, `username`, `password`, `email`) VALUES
(101,"manisha","manisha","manishathakkar@gmail.com")
```



localhost/phpmyadmin/tbl_sql.php?db=nodelogin&table=accounts

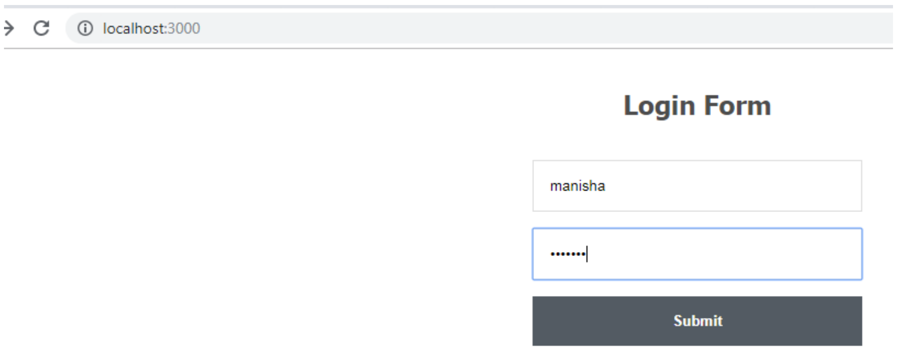
Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT * FROM `accounts`
```

Number of rows: 25

id	username	password	email
1	test	test	test@test.com
101	manisha	manisha	manishathakkar@gmail.com

Login with new user name and password

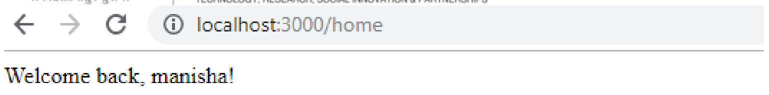


localhost:3000

Login Form

manisha

•••••••



Welcome back, manisha!