--Installation node.js

npm -init -y

npm install node.js

npm install prompt-sync

npm install mysql

create table student(s\_id int,s\_name varchar(20),s\_mobile varchar(10),s\_emailid(20));

ALTER USER 'your\_username'@'your\_host' IDENTIFIED WITH mysql\_native\_password BY 'your\_password'

Steps for 1-4:-

Steps-> create folder->open->create js files given->go to folder path and type cmd->press enter-> node main.js

Practical 1: **Node.js Modules**

Aim: Create an application to demonstrate Node.js Modules

**math.js**

module.exports = {

add : (a,b) => a+b,

subtract : (a,b) => a-b,

multiply : (a,b) => a\*b,

divide:(a,b) => a/b,

};

**String.js**

module.exports={

capitalize:(str) => str.charAt(0).toUpperCase() +str.slice(1),

reverse :(str) => str.split(' ').reverse().join(' '),

};

**Main.js**

const math = require(`./math`);

const string = require (`./string`);

const resultAdd = math.add(10,5);

console.log(`10 + 5 = ${resultAdd}`);

const resultMultiply = math.multiply(7,2);

console.log(`7 \* 2 = ${resultMultiply}`);

const inputString = 'hello world !';

const CapitalizedString = string.capitalize(inputString);

console.log(`Capitalized :${CapitalizedString}`);

const ReversedString = string.reverse(inputString);

console.log(`Reversed :${ReversedString}`);

Practical 2:Node.js Events(event emitter)

Aim: Create an application to demonstrate various Node.js Events

const prompt = require('prompt-sync')();

const EventEmitter = require('events');

const eventEmitter = new EventEmitter();

function performOperation(operator, num1, num2)

{

let result;

switch (operator)

{

case '+':

result = num1 + num2;

break;

case '-':

result = num1 - num2;

break;

case '\*':

result = num1 \* num2;

break;

case '/':

result = num1 / num2;

break;

default:

console.log('Invalid operator.');

return;

}

eventEmitter.emit('calculationResult', result);

}

eventEmitter.on('calculationResult', (result) => {

console.log('Result: '+result);

askForInput();

});

function askForInput()

{

const input = prompt('Enter expression (e.g., 54 + 96): ');

if(input == 'exit')

{

process.exit(0);

}

else

{

const [num1, operator, num2] = input.split(' ');

if (num1 && operator && num2)

{

performOperation(operator, parseFloat(num1), parseFloat(num2));

}

else

{

console.log('Invalid input. Please enter two numbers and an operator.');

askForlnput();

}

}

}

console.log('Simple Calculator');

console.log('Usage: Enter two numbers and an operator (+, -, \*, /) please dont put spaces or type (exit) to terminate');

askForInput();

**Practical 3:Node.js Functions**

Aim: Create an application to demonstrate Node.js Functions

const prompt = require('prompt-sync')();

function prime(num)

{

for(let i=2;i<num;i++)

{

if(num%i == 0)

{

return false;

}

}

return true;

}

function factorial(num)

{

if(num==0 || num ==1)

{

return 1;

}

let result = 1;

for(let i =2;i<=num;i++)

{

result\*=i;

}

return result;

}

function fibonacci(n){

let fibSeries = [0,1];

for(let i=2;i<n; i++)

{

fibSeries.push(fibSeries[i-1]+ fibSeries[i-2]);

}

return fibSeries;

}

 const num= parseInt(prompt('Enter number:'));

if(prime(num))

{

console.log(num+" is a prime number.");

}

else

{

console.log(num + " is not a prime number.");

}

const fact=factorial(num);

console.log("factorial of " + num+ " is:" +fact);

const fibSeries = fibonacci(num);

console.log("Fibonacci series up to " +num+ "terms:" +fibSeries.join(','));

**Practical 4:File Handling(CRUD Operation)**

Aim: Using File Handling demonstrate all basic file operations(Create, write,read,delete)

const fs = require('fs');

const prompt1 = require('prompt-sync')();

function createFile(fileName)

{

fs.writeFile(fileName, '',(error) => {

if(error)

{

console.log('error creating the file:', error);

}

else

{

console.log('File created Successfully');

}

}

);

}

function writeToFile(fileName,content)

{

fs.appendFile(fileName,content,(error) =>{

if(error)

{

console.log('Error writing to the file: ',error);

}

else

{

console.log('Content has been written to the file Successfullly');

}

});

}

function readFromFile(fileName)

{

fs.readFile(fileName, 'utf8',(error,data) =>{

if (error)

{

console.log('Error reading the file:',error);

}

else

{

console.log('content read from the file:');

console.log(data);

}

});

}

function deleteFile(fileName)

{

fs.unlink(fileName,(error) => {

if (error)

{

console.log('Error deleting the file:',error);

}

else

{

console.log('content has been delted successfully')

}

});

}

function main()

{

let fileName;

console.log("creating file press: 1");

console.log("writing file press: 2");

console.log("reading file press: 3");

console.log("deleting file press: 4");

console.log("exit file press: 5");

const choice = parseInt(prompt1('Enter an operation to be performed : '));

switch(choice)

{

case 1:

fileName = prompt('Enter file name to be created :');

createFile(fileName);

setTimeout(() => {main();}, 1000);

break;

case 2:

fileName = prompt('Enter file name in which data should be added:');

const content = prompt('Enter content to write in a file:');

writeToFile(fileName,content);

setTimeout(() => {main();},1000);

break;

case 3:

fileName = prompt('Enter file name to be read:');

readFromFile(fileName);

setTimeout(() => {main();},1000);

break;

case 4:

fileName = prompt('Enter file name to be created: ');

deleteFile(fileName);

setTimeout(() => {main();}, 1000);

break;

case 5:

console.log('exiting process');

process.exit(0);

break;

default:

console.log('Invalid choice');

}

}

main();

**Practical 5:**

Aim: Create an HTTP Server and perform operations on it.

Steps: Create folder->create Http.js and Https-server.html->

**Http.js**

const http = require('http');

const url = require('url');

const prompt = require('prompt-sync')();

function calculateGCD(num1, num2) {

while (num2 !== 0) {

const temp = num2;

num2 = num1 % num2;

num1 = temp;

}

return num1;

}

function reverseNumber(num) {

let reversed = 0;

while (num !== 0) {

reversed = reversed \* 10 + num % 10;

num = Math.floor(num / 10);

}

return reversed;

}

const server = http.createServer((req, res) => {

const queryObject = url.parse(req.url, true).query;

const num1 = parseInt(queryObject.num1);

const num2 = parseInt(queryObject.num2);

const gcd = calculateGCD(num1, num2);

const reversedNum1 = reverseNumber(num1);

const reversedNum2 = reverseNumber(num2);

const htmlResponse = `

<h1><center>Welcome to the Result Page</h1>

<h2>The Result is:</h2>

<p>GCD of ${num1} and ${num2} is: ${gcd}</p>

<p>Reverse of ${num1} is: ${reversedNum1}</p>

<p>Reverse of ${num2} is: ${reversedNum2}</p>

`;

res.writeHead(200, { 'Content-Type': 'text/html' });

res.end(htmlResponse);

});

const port = 3000;

server.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

**Http-server.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Calculate GCD and Reverse Numbers</title>

</head>

<body>

<h1>Enter two numbers:</h1>

<form action="http://localhost:3000" method="GET">

<label for="num1">Number 1:</label>

<input type="number" id="num1" name="num1" required><br><br>

<label for="num2">Number 2:</label>

<input type="number" id="num2" name="num2" required><br><br>

<button type="submit">Calculate</button>

</form>

</body>

</html>

Practical 6:

Aim: Create an application to establish a connection with the MySQL database and perform basic database operations on it

npm install mysql

db.js

const mysql = require('mysql');

const connection = mysql.createConnection ({

host: 'localhost',

user: 'root',

password: 'hellonodejs',

database:'prac6',

});

connection.connect((err)=>{

if(err){

console.error(' errorconnecting to mysql:' + err.stack);

return;}

console.log(' connected to my sql as id ' + connection.threadID

);});

module.exports= connection;

**Delete.js:**

const db= require('./db');

const s\_id = 1;

db.query('DELETE FROM student WHERE s\_id=?' ,s\_id, (err, results)=>{

if (err){

console.error(' error deleting data :' + err);

process.exit (0);}

else{

console.log ('Data deleted :',results.affectedRows);

process.exit(0);

}

});

**Insert.js:**

const db = require('./db');

const newData={

s\_id:1,

s\_name:'rohit yadav',

s\_mobile: '9895463278',

s\_emailid:'rohit@gmail.com'}

db.query (' INSERT INTO student SET ? ',newData , (err,results)=>{

if (err){

console.error('error inserting data:' + err);

process.exit(0);

}

else{

console.log ('Data inserted.ID:',results.insertId);

process.exit(0);

}

});

**Insert.test:**

function insert\_student(){

const prompt = require('prompt-sync')();

st\_id=prompt('enter the student id:');

st\_name= prompt ('enter the student name :');

st\_mobile= prompt ('enter the mobile number :');

st\_emailid= prompt ('enter the email id :');

const newData={

s\_id:st\_id,

s\_name:st\_name,

s\_mobile:st\_mobile,

s\_emailid:st\_emailid};

const db = require('./db');

db.query('insert into student SET ?',newData, (err,results)=>{

if (err){

console.error('error inserting data:' + err);

process.exit(0);

}

else{

console.log ('Data inserted.ID:',results.insertId);

process.exit(0);

}

});

}

//module.exports = { insert\_student,};

insert\_student();

**Select.js**

const db= require('./db');

db.query('SELECT \* FROM student' , (err, results)=>{

if (err){

console.error(' error selecting data :' +

err);

process.exit (0);}

else{

console.log ('selected data:',results);

process.exit(0);

}

});

**Update.js:**

const db= require('./db');

const updatedData = { s\_name:'raj yadav',

s\_emailId: ' raj@gmai.com',};

const studentid = 1;

db.query('UPDATE student SET ? WHERE s\_id=?' ,[updatedData,studentid], (err, results)=>{

if (err){

console.error(' error updating data :' + err);

process.exit (0);}

else{

console.log ('Data updated:',results.affectedRows);

process.exit(0);

}

});

Practical 7:

Aim: Create an application using Filters

7.1

<html>

<head>

<title>Angular JS Filters</title>

<script src = "angular.min.js">

</script>

</head>

<body>

<h2>AngularJS Sample Application</h2>

<div ng-app = "mainApp" ng-controller = "studentController">

<table border = "0">

<tr>

<td>Enter first name:</td>

<td><input type = "text" ng-model = "student.firstName"></td>

</tr>

<tr>

<td>Enter last name: </td>

<td><input type = "text" ng-model = "student.lastName"></td>

</tr>

<tr>

<td>Enter fees: </td>

<td><input type = "text" ng-model = "student.fees"></td>

</tr>

<tr>

<td>Enter subject: </td>

<td><input type = "text" ng-model = "subjectName"></td>

</tr>

</table>

<br/>

<table border = "0">

<tr>

<td>Name in Upper Case: </td><td>{{student.fullName() | uppercase}}</td>

</tr>

<tr>

<td>Name in Lower Case: </td><td>{{student.fullName() | lowercase}}</td>

</tr>

<tr>

<td>fees: </td><td>{{student.fees | currency: " RS " : a}}

</td>

</tr>

<tr>

<td>Subject:</td>

<td>

<ul>

<li ng-repeat = "subject in student.subjects | filter: subjectName |orderBy:'marks'">

{{ subject.name + ', marks:' + subject.marks }}

</li>

</ul>

</td>

</tr>

</table>

</div>

<script>

var mainApp = angular.module("mainApp", []);

mainApp.controller('studentController', function($scope) {

$scope.student = {

firstName: "Shivesh",

lastName: "singh",

fees:158970,

subjects:[

{name:'AJAVA',marks:80},

{name:'ADBMS',marks:86},

{name:'MATHS',marks:67}

],

fullName: function() {

var studentObject;

studentObject = $scope.student;

return studentObject.firstName + " " + studentObject.lastName;

}

};

});

</script>

</body>

</html>

**7.2:**

**<!doctype html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>Example - example-number-filter-production</title>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>**

**</head>**

**<body ng-app="numberFilterExample">**

**<script>**

**angular.module('numberFilterExample', [])**

**.controller('ExampleController', ['$scope', function($scope) {**

**$scope.val = 1234.56789;**

**}]);**

**</script>**

**<div ng-controller="ExampleController">**

**<label>Enter number: <input ng-model='val'></label><br>**

**Default formatting: <span id='number-default'>{{val | number}}</span><br>**

**No fractions: <span>{{val | number:0}}</span><br>**

**Negative number: <span>{{-val | number:4}}</span>**

**</div>**

**</body>**

**</html>**

**7.3:**

**<!DOCTYPE html>**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>**

**AngularJs Date filter Example**

**</title>**

**<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>**

**<script>**

**var app = angular.module("AngulardateApp", []);**

**app.controller("datectrl", function ($scope) {**

**$scope.sampledate = new Date();**

**});**

**</script>**

**</head>**

**<body ng-app="AngulardateApp">**

**<div ng-controller="datectrl">**

**Enter Number: <input type="text" ng-model="sampledate" style="width:400px" /><br /><br />**

**Date with short expression:{{sampledate | date:"short" }}<br /><br />**

**Date with mediumDate expression: {{sampledate | date : "mediumDate"}} <br /><br />**

**Date with yyyy-mm-dd hh:mm:ss expression: {{sampledate | date : "MMM,dd yyyy**

**hh:mm:ss" : 0}} <br /><br />**

**Date with yyyy-mm-dd hh:mm:ss expression: {{sampledate | date : "dd/MMM/yyyy 'at'**

**hh:mma" : 0}}**

**</div>**

**</body>**

**</html>**

**7.4:**

**Angularjs-4.html**

**<!DOCTYPE html>**

**<html ng-app="filterApp">**

**<head>**

**<meta charset="UTF-8">**

**<title>Custom Filter App</title>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>**

**<script src="app.js"></script>**

**</head>**

**<body ng-controller="FilterController">**

**Enter firstname: <input type="text" ng-model="firstname" placeholder="Enter**

**firstname"><br/><br/>**

**Enter lastname: <input type="text" ng-model="lastname" placeholder="Enter**

**lastname"><br/><br/>**

**<button ng-click="convertText('check')">check</button>**

**<button ng-click="convertText('uppercase')">Convert to Uppercase</button>**

**<button ng-click="convertText('lowercase')">Convert to Lowercase</button>**

**<p>You have entered: {{ yourtext }}</p>**

**<p>Converted Text: {{fullname }}</p>**

**</body>**

**</html>**

**App.js**

**var app = angular.module('filterApp', []);**

**app.controller('FilterController', function($scope)**

**{**

**$scope.firstname = '';**

**$scope.lastname = '';**

**$scope.fullname = '';**

**$scope.yourtext='';**

**$scope.convertText = function(type)**

**{**

**if (type === 'uppercase')**

**{**

**$scope.fullname = $scope.firstname.toUpperCase() + ' ' +**

**$scope.lastname.toUpperCase();**

**}**

**else if (type === 'lowercase')**

**{**

**$scope.fullname = $scope.firstname.toLowerCase() + ' ' +**

**$scope.lastname.toLowerCase();**

**}**

**else**

**{**

**$scope.yourtext = $scope.firstname + ' ' + $scope.lastname;**

**}**

**};**

**});**

**7.5**

**custom.filter.html:**

**<!DOCTYPE html>**

**<html lang="en" ng-app="filterApp">**

**<head>**

**<meta charset="UTF-8">**

**<title>Custom Filter App</title>**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>**

**<script src="custom.js"></script>**

**</head>**

**<body ng-controller="FilterController">**

**<input type="text" ng-model="inputText" placeholder="Enter text">**

**<p>Filtered Output: {{ inputText | customFilter }}</p>**

**</body>**

**</html>**

**Custom.js:**

**// Define a new AngularJS module**

**var app = angular.module('filterApp', []);**

**app.filter('customFilter', function() {**

**return function(input) {**

**return input.toUpperCase();**

**};**

**});**

**// Create a controller for the module**

**app.controller('FilterController', function($scope) {**

**$scope.inputText = '';**

**});**

**Practical 8:**

Aim: Create an application to demonstrate directives

**8.1.Even-odd:**

<!DOCTYPE html>

<html lang="en" ng-app="evenOddApp">

<head>

<meta charset="UTF-8">

<title>Even Odd Color Change</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>

<style>

.even {

background-color: green;

}

.odd {

background-color: red;

}

</style>

</head>

<body>

<div ng-controller="MainController">

<label for="numberInput">Enter a number:</label>

<input type="text" id="numberInput" ng-model="inputNumber" ng-class="{ 'even':

isEven(), 'odd': !isEven() }">

<button ng-click="checkNumber()">Check Number</button>

</div>

<script>

angular.module('evenOddApp', [])

.controller('MainController', function ($scope) {

$scope.inputNumber = '';

$scope.checkNumber = function () {

// Convert the input to a number

var number = parseInt($scope.inputNumber);

// Check if the number is even

$scope.isEven = function () {

return number % 2 === 0;

};

};

});

</script>

</body>

</html>

8.2.Prime-number:

<!DOCTYPE html>

<html lang="en" ng-app="primeApp">

<head>

<meta charset="UTF-8">

<title>Prime Number Message</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>

<style>

.prime {

color: green;

}

.not-prime {

color: red;

}

</style>

</head>

<body>

<div ng-controller="MainController">

<label for="numberInput">Enter a number:</label>

<input type="text" id="numberInput" ng-model="inputNumber">

<button ng-click="checkNumber()">Check Number</button>

<div ng-if="isPrime()" ng-class="{ 'prime': isPrime(), 'not-prime': !isPrime() }">

{{ resultMessage }}

</div>

<div ng-if="!isPrime()" ng-class="{ 'prime': isPrime(), 'not-prime': !isPrime() }">

{{ resultMessage }}

</div>

</div>

<script>

angular.module('primeApp', [])

.controller('MainController', function ($scope) {

$scope.inputNumber = '';

$scope.resultMessage = '';

$scope.checkNumber = function () {

// Convert the input to a number

var number = parseInt($scope.inputNumber);

// Check if the number is prime

$scope.isPrime = function ()

{

if (number <= 1) {

$scope.resultMessage = 'The number is not a prime number';

return false;

}

for (var i = 2; i <number; i++)

{

if (number % i === 0)

{

$scope.resultMessage = 'The number is not a prime number';

return false;

}

}

$scope.resultMessage = 'The number is a prime number';

return true;

};

};

});

</script>

</body>

</html>

**Practical 9:**

**Aim: Demonstrate controllers in Anjular.js through an application**

**9.1. Shopping cart:**

**<!DOCTYPE html>**

**<html ng-app="shoppingCartApp">**

**<head>**

**<meta charset="UTF-8">**

**<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>**

**</head>**

**<body style="background-color:white;">**

**<div ng-controller="ShoppingCartController">**

**<h1 style="font-size:300%; color:aquamarine">Shopping cart</h1>**

**<div ng-repeat="product in products">**

**<p>**

**{{ product.name }} - ₹{{ product.price }}**

**<button ng-click="addToCart(product)">&#128722;</button>**

**</p>**

**</div>**

**<!-- Your Cart -->**

**<h1 style="color:blue;">Your Cart</h1>**

**<ul>**

**<li ng-repeat="item in cartItems">**

**{{ item.product.name }} -**

**Quantity:**

**<button ng-click="decrementQuantity(item)">-</button>**

**<input type="number" ng-model="item.quantity" min="1" ngchange="updateCart(item)" />**

**<button ng-click="incrementQuantity(item)">+</button>**

**- ₹{{ item.product.price \* item.quantity }}**

**<button ng-click="removeFromCart(item)">&#128465;</button>**

**</li>**

**</ul>**

**<p>Total: ₹{{ getTotal() }}</p>**

**</div>**

**<script>**

**var app = angular.module('shoppingCartApp', []);**

**app.controller('ShoppingCartController', function ($scope) {**

**$scope.products = [**

**{ id: 1, name: 'Chips', price: 50 },**

**{ id: 2, name: 'Milk', price: 20 },**

**{ id: 3, name: 'Bisleri', price: 20 },**

**{ id: 4, name: 'Coldrinks', price: 60 },**

**];**

**$scope.cartItems = [];**

**$scope.addToCart = function (product) {**

**var found = false;**

**// Check if the product is already in the cart**

**for (var i = 0; i < $scope.cartItems.length; i++) {**

**if ($scope.cartItems[i].product.id === product.id) {**

**$scope.cartItems[i].quantity++;**

**found = true;**

**break;**

**}**

**}**

**// If the product is not in the cart, add it**

**if (!found) {**

**$scope.cartItems.push({**

**product: product,**

**quantity: 1**

**});**

**}**

**};**

**$scope.removeFromCart = function (item) {**

**var index = $scope.cartItems.indexOf(item);**

**if (index !== -1) {**

**$scope.cartItems.splice(index, 1)**

**}**

**};**

**$scope.updateCart = function (item) {**

**if (item.quantity < 1) {**

**item.quantity = 1; // Set a minimum quantity**

**}**

**};**

**$scope.incrementQuantity = function (item) {**

**item.quantity++;**

**};**

**$scope.decrementQuantity = function (item) {**

**if (item.quantity > 0) {**

**item.quantity--;**

**}**

**};**

**$scope.getTotal = function () {**

**var total = 0;**

**for (var i = 0; i < $scope.cartItems.length; i++) {**

**total += $scope.cartItems[i].product.price \* $scope.cartItems[i].quantity;**

**}**

**return total;**

**};**

**});**

**</script>**

**</body>**

**</html>**

**9.2.Factor\_finder:**

<html ng-app="factorFinderApp">

<head>

<meta charset="UTF-8">

<title>AngularJS Factor Finder</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>

</head>

<body>

<div ng-controller="FactorFinderController">

<h1>Factor Finder</h1>

<label for="numberInput">Enter a number:</label>

<input type="number" id="numberInput" ng-model="inputNumber" required>

<button ng-click="findFactors()">Find Factors</button>

<div ng-if="factors.length > 0">

<p>Factors of {{ number }} are: {{ factors.join(', ') }}</p>

</div>

</div>

<script>

var app = angular.module('factorFinderApp', []);

app.controller('FactorFinderController', function($scope) {

$scope.inputNumber = undefined;

$scope.number = '';

$scope.factors = [];

$scope.findFactors = function() {

$scope.factors = []; // Reset factors on each button click

if ($scope.inputNumber <= 0) {

alert("Please enter a positive integer greater than 0.");

return;

}

for (var i = 1; i <= $scope.inputNumber; i++) {

if ($scope.inputNumber % i === 0) {

$scope.factors.push(i);

}

$scope.number=$scope.inputNumber;

}

};

});

</script>

</body></html>

**Practical 10:**

**Aim: Demonstrate features of Angular.js forms with a perform**

<!DOCTYPE html>

<html ng-app="myApp">

<head>

<title>My Forms using AngularJS</title>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

<style>

.error {

color: red;

}

</style>

</head>

<body>

<div ng-controller="myController">

<h2 align="center">Kindly Fill the Form</h2>

<form name="myForm" ng-submit="submitForm()" novalidate>

<table align="center">

<tr>

<td><label for="username">Username:</label></td>

<td>

<input type="text" id="username" name="username" ng-model="user.username"

ng-pattern="/^[a-zA-Z0-9]\*$/" required>

<span class="error" ng-show="myForm.username.$error.pattern">Invalid characters in

username.</span>

</td>

</tr>

<tr>

<td><label for="email">Email:</label></td>

<td>

<input type="text" id="email" name="email" ng-model="user.email"

ng-pattern="/^[^\s@]+@[^\s@]+\.[a-zA-Z0-9]{2,}$/" required>

<span class="error" ng-show="myForm.email.$error.pattern">Invalid email address

format.</span>

</td>

</tr>

<tr>

<td><label for="password">Password (at least 8 characters):</label></td>

<td>

<input type="password" id="password" name="password" ng-model="user.password"

ng-pattern="/^.{8,}$/" required>

<span class="error" ng-show="myForm.password.$error.pattern">Password must be at

least 8 characters.</span>

</td>

</tr>

<tr>

<td colspan="2" style="text-align: center;">

<button type="submit" ng-disabled="myForm.$invalid">Submit</button>

</td>

</tr>

</table>

</form>

<div ng-show="submitted">

<h3>You have submitted</h3>

<p>Username: {{user.username}}</p>

<p>Email: {{user.email}}</p>

<p>Password: {{user.password}}</p>

</div>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myController', function($scope) {

$scope.user = {};

$scope.submitted = false;

$scope.submitForm = function() {

$scope.submitted = true;

};

});

</script>

</body>

</html>