**A Skill Oriented Course Mean Stack Technologies report** Submitted in the partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology In

Computer Science and Engineering by

## Laghuvarapu Jyothi Prakash

## 21761A05A2

Under the guidance of Dr. K. Devi Priya,

Associate Professor, Dept. of CSE



## Computer Science and Engineering

**Lakireddy Bali Reddy College of Engineering (Autonomous) Accredited by NAAC with ‘A’ Grade & ISO 9001:2015 Certified Institution Approved by AICTE, New Delhi, and Affiliated to JNTUK, Kakinada**

**2023-24**

# LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

**(AUTONOMOUS)**

**Accredited by NAAC with ‘A’ Grade & ISO 9001:2015 Certified Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada**

**L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.**

[hodcse@lbrce.ac.in,](mailto:hodcse@lbrce.ac.in) [cseoffice@lbrce.ac.in,](mailto:cseoffice@lbrce.ac.in) Phone: 08659-222 933, Fax: 08659-222931

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Certificate**

This is to certify that the **Mean Stack Technologies Module-III (20CSS3**) project entitled “**FREEMER**” is being submitted by **Laghuvarapu Jyothi Prakash(21761A05A2)** in partial fulfilment for the award of B. Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Kakinada is a record of bonafide work carried out by him/her under our guidance.

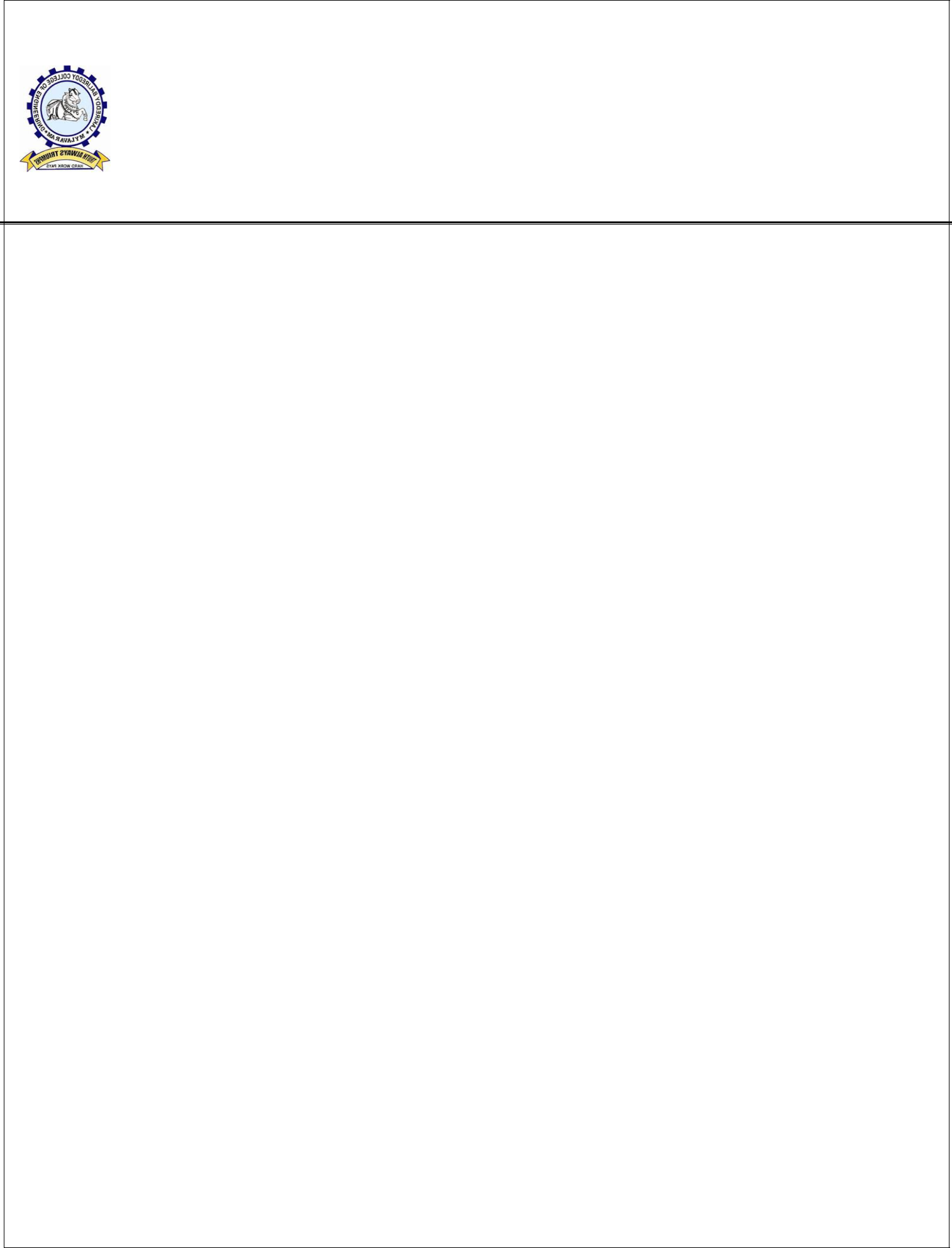
The results embodied in this Skill Oriented Course Mean Stack Technologies / Project report have not been submitted to any other University or Institute for the award of any degree or diploma.

### Project Guide

Dr. K. Devi Priya

Associate Professor.

**Head of the department** Dr. D. Veeraiah, Professor.



**External Examiner**

### ACKNOWLEDGEMENT

I would like to thank **Dr. K. Devi Priya,** Associate Professor**,** CSE department for the encouraging and support in carrying out this Skill Oriented Course Mean Stack Technologies.

I also take the privilege to record my thanks to **Dr. D. Veeraiah,** Professor, Head of the Department of CSE whose encouragement, cooperation and valuable support crown my success.

I express my thanks to the support given by management in completing my Mean Stack Technologies / Project. I also express my sincere gratitude & deep sense of respect to the Principal, **Dr. K. Appa Rao** for making us available all the required assistance and his support and inspiration to carry out this Skill Oriented Course Mean Stack Technologies / Project in the Institute.

I am thankful to the teaching and non-teaching staff of CSE department for their direct as well as indirect help in my Skill Oriented Course Mean Stack Technologies / Project.

I am elated to avail my selves to this opportunity to express my deep sense of gratitude to my parents.

### ABSTRACT

The "Adopt-a-Farmer" project represents a groundbreaking approach to promoting sustainable agriculture and empowering rural farming communities. This initiative transcends conventional boundaries, fostering a unique partnership between urban stakeholders and rural farmers.

Our project offers individuals the opportunity to engage directly with the farming community. By adopting a real farmer's land and crops, participants gain an immersive understanding of agricultural practices. This program provides a platform for clients to actively contribute to the cultivation process, allowing them to participate and learn firsthand.

Farmers stand to benefit significantly from this partnership. They receive not only financial support but also access to modern technology and knowledge exchange. The collaboration facilitates the dissemination of improved farming techniques and crop management, potentially leading to increased agricultural productivity.

The "Adopt-a-Farmer" initiative leverages technology to bridge geographical and knowledge gaps, facilitating a sense of community and shared responsibility. Our objective is to rekindle interest in agriculture among urban populations, promote sustainable farming practices, and empower both clients and farmers towards achieving agricultural excellence.

### CONTENTS

**CONCEPTS Page No**

1. Introduction 6
2. Modules 7
3. Technologies 8-27
   1. HTML5 8-9
   2. CSS 10-11
   3. Java Script 12-17
   4. Node JS 18-19
   5. Express JS 20-21
   6. MongoDB 22-25
   7. Type Script 26-27
4. Implementation (For Developed Application) 28-78
   1. HTML Code 28-60
   2. List of Database Tables 60-61
   3. Node & Express Code 61-68
5. Screen Shots 69-78
6. Conclusion 79
7. Web References 80
8. Self-Assessment Certificates 81-83

### INTRODUCTION

Blood Bank Management is an essential component of healthcare systems worldwide. It encompasses a range of activities aimed at efficiently collecting, storing, and displaying the details of patients in need of blood transfusions. To streamline the blood donation process and ensure the availability of safe blood, a comprehensive blood bank management system is crucial. This abstract provides an overview of the key functionalities and features of such a system.

The donor registration module allows individuals to register as blood donors, providing their personal information, blood group, and contact details. The system ensures the confidentiality and security of donor data, creating a sense of trust and encouraging participation. Once registered, donors can easily schedule appointments for blood donation and receive timely reminders, facilitating a steady supply of blood.

Efficient blood collection is facilitated through the dedicated module within the system. It tracks the collection of blood units from registered donors, categorizing them based on blood type. This categorization enables the system to match specific blood types with requests made by hospitals and medical professionals, ensuring a prompt response to patients' needs.

The blood request management feature enables hospitals to place requests for specific blood types and quantities. By utilizing the centralized platform, healthcare facilities can communicate their requirements effectively, streamlining the process of obtaining suitable blood units. This module acts as a bridge between the blood bank and hospitals, enabling efficient collaboration for the timely and accurate distribution of blood.

Furthermore, the blood bank management system incorporates reporting and analytics capabilities. These functionalities generate detailed reports and provide valuable insights into donor statistics, blood inventory, and usage patterns. By monitoring and evaluating these metrics, blood banks can identify areas for improvement and make informed decisions to optimize their operations.

In conclusion, a blood bank management system plays a vital role in optimizing the blood donation and distribution process. By providing a centralized platform for collaboration and streamlining activities such as donor registration, blood collection, blood request management, and reporting, it enhances operational efficiency, promotes transparency, and improves patient care. With the power of technology, healthcare organizations can effectively manage their blood banks, ensuring a steady supply of safe blood for those in need.

# Modules

## The contents of website are:-

* Home
* Blood Needs
* Acceptable Types
* Blood Donation
* Login
* Signup
* CRUD

**Home module:** This page allow users to view the home page which contain details about theservices offered by this mobile website via login or signup.

**Blood Needs:** Users will able to see blood needs in that hospital.

**Acceptable Types:** Users can see the blood types which can be acceptable at the hospital.

**Blood Donation:** This module gives only access to the admin to modify the donations database.

**Signup module:** This page will allow the users to signup to this website.

**Login module:** This page will allow the users to login if we signup already before.

**CRUD module:** It consists of insert, update, delete and displays the data which was storedin mongodb database.

# Technologies Used

## HTML (HyperText Markup Language)

It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between web pages. A markup language is used to define the text document within the tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g., HTML) are human readable. The language uses tags to define what manipulation has to be done on the text.

HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1995.

**Elements and Tags:** HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end of the page.

**HTML page structure:** The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e., doctype declaration, HTML, head, title, and body elements) upon which all web pages are created. :

**<HTML>** This is called the HTML root element. All other elements are contained within it.

**<HEAD>:** The head tag contains the “behind the scenes” elements for webpage. Elements within the head aren’t visible on the front-end of a webpage. HTML elements used inside the<HEAD> element include

**<style>-**This html tag allows us to insert styling into our webpages and make them appealing to look at with the help of CSS.

**<title>-**The title is what is displayed on the top of your browser when you visit a website and

contains title of the webpage that you are viewing.

**<base>-**It specifies the base URL for all relative URLs in a document.

**<script>-**This tag is used to add functionality in the website with the help of JavaScript.

**<body>:** The body tag is used to enclose all the visible content of a webpage. In other words,the body content is what the browser will show on the front-end.

An HTML document can be created using any text editor. Save the text file using

**.html** or

**.htm**.Once saved as an HTML document, the file can be opened as a webpage in the browser.



## Cascading Style Sheets

CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describeshow a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, css uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

**Why CSS?**

* + **CSS saves time:** You can write CSS once and reuse the same sheet in multiple HTML pages.
  + **Easy Maintenance:** To make a global change simply change the style, and all elements in all the webpages will be updated automatically.
  + **Search Engines:** CSS is considered a clean coding technique, which means search engines won’t have to struggle to “read” its content.
  + **Superior styles to HTML:** CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
  + **Offline Browsing:** CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

**CSS Syntax:**

CSS comprises style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule set consists of a selector and declaration block

**Selector – h1**

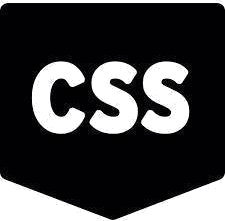
**Declaration – {color:blue;font size:12px;}**

* + The selector points to the HTML element you want to style.
  + The declaration block contains one or more declarations separated by semicolons.
  + Each declaration includes a CSS property name and a value, separated by a colon.

For Example:

color is property and blue is value. font-size is property and 12px is value.

* + CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly



## Java Script

JavaScript is used to create client-side dynamic pages.

JavaScript is an object-based scripting language which is lightweight and cross-platform.

JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

**JavaScript** (**js**) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

* + All popular web browsers support JavaScript as they provide built-in execution environments. JavaScript follows the syntax and structure of the C programming.
  + JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
  + JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
  + It is a light-weighted and interpreted language.
  + It is a case-sensitive language.
  + JavaScript is supportable in several operating systems including, Windows, macOS, etc.
  + It provides good control to the users over the web browsers.

Application of JavaScript

JavaScript is used to create interactive websites. It is mainly used for:

Client-side validation, Dynamic drop-down menus, Displaying date and time, Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialogbox), Displaying clocks etc.



## Lab Experiments:-

### Identifiers:

<html>

<head>

<script>

var radius; const PI=3.14;

radius=parseInt(window.prompt("Enter radius value:","10")); document.write("<h2 style='color:red'>"); document.write("Given Radius"+radius); document.write("</h2>");

var result=PI\*radius\*radius; document.write("<h2 style='color:green'>"); document.write("Area of the circle is ="+result); document.write("</h2>");

</script>

</head>

</html>

### Data Types:

<!DOCTYPE html>

<html>

<head>

<title>Movie Details</title>

<style>

div#maincontent { height: 100px; width: 500px;

border: 1px solid #CEE2FA; text-align: left;

color: #08438E; font-family: calibri; font-size: 20; padding: 5px;

}

div#heading {

text-decoration: bold; text-align: center; margin-top: 80px; width: 500px;

border: 1px solid #CEE2FA; text-align: center;

color: #08438E;

background-color: #CEE2FA; font-family: calibri;

font-size: 20; padding: 5px;

}

</style>

</head>

<body>

<center>

<div id="heading">Movie Details</div>

<div id="maincontent">

<script>

let movie="Twilight"; let lang="English"; var rat=4.5;

document.write("Movie:"+movie+"<br>"); document.write("Language:"+lang+"<br>"); document.write("Ratings:"+rat);

</script>

</div>

</center>

</body>

</html>

### Operators:

<!DOCTYPE html>

<html>

<head>

<title>Ticket Details</title>

<style>

div#maincontent { height: 150px; width: 500px;

border: 1px solid #CEE2FA; text-align: left;

color: #08438E; font-family: calibri; font-size: 20; padding: 5px;

}

div#heading {

text-decoration: bold; text-align: center; margin-top: 80px; width: 500px;

border: 1px solid #CEE2FA; text-align: center;

color: #08438E;

background-color: #CEE2FA; font-family: calibri;

font-size: 20; padding: 5px;

}

h4 {

padding: 0;

margin: 0;

}

</style>

</head>

<body>

<center>

<div id="heading">

<b>Theatre Drama</b>

</div>

<div id="maincontent">

<h4>Your Ticket Details:</h4>

<br>

<script>

let tottick=3; let tick=9; let disc=10;

let td=tottick\*tick;

let ad=td-((td\*disc)/100);

document.write("Total number of seats booked: $"+td+"<br>"); document.write("Festive season discount is:"+disc+"%"+"<br>"); document.write("Total cost after discount is: $"+ad);

</script>

</div>

</center>

</body>

</html>

### Statements and Expressions:

<!DOCTYPE html>

<html>

<head>

<title>Booking Details</title>

<style>

div#maincontent { height: 200px; width: 600px;

border: 1px solid #CEE2FA; text-align: left;

color: #08438E; font-family: calibri; font-size: 20; padding: 5px;

}

div#heading {

text-decoration: bold; text-align: center; margin-top: 80px; width: 600px;

border: 1px solid #CEE2FA; text-align: center;

color: #08438E;

background-color: #CEE2FA; font-family: calibri;

font-size: 20; padding: 5px;

}

h4 {

padding: 0;

margin: 0;

}

</style>

</head>

<body>

<center>

<div id="heading">

<b>Theatre Drama</b>

</div>

<div id="maincontent">

<h4>Your Ticket Details:</h4>

<br>

<script>

// Write the code to display the total price and discounted price of tickets let seats=4;

var cost=9;

let c1,c2,tc,c3,c4; if(seats<=2)

{

cost=9; tc=seats\*cost;

document.write("For "+seats+", you need to pay:"+tc);

}

else if(seats>=5)

{

document.write("Booking is not allowed");

}

else if(seats>2 && seats<5)

{

tc=seats\*9;

c1=cost-((cost\*5)/100); c2=cost-((cost\*7)/100); c3=cost-((cost\*9)/100); c4=cost-((cost\*11)/100); let tot=c1+c2+c3+c4;

document.write("Ticket for Customer1 gets 5% discount!, Cost is:

$"+c1+"<br>");

document.write("Ticket for Customer2 gets 7% discount!, Cost is:

$"+c2+"<br>");

document.write("Ticket for Customer3 gets 9% discount!, Cost is:

$"+c3+"<br>");

document.write("Ticket for Customer4 gets 11% discount!, Cost is:

$"+c4+"<br>");

document.write("For "+seats+", you need to pay:$"+tot+"instead of $"+tc);

}

</script>

</div>

</center>

</body>

</html>

## Node Js

Node.js is an open-source, cross-platform runtime environment that built on top of Chrome's V8 JavaScript engine. It uses an asynchronous, event-driven model that makes it lightweight and efficient. Node.js is often used for building real-time applications, such as chat applications, web servers, and data streaming applications.

### Here are some of the key features of Node.js:

* + Asynchronous programming: Node.js uses asynchronous programming, which means that it can handle multiple requests at the same time without blocking. This makes it ideal for real-time applications.
  + Event-driven model: Node.js uses an event-driven model, which means that it can respond to events as they occur. This makes it very efficient and scalable.
  + JavaScript: Node.js is written in JavaScript, which makes it a very easy language to learn and use.

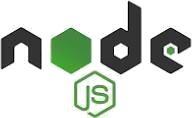
Open source: Node.js is open source, which means that it is free to use and modify.

### Here are some of the benefits of using Node.js:

* + Scalable: Node.js is very scalable, which means that it can handle a large number of requests at the same time.
  + Efficient: Node.js is very efficient, which means that it can use less resources than other programming languages.
  + Easy to learn: Node.js is very easy to learn, which makes it a good choice for beginners.
  + Flexible: Node.js is very flexible, which means that it can be used to build a variety of applications.

### Here are some of the drawbacks of using Node.js:

* + Not well-suited for complex applications: Node.js is not well- suited for complex applications that require a lot of state management.
  + Not well-suited for long-running applications: Node.js is not well- suited for long-running applications



## Lab Experiments:-

### Creating a first.js file

console.log("My First Node.js program");

### #Output

D:\NodeJS>node first.js My First Node.js program

### Executing fun.js file using functions

function greet(){ var a=1;

var message; if (a==1) {

message = "Hi!";

} else {

message = "Bye!";

}

console.log(message);

}

greet();

### #output

D:\NodeJS>node fun.js Hi!

### Creating a web server

var server = http.createServer((req, res) => { res.write("Hello World! I have created my first server!"); res.end();

});

server.listen(2000);

console.log("Server started... Running on localhost:3000");

### #Output

D:\NodeJS>node httpserver.js

Server started... Running on localhost:2000

### #Output on webpage

**After typing localhost:2000 we get the message…**

Hello World! I have created my first server!

## Express Js

Express.js is a web framework for Node.js. It is a fast, robust and asynchronous in nature.

Our Express.js tutorial includes all topics of Express.js such as Express.js installation on windows

And Linux, request object, response object, get method, post method, cookie management, scaffolding, file upload, template etc.

Express is a fast, assertive, essential and moderate web framework of Node.js. You can assume express as a layer built on the top of the Node.js that helps manage a server and routes. It provides a robust set of features to develop web and mobile applications.

Let's see some of the core features of Express framework:

* + It can be used to design single-page, multi-page and hybrid web applications.
  + It allows to setup middlewares to respond to HTTP Requests.
  + It defines a routing table which is used to perform different actions based on HTTP method and URL.
  + It allows to dynamically render HTML Pages based on passing arguments to templates.
  + Asynchronous and single threaded
  + MVC like structure
  + Robust API makes routing easy



## Lab Experiments:-

### Connecting to port

var express = require('express'); var app = express();

app.get('/', function (req, res) { res.send("Welocme to GeeksforGeeks!");

});

app.listen(5000);

### Credential checking

const express = require('express');

const bodyParser = require('body-parser'); const app = express();

const port = 3000; // Change to your desired port

// Middleware for parsing form data app.use(bodyParser.urlencoded({ extended: true }));

// Serve static HTML files app.use(express.static( dirname));

// Admin login route app.post('/adminlogin', (req, res) => {

const username = req.body.name; const password = req.body.password;

// Check admin credentials (replace with more secure authentication) if (username === 'admin' && password === 'admin123') {

// Admin login successful, redirect to the admin registration page res.redirect('/adonate.html');

} else {

// Admin login failed, you can customize this response

res.send('Admin login failed. Please check your username and password.');

}

});

app.listen(port, () => {

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

});

## MongoDB

MongoDB is a No SQL database. It is an open-source, cross-platform, document- oriented database written in C++.

Our MongoDB tutorial includes all topics of MongoDB database such as insert documents, update documents, delete documents, query documents, projection, sort() and limit() methods, create a collection, drop collection, etc. There are also given MongoDB interview questions to help you better understand the MongoDB database.

What is MongoDBMongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

The manufacturing company 10gen has defined MongoDB as:

"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen

MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

Window azure is a cloud computing platform and infrastructure, created by Microsoft, to build, deploy and manage applications and service through a global network.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market

as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

MongoDB2.4.9 was the latest and stable version which was released on January 10, 2014. Purpose of Building MongoDB

The primary purpose of building MongoDB is:

* + Scalability
  + Performance
  + High Availability
  + Scaling from single server deployments to large, complex multi-site architectures.



## Lab Experiments:-

### Connecting to DB

const express = require('express'); const mongoose = require('mongoose');

const bodyParser = require('body-parser'); const app = express();

const port = 3000;

// Connect to MongoDB mongoose.connect('mongodb://127.0.0.1:27017/student', { useNewUrlParser: true,

useUnifiedTopology: true

}).then(() => {

console.log('Connected to MongoDB');

}).catch(err => {

console.error('MongoDB connection error:', err);

});

### Create

router.get('/save', function (req, res) {

const newStudent = new StudentModel({ StudentId: 101,

Name: "Sam", Roll: 1, Birthday: 2001 - 09 - 08

});

newStudent.save(function (err, data) { if (err) {

console.log(error);

}

else {

res.send("Data inserted");

}

});

});

### Find

app.get('/findall', function (req, res) { StudentModel.find(function (err, data) {

if (err) { console.log(err);

}

else {

res.send(data);

}

});

});

### Find One

app.get('/findfirst', function (req, res) { StudentModel.findOne({ StudentId: { $gt: 185 } },

function (err, data) { if (err) {

console.log(err);

}

else {

res.send(data);

});

}

});

### Delete

app.get('/delete', function (req, res) { StudentModel.remove({ StudentId: 188 },

function (err, data) { if (err) {

console.log(err);

}

else {

res.send(data);

});

}

});

### Update

app.post('/update', function (req, res) { StudentModel.findByIdAndUpdate(req.body.id,

{ Name: req.body.Name }, function (err, data) { if (err) {

console.log(err);

}

else {

res.send(data); console.log("Data updated!");

});

}

});

## Type Script

TypeScript is an open-source programming language developed by Microsoft that extends and enhances JavaScript. It is designed to address some of the shortcomings of JavaScript by adding static typing to the language. TypeScript is often referred to as a "superset" of JavaScript because any valid JavaScript code is also valid TypeScript code, but TypeScript introduces optional static type checking.

Here are some key features and characteristics of TypeScript:

**Static Typing**: TypeScript allows developers to specify types for variables, function parameters, and return values. This enables early detection of type-related errors during the compilation phase, leading to more robust and reliable code.

**Type Inference**: TypeScript's type system can often infer types based on the context, reducing the need for explicit type annotations. This strikes a balance between static typing and the dynamic nature of JavaScript.

**Interfaces and Custom Types**: TypeScript allows you to define custom data types using interfaces or type aliases, providing a way to describe the shape of objects and data structures more explicitly.

**Classes and Modules**: TypeScript supports modern JavaScript features, including classes and modules, making it easier to organize and structure code in a more maintainable way.

**ESNext Support**: TypeScript is designed to work with the latest ECMAScript (ES) standards. It supports features from ES6, ES7, and beyond, enabling developers to use modern JavaScript syntax.

**Tooling and IDE Support**: TypeScript has excellent tooling support, including code editors like Visual Studio Code. It offers features like code completion, refactoring, and type checking, enhancing developer productivity.

**Compatibility**: TypeScript is backward-compatible with JavaScript, so you can gradually adopt it in your existing projects without rewriting everything.

**Strong Ecosystem**: TypeScript has a strong and growing ecosystem of libraries and frameworks, making it suitable for a wide range of web and server-side development projects.

**Community and Documentation**: TypeScript has an active and supportive community, with extensive documentation, online resources, and a central repository on GitHub.

**Transpilation**: TypeScript code is transpiled into standard JavaScript before running in a web browser or server environment, ensuring compatibility with a wide range of platforms.

Developers often choose TypeScript for projects where code maintainability, collaboration, and catching errors early in the development process are essential. It's particularly popular for web development, including frontend and backend development with frameworks like Angular, React, and Node.js, but can be used in various other application domains as well.

## Lab Experiments:

### Arrays

const numbers = [1, 2, 3]; // inferred to type number[] numbers.push(4); // no error

numbers.push("2");

let head: number = numbers[0];

### Functions

// the `: number` here specifies that this function returns a number function getTime(): number {

return new Date().getTime();

}

### Print

function printHello(): void { console.log('Hello!');

}

### Multiply

function multiply(a: number, b: number) { return a \* b;

}

### Classes

class Person { name: string;

}

const person = new Person(); person.name = "Jane";

### class2

class Person {

private name: string;

public constructor(name: string) { this.name = name;

}

public getName(): string { return this.name;

}

}

const person = new Person("Jane"); console.log(person.getName());

# Implementation

## HTML CODES

## ADMIN PAGE

## <!DOCTYPE html>

## <html>

## <head>

## <style>

## body {

## font-family: Arial, sans-serif;

## background-color: #f2f2f2;

## margin: 0;

## }

## header {

## background-color: #333;

## color: #fff;

## text-align: center;

## padding: 10px;

## }

## .logo {

## font-size: 24px;

## }

## .options {

## text-align: center;

## margin-top: 20px;

## }

## .options a {

## text-decoration: none;

## }

## .button-farmer {

## background-color: #1E90FF;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## margin-right: 10px;

## cursor: pointer;

## }

## .button-farm {

## background-color: #228B22;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## margin-right: 10px;

## cursor: pointer;

## }

## .button-display {

## background-color: #FF6347;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## cursor: pointer;

## }

## .options button:hover {

## opacity: 0.9;

## }

## </style>

## </head>

## <body>

## <header>

## <div align="center">

## <h1 class="logo"><b>MODIFICATION PAGE</b></h1>

## </div>

## </header>

## 

## <div class="options">

## <a href="farmer.html"><button class="button-farmer">Farmer</button></a>

## <a href="farm.html"><button class="button-farm">Farm</button></a>

## <a href="display.html"><button class="button-display">Display</button></a>

## </div>

## </body>

## </html>

## ADMIN\_LOGIN(admin.login)

## <!DOCTYPE html>

## <html>

## <head>

## <title>ADMIN LOGIN</title>

## <link rel="stylesheet" type="text/css" href="food.css">

## <style>

## body {

## background-color: #cccccc;

## display: flex;

## flex-direction: column;

## align-items: center;

## text-align: center;

## }

## .login-container {

## background-color: #fff;

## border: 1px solid #ccc;

## border-radius: 5px;

## box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

## padding: 20px;

## width: 300px;

## }

## .login-container h2 {

## text-align: center;

## }

## .form-group {

## margin: 15px 0;

## }

## .form-group label {

## display: block;

## font-weight: bold;

## }

## .form-group input[type="text"],

## .form-group input[type="password"] {

## width: 100%;

## padding: 10px;

## border: 1px solid #ccc;

## border-radius: 5px;

## }

## .form-group button {

## background-color: #007BFF;

## color: #fff;

## border: none;

## border-radius: 5px;

## padding: 10px 20px;

## cursor: pointer;

## }

## .form-group button:hover {

## background-color: #0056b3;

## }

## </style>

## </head>

## <body>

## <h1>ADMIN LOGIN</h1>

## <div class="login-container">

## <h2>Login</h2>

## <form action="/admin" method="post">

## <div class="form-group">

## <label for="uname">Username:</label>

## <input type="text" id="uname" name="uname" placeholder="Enter your username">

## </div>

## <div class="form-group">

## <label for="password">Password:</label>

## <input type="password" id="password" name="password" placeholder="Enter your password">

## </div>

## <div class="form-group">

## <button type="submit">Login</button>

## </div>

## </form>

## </div>

## 

## </body>

## </html>

## FARMER(farmer.html)

## <!DOCTYPE html>

## <html>

## <head>

## <style>

## body {

## font-family: Arial, sans-serif;

## background-color: #f2f2f2;

## margin: 0;

## }

## header {

## background-color: #333;

## color: #fff;

## text-align: center;

## padding: 10px;

## }

## .logo {

## font-size: 24px;

## }

## .options {

## text-align: center;

## margin-top: 20px;

## }

## .options a {

## text-decoration: none;

## }

## .button-add {

## background-color: #1E90FF;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## margin-right: 10px;

## cursor: pointer;

## }

## .button-delete {

## background-color: #FF6347;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## margin-right: 10px;

## cursor: pointer;

## }

## .button-update {

## background-color: #32CD32;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## cursor: pointer;

## }

## .button-Display {

## background-color: #ea537e;

## color: #fff;

## padding: 10px 20px;

## font-weight: bold;

## border: none;

## cursor: pointer;

## }

## .options button:hover {

## opacity: 0.9;

## }

## </style>

## </head>

## <body>

## <header>

## <div align="center">

## <h1 class="logo"><b>Farmer Page</b></h1>

## </div>

## </header>

## 

## <div class="options">

## <a href="add\_farmer.html"><button class="button-add">Add</button></a>

## <a href="deleteFarmer.html"><button class="button-delete">Delete</button></a>

## <a href="updatefarmer.html"><button class="button-update">Update</button></a>

## <a href="display\_farmers.html"><button class="button-Display">Display</button></a>

## </div>

## </body>

## </html>

## ADD\_FARMER(addfarmer.html)

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Add Farmer</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 5px;

}

h2 {

text-align: center;

}

label {

font-weight: bold;

}

input[type="text"],

input[type="number"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

width: 100%;

background-color: #007bff;

color: #fff;

padding: 10px;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="container">

<h2>Add Farmer</h2>

<form action="/addFarmer" method="post">

<div>

<label for="name">Name:</label>

<input type="text" id="name" name="name" placeholder="Farmer's Name" required>

</div>

<div>

<label for="unique\_id">Unique ID:</label>

<input type="text" id="unique\_id" name="unique\_id" placeholder="Unique ID" required>

</div>

<div>

<label for="place">Place:</label>

<input type="text" id="place" name="place" placeholder="Farmer's Place" required>

</div>

<div>

<label for="age">Age:</label>

<input type="number" id="age" name="age" placeholder="Farmer's Age" required>

</div>

<button type="submit">Add Farmer</button>

</form>

</div>

</body>

</html>

**DELETE FARMER(deletefarmer.html)**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Delete Farmer</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 5px;

}

h2 {

text-align: center;

}

label {

font-weight: bold;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

width: 100%;

background-color: #ff0000;

color: #fff;

padding: 10px;

border: none;

cursor: pointer;

}

button:hover {

background-color: #cc0000;

}

</style>

</head>

<body>

<div class="container">

<h2>Delete Farmer by Unique ID</h2>

<form action="/deleteFarmer" method="post">

<div>

<label for="unique\_id">Unique ID:</label>

<input type="text" id="unique\_id" name="unique\_id" placeholder="Enter Unique ID" required>

</div>

<button type="submit">Delete Farmer</button>

</form>

</div>

</body>

</html>

**DISPLAY FARMER LIST(displayfarmer.html)**

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

}

h1 {

background-color: #333;

color: #fff;

text-align: center;

padding: 10px;

}

table {

width: 80%;

margin: 20px auto;

border-collapse: collapse;

border: 1px solid #ccc;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

}

table th, table td {

padding: 10px;

text-align: left;

}

table th {

background-color: #333;

color: #fff;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

</style>

</head>

<body>

<h1>Farmer List</h1>

<table border="1">

<thead>

<tr>

<th>Name</th>

<th>Unique ID</th>

<th>Place</th>

<th>Age</th>

</tr>

</thead>

<tbody id="farmer-list">

<!-- Farmer data will be displayed here dynamically -->

</tbody>

</table>

<script>

// JavaScript code to fetch and display farmer data

// Make a fetch request to /displayFarmers

fetch('/displayFarmers')

.then(response => response.json())

.then(data => {

const farmerList = document.getElementById('farmer-list');

data.forEach(farmer => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${farmer.name}</td>

<td>${farmer.unique\_id}</td>

<td>${farmer.place}</td>

<td>${farmer.age}</td>

`;

farmerList.appendChild(row);

});

})

.catch(error => console.error(error));

</script>

</body>

</html>

**UPDATE\_FARMER(updatefarmer.html)**

<!DOCTYPE html>

<html>

<head>

<title>Update Farmer</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

}

h1 {

text-align: center;

margin: 20px;

}

form {

background-color: #fff;

padding: 20px;

max-width: 400px;

margin: 0 auto;

border: 1px solid #ccc;

border-radius: 5px;

}

label {

display: block;

font-weight: bold;

}

input {

width: 100%;

padding: 10px;

margin: 5px 0;

}

button {

background-color: #1E90FF;

color: #fff;

padding: 10px 20px;

font-weight: bold;

border: none;

cursor: pointer;

margin-top: 10px;

}

button:hover {

opacity: 0.9;

}

</style>

</head>

<body>

<h1>Update Farmer Information</h1>

<form id="update-form" action="/updateFarmer" method="POST">

<label for="unique\_id">Unique ID:</label>

<input type="text" id="unique\_id" name="unique\_id" required>

<label for="place">Place:</label>

<input type="text" id="place" name="place" required>

<label for="age">Age:</label>

<input type="number" id="age" name="age" required>

<button type="submit">Update Farmer</button>

</form>

</body>

</html>

**CROPS PAGE(crops.html)**

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

}

header {

background-color: #333;

color: #fff;

text-align: center;

padding: 10px;

}

.logo {

font-size: 24px;

}

.options {

text-align: center;

margin-top: 20px;

}

.options a {

text-decoration: none;

}

.button-add {

background-color: #1E90FF;

color: #fff;

padding: 10px 20px;

font-weight: bold;

border: none;

margin-right: 10px;

cursor: pointer;

}

.button-delete {

background-color: #FF6347;

color: #fff;

padding: 10px 20px;

font-weight: bold;

border: none;

margin-right: 10px;

cursor: pointer;

}

.button-update {

background-color: #32CD32;

color: #fff;

padding: 10px 20px;

font-weight: bold;

border: none;

margin-right: 10px;

cursor: pointer;

}

.button-display {

background-color: #ea537e;

color: #fff;

padding: 10px 20px;

font-weight: bold;

border: none;

cursor: pointer;

}

.options button:hover {

opacity: 0.9;

}

</style>

</head>

<body>

<header>

<div align="center">

<h1 class="logo"><b>Farm (Crops) Page</b></h1>

</div>

</header>

<div class="options">

<a href="addcrop.html"><button class="button-add">Add</button></a>

<a href="delete\_crop.html"><button class="button-delete">Delete</button></a>

<a href="update\_crop.html"><button class="button-update">Update</button></a>

<a href="display\_crops.html"><button class="button-display">Display</button></a>

</div>

</body>

</html>

**ADD\_CROP(addcrop.html)**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Add Crop</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 5px;

}

h2 {

text-align: center;

}

label {

font-weight: bold;

}

input[type="text"],

input[type="number"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

width: 100%;

background-color: #007bff;

color: #fff;

padding: 10px;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

</style>

</head>

<body>

<div class="container">

<h2>Add Crop</h2>

<form action="/addCrop" method="post">

<div>

<label for="crop">Crop:</label>

<input type="text" id="crop" name="crop" placeholder="Crop Name" required>

</div>

<div>

<label for="time">Time:</label>

<input type="text" id="time" name="time" placeholder="Crop Time" required>

</div>

<div>

<label for="crop\_id">Crop ID:</label>

<input type="text" id="crop\_id" name="crop\_id" placeholder="Crop ID" required>

</div>

<div>

<label for="season">Season:</label>

<input type="text" id="season" name="season" placeholder="Crop Season" required>

</div>

<button type="submit">Add Crop</button>

</form>

</div>

</body>

</html>

**DELETE\_CROP(deletecrop.html)**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Delete Crop</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

}

.container {

max-width: 500px;

margin: 0 auto;

padding: 20px;

background: #fff;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 5px;

}

h2 {

text-align: center;

}

label {

font-weight: bold;

}

input[type="text"] {

width: 100%;

padding: 8px;

margin-bottom: 10px;

}

button {

width: 100%;

background-color: #ff0000;

color: #fff;

padding: 10px;

border: none;

cursor: pointer;

}

button:hover {

background-color: #cc0000;

}

</style>

</head>

<body>

<div class="container">

<h2>Delete Crop by Crop ID</h2>

<form action="/deleteCrop" method="post">

<div>

<label for="crop\_id">Crop ID:</label>

<input type="text" id="crop\_id" name="crop\_id" placeholder="Enter Crop ID" required>

</div>

<button type="submit">Delete Crop</button>

</form>

</div>

</body>

</html>

**DISPLAY\_CROP(display.html)**

<!DOCTYPE html>

<html>

<head>

<style>

body {

font-family: Arial, sans-serif;

background-color: #f2f2f2;

margin: 0;

}

h1 {

background-color: #333;

color: #fff;

text-align: center;

padding: 10px;

}

table {

width: 80%;

margin: 20px auto;

border-collapse: collapse;

border: 1px solid #ccc;

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

}

table th, table td {

padding: 10px;

text-align: left;

}

table th {

background-color: #333;

color: #fff;

}

tr:nth-child(even) {

background-color: #f2f2f2;

}

</style>

</head>

<body>

<h1>Crop List</h1>

<table border="1">

<thead>

<tr>

<th>Crop</th>

<th>Crop ID</th>

<th>Time</th>

<th>Season</th>

</tr>

</thead>

<tbody id="crop-list">

<!-- Crop data will be displayed here dynamically -->

</tbody>

</table>

<script>

// JavaScript code to fetch and display crop data

// Make a fetch request to /displayCrops (the route to retrieve crop data)

fetch('/displayCrops')

.then(response => response.json())

.then(data => {

const cropList = document.getElementById('crop-list');

data.forEach(crop => {

const row = document.createElement('tr');

row.innerHTML = `

<td>${crop.crop}</td>

<td>${crop.crop\_id}</td>

<td>${crop.time}</td>

<td>${crop.season}</td>

`;

cropList.appendChild(row);

});

})

.catch(error => console.error(error));

</script>

</body>

</html>

**USER LOGIN(login.html)**

<!DOCTYPE html>

<html>

<head>

<title>Bon-Appetit</title>

<link rel="stylesheet" type="text/css" href="food.css">

<style>

body {

background-color: #cccccc;

display: flex;

flex-direction: column;

align-items: center;

text-align: center;

}

.login-container {

background-color: #fff;

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

padding: 20px;

width: 300px;

}

.login-container h2 {

text-align: center;

}

.form-group {

margin: 15px 0;

}

.form-group label {

display: block;

font-weight: bold;

}

.form-group input[type="text"],

.form-group input[type="password"] {

width: 100%;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

.form-group button {

background-color: #007BFF;

color: #fff;

border: none;

border-radius: 5px;

padding: 10px 20px;

cursor: pointer;

}

.form-group button:hover {

background-color: #0056b3;

}

</style>

</head>

<body>

<h1>FREEMER</h1>

<div class="login-container">

<h2>Login</h2>

<form action="/login" method="post">

<div class="form-group">

<label for="uname">Username:</label>

<input type="text" id="uname" name="uname" placeholder="Enter your username">

</div>

<div class="form-group">

<label for="password">Password:</label>

<input type="password" id="password" name="password" placeholder="Enter your password">

</div>

<div class="form-group">

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

</div>

</form>

</div>

<div class="register-button">

<a href="register.html" class="btn">Register Now</a>

</div>

</body>

</html>

**REGISTRATION(registration.html)**

<!DOCTYPE html>

<html>

<head>

<style>

/\* Add custom CSS styles for the registration form container \*/

.register-container {

background-color: #fff;

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

padding: 20px;

width: 300px;

margin: 0 auto; /\* Center the form horizontally \*/

margin-top: 20px; /\* Add some space between login and registration forms \*/

}

/\* Add custom CSS styles for form elements \*/

.form-group {

margin: 10px 0;

}

.form-group label {

display: block;

}

.form-group input {

width: 100%;

padding: 5px;

border: 1px solid #ccc;

border-radius: 3px;

}

.form-group button {

background-color: #337ab7;

color: #fff;

border: none;

border-radius: 3px;

padding: 10px 15px;

cursor: pointer;

}

.login-link {

text-align: center;

margin-top: 10px;

}

.login-link a {

text-decoration: none;

color: #337ab7;

}

</style>

</head>

<body>

<!-- Registration Container -->

<div class="register-container">

<h2>Register</h2>

<form action="/register" method="post">

<div class="form-group">

<label for="fname">First name:</label>

<input type="text" id="fname" name="fname" placeholder="Choose a username">

</div>

<div class="form-group">

<label for="lname">Last name:</label>

<input type="text" id="lname" name="lname" placeholder="Choose a username">

</div>

<div class="form-group">

<label for="uname">Username:</label>

<input type="text" id="uname" name="uname" placeholder="Choose a username">

</div>

<div class="form-group">

<label for="password">Password:</label>

<input type="password" id="reg-password" name="password" placeholder="Choose a password">

</div>

<!-- Add more registration fields as needed -->

<div class="form-group">

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

</div>

</form>

<div class="login-link">

<p>If already have an account, <a href="login.html">go to the login page</a></p>

</div>

</div>

</body>

</html>

**HOMEPAGE**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>FEERMER</title>

<style>

/\* CSS styles go here \*/

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background: linear-gradient(180deg, #ffffff, #e0e0e0); /\* Background gradient \*/

}

table {

width: 100%;

background: linear-gradient(90deg, #e65c00, #ff833a); /\* Table background gradient \*/

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);

border: none; /\* Remove the border \*/

}

tr.header {

background: #ffffff;

}

tr.header td {

colspan: 6;

text-align: center;

}

tr.navigation {

background: linear-gradient(90deg, #ff833a, #e65c00); /\* Navigation background gradient \*/

}

tr.navigation td {

padding: 10px;

text-align: center;

}

tr.navigation a {

display: inline-block;

padding: 10px 20px;

margin: 5px;

background: #337ab7;

color: #fff;

text-decoration: none;

border: none;

cursor: pointer;

border-radius: 5px;

}

tr.image {

background: #e65c00;

}

tr.image img {

width: 100%;

}

</style>

</head>

<body>

<table>

<tr class="header">

<td>

<img src="images/logo.jpg" style="height:100px;">

</td>

<td colspan="5" align="center">

<h1>FEERMER</h1>

</td>

</tr>

<tr class="navigation">

<td>

<a href="login.html">LOGIN</a>

</td>

<td>

<a href="register.html">REGISTER</a>

</td>

<td>

<a href="admin.html">ADMIN LOGIN</a>

</td>

</tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr></tr>

<tr class="image">

<td colspan="6">

<img src="https://thumbs.dreamstime.com/b/indian-farmer-green-corn-field-indian-farmer-green-corn-field-139750404.jpg">

</td>

</tr>

</table>

</body>

</html>

**HRPAGE**

<!DOCTYPE html>

<html>

<head>

<title>Hire a Farmer</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background: #f4f4f4;

}

.container {

max-width: 1400px; /\* Increased the max-width \*/

margin: 0 auto;

padding: 20px;

background: #fff;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);

}

.header {

text-align: center;

padding: 20px 0;

}

.header img {

max-width: 200px;

height: auto;

}

.navigation {

background: linear-gradient(90deg, #ff833a, #e65c00);

text-align: center;

padding: 10px 0;

}

.navigation a {

display: inline-block;

padding: 10px 20px;

margin: 0 10px;

background: #337ab7;

color: #fff;

text-decoration: none;

border: none;

cursor: pointer;

border-radius: 5px;

}

.welcome {

text-align: center;

padding: 20px;

background: pink;

}

.welcome h1 {

color: #ff6600;

font-size: 24px;

}

.farmer-list {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

gap: 20px;

padding: 20px;

}

.farmer-card {

flex: 1;

box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);

border-radius: 5px;

overflow: hidden;

}

.farmer-card img {

max-width: 100%;

height: auto;

}

.farmer-info {

padding: 20px;

}

.farmer-info p {

margin: 10px 0;

}

.farmer-info p b {

color: green;

}

.footer {

background: #ff80d5;

text-align: center;

padding: 10px;

color: #fff;

}

.buttons {

text-align: center;

padding: 20px;

}

.buttons a {

display: inline-block;

padding: 10px 20px;

margin: 10px;

background: #337ab7;

color: #fff;

text-decoration: none;

border: none;

cursor: pointer;

border-radius: 5px;

}

</style>

</head>

<body>

<div class="container">

<div class="header">

<img src="images/logo.jpg" alt="Logo">

</div>

<div class="navigation">

<a href="HR.html">Hire Human Resources</a>

<a href="adopt farm.html">Adopt a farm</a>

<a href="login.html">Log Out</a>

</div>

<div class="welcome">

<h1>Welcome to Hire a farmer site... 🤗. If you have land that is just kept ideally,

and if you want to make some money with it, you came to the right place...

Hire the best farmer there to offer. Let them make money for you. Invest your money in nature.

And you are employing many people and doing a service to our nation.

</h1>

</div>

<div class="farmer-list">

<div class="farmer-card">

<img src="images/farmer5.jpg" alt="Farmer 1">

<div class="farmer-info">

<p><b>Name: M.Veera babu</b></p>

<p>Age: 42</p>

<p>Rating: 7.5/10</p>

<p>Locality: Vijayawada</p>

<p>Phone: 6283979341</p>

</div>

</div>

<div class="farmer-card">

<img src="images/farmer6.jpg" alt="Farmer 2">

<div class="farmer-info">

<p><b>Name: N.Ramayya</b></p>

<p>Age: 39</p>

<p>Rating: 8/10</p>

<p>Locality: Kondapalli</p>

<p>Phone: 9374938409</p>

</div>

</div>

<div class="farmer-card">

<img src="images/farmer7.webp" alt="Farmer 3">

<div class="farmer-info">

<p><b>Name: Prashanth</b></p>

<p>Age: 31</p>

<p>Rating: 9/10</p>

<p>Locality: Vijayawada</p>

<p>Phone: 8394822391</p>

</div>

</div>

<div class="farmer-card">

<img src="images/farmers200.jpg" alt="Farmer 4">

<div class="farmer-info">

<p><b>Name: S. Sathyam</b></p>

<p>Age: 45</p>

<p>Rating: 7/10</p>

<p>Locality: Mylavaram</p>

<p>Phone: 8984793139</p>

</div>

</div>

<div class="farmer-card">

<img src="images/farmers100.jpg" alt="Farmer 5">

<div class="farmer-info">

<p><b>Name: R. Gopi</b></p>

<p>Age: 46</p>

<p>Rating: 8/10</p>

<p>Locality: Ongole</p>

<p>Phone: 7394898392</p>

</div>

</div>

<div class="farmer-card">

<img src="images/farmers300.jpg" alt="Farmer 6">

<div class="farmer-info">

<p><b>Name: N. Srinu</b></p>

<p>Age: 44</p>

<p>Rating: 9.5/10</p>

<p>Locality: Vijayawada</p>

<p>Phone: 8294893190</p>

</div>

</div>

</div>

<div class="buttons">

<a href="adopt farm.html">Adopt a farm</a>

</div>

<div class="footer">

By JP 😉

</div>

</div>

</body>

</html>

**ADOPT FARM**

<!DOCTYPE html>

<html>

<head>

<title>Adopt a Farm</title>

<style>

body {

background-color: #f7f7f7;

margin: 0;

font-family: Arial, sans-serif;

}

table {

width: 100%;

border-collapse: collapse;

}

tr.heading {

background: #33cc33;

}

tr.sub-heading {

background: #337ab7;

}

h2 {

text-align: center;

padding: 20px 0;

color: #337ab7;

}

tr.footer {

background: #33cc33;

color: #fff;

}

tr.footer td {

text-align: center;

padding: 10px;

}

.image-grid {

display: flex;

justify-content: center;

flex-wrap: nowrap; /\* Prevent wrapping to keep images in a single row \*/

}

.image-card {

width: auto; /\* Allow images to determine their width \*/

padding: 20px;

margin: 0 10px; /\* Add horizontal margin to separate images \*/

border: 1px solid #ccc;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

text-align: center;

background: #fff;

}

.image-card img {

width: 100%; /\* Make the images fill the image cards \*/

max-height: 100%; /\* Limit the image card height \*/

}

.image-button {

display: block;

width: 100%;

padding: 10px;

background: #337ab7;

color: #fff;

text-decoration: none;

border: none;

cursor: pointer;

border-radius: 5px;

}

.image-button:hover {

background: #ff1a1a;

}

.image-name {

font-weight: bold;

color: #337ab7;

}

</style>

</head>

<body>

<table>

<tr class="heading">

<td colspan="6" align="center">

<img src="images/logo.jpg" style="height:100px;">

</td>

</tr>

<tr class="sub-heading">

<td align="center" style="width: 33.33%;">

<a href="HR.html" class="image-button">Hire Human Resources</a>

</td>

<td align="center" style="width: 33.33%;">

<a href="adopt farm.html" class="image-button">Adopt a farm</a>

</td>

<td align="center" style="width: 33.33%;">

<a href="vehicles.html" class="image-button">Hire vehicles</a>

</td>

</tr>

<tr>

<td colspan="6">

<h2>Farms near you</h2>

</td>

</tr>

<tr>

<td class="info">

<div class="image-grid">

<div class="image-card">

<img src="images/brinjal.jpg" alt="Brinjal">

<p class="image-name">Brinjal</p>

<a href="#" class="image-button">Adopt</a>

</div>

<div class="image-card">

<img src="images/Wheat.jpg" alt="Wheat">

<p class="image-name">Wheat</p>

<a href="#" class="image-button">Adopt</a>

</div>

<div class="image-card">

<img src="images/tomato.jpg" alt="Tomato">

<p class="image-name">Tomato</p>

<a href="#" class="image-button">Adopt</a>

</div>

<div class="image-card">

<img src="images/Chilli.jpg" alt="Chilli">

<p class="image-name">Chilli</p>

<a href="#" class="image-button">Adopt</a>

</div>

<div class="image-card">

<img src="images/sugar.jpg" alt="Sugarcane">

<p class="image-name">Sugarcane</p>

<a href="#" class="image-button">Adopt</a>

</div>

<div class="image-card">

<img src="images/mango.jpg" alt="Mango">

<p class="image-name">Mango</p>

<a href="#" class="image-button">Adopt</a>

</div>

</div>

</td>

</tr>

<tr class="footer">

<td colspan="6" align="center">

<p>By JP ☺</p>

</td>

</tr>

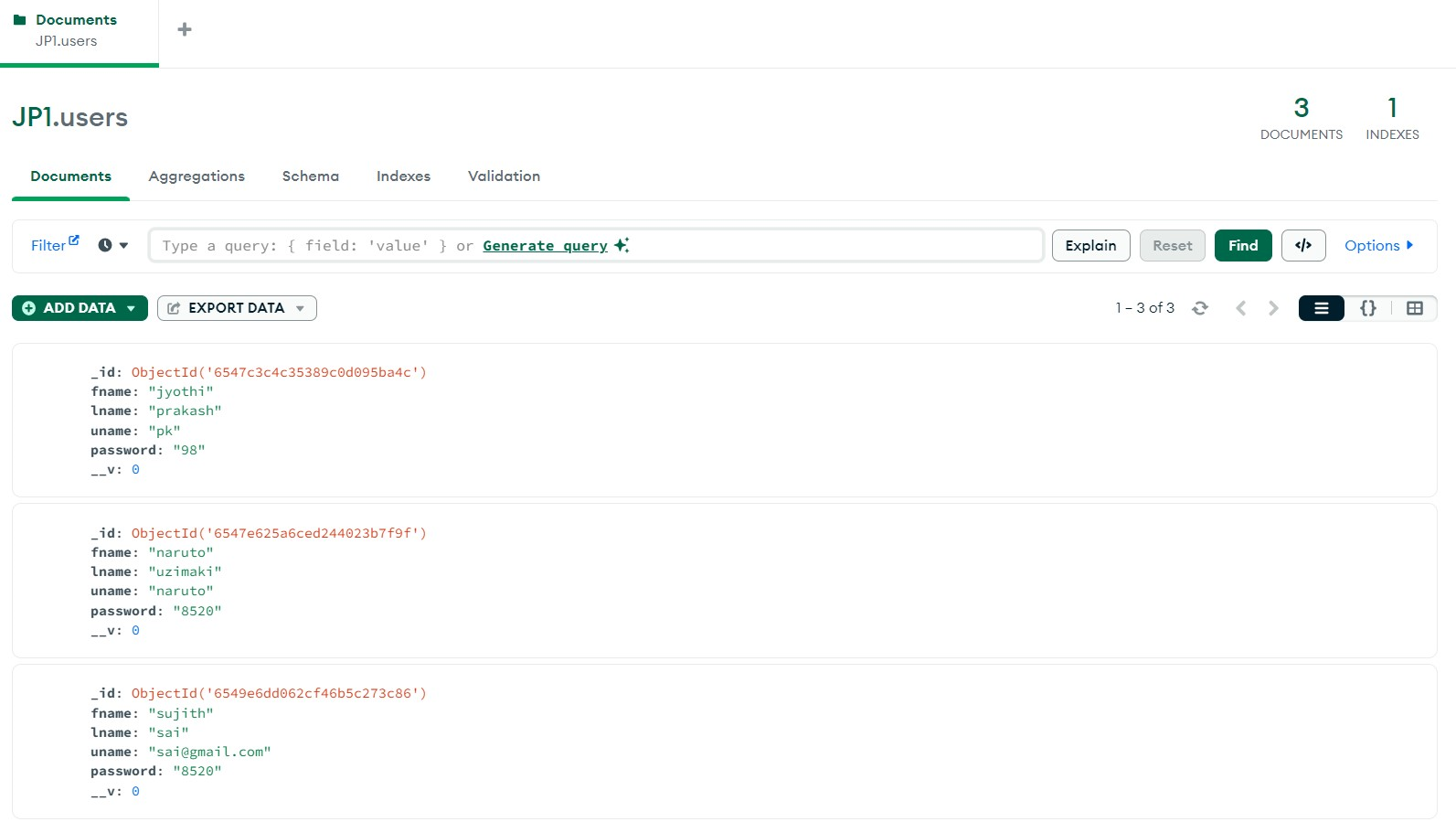
</table>

</body>

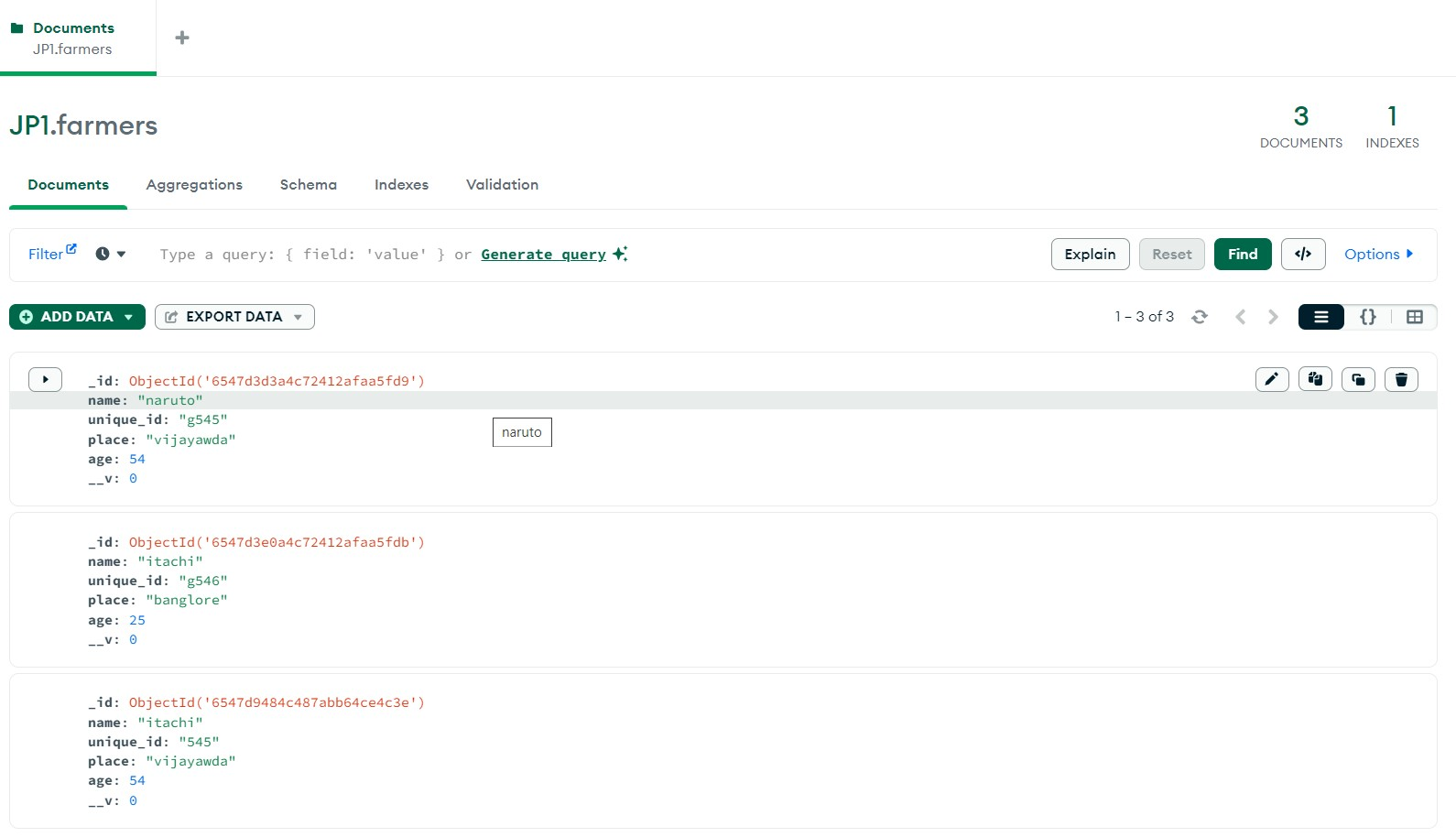
</html>

**LIST OF DATABASE TABLES**

**SIGNUP**



**FARMER**



**CROPS**



**EXPRESS.js and NODE.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const path = require('path');

const app = express();

const port = 3002;

mongoose.connect('mongodb://127.0.0.1:27017/JP1');

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'MongoDB connection error:'));

db.once('open', () => {

console.log('Connected to MongoDB');

});

**FARMER DATABASE**

//farmer

const farmerSchema = new mongoose.Schema({

name: {

type: String,

required: true,

},

unique\_id: {

type: String,

required: true,

},

place: {

type: String,

required: true,

},

age: {

type: Number,

required: true,

},

});

const Farmer = mongoose.model('Farmer', farmerSchema);

module.exports = Farmer;

**CROP DATABASE**

const cropSchema = new mongoose.Schema({

crop: {

type: String,

required: true,

},

time: {

type: String,

required: true,

},

crop\_id: {

type: String,

required: true,

},

season: {

type: String,

required: true,

},

});

const Crop = mongoose.model('Crop', cropSchema);

module.exports = Crop;

**USER DATABASE**

const userSchema = new mongoose.Schema({

fname: String,

lname: String,

uname: String,

password: String,

});

const User = mongoose.model('User', userSchema);

app.use(bodyParser.urlencoded({ extended: true }));

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

app.use(express.static(\_\_dirname));

**REGISTRATION**

app.get('/', (req, res) => {

res.sendFile(\_\_dirname + '/page.html');

});

app.post('/register', (req, res) => {

const userData = {

fname: req.body.fname,

lname: req.body.lname,

password: req.body.password,

uname: req.body.uname

};

User.create(userData)

.then(user => {

// Registration successful, handle user object

res.redirect('/login'); // Redirect to login page

})

.catch(err => {

console.error(err);

res.send('An error occurred during registration');

});

});

**LOGIN**

app.post('/login', async (req, res) => {

try {

const { uname, password } = req.body;

// Log the values received from the form for debugging

console.log('Login attempt with uname:', uname, 'password:', password);

const user = await User.findOne({ uname, password }).exec();

if (!user) {

console.log('User not found in the database');

res.send('Login failed. Invalid credentials.');

} else {

console.log('Login successful');

res.redirect('/hr.html');

}

} catch (err) {

console.error(err);

res.send('An error occurred during login.');

}

});

app.post('/update', (req, res) => { const { email, phone } = req.body;

const newUser = new User({ email, phone }); newUser.save(err => {

if (err) {

res.send('Update failed');

} else {

res.send('Update successful');

}

});

});

app.post('/delete', async (req, res) => {

const confirmation = req.body['confirm-delete'];

if (confirmation === 'DELETE') {

const usernameToDelete = req.body.uname; // Get the username from the form input

try {

const user = await User.findOneAndDelete({ uname: usernameToDelete });

if (!user) {

res.status(404).send('User not found.');

} else {

res.send('User deleted successfully.');

}

} catch (err) {

res.status(500).send('Error deleting user.');

}

} else {

res.status(400).send('Confirmation text does not match. Account not deleted.');

}

});

**ADMIN LOGIN**

app.post('/admin', (req, res) => {

const adminUsername = req.body.uname;

const adminPassword = req.body.password;

// Log the values received from the form for debugging

// Check if the provided username and password match the admin credentials

if (adminUsername === 'jp' && adminPassword === 'jp123') {

res.redirect('/modified.html');

} else {

res.send('Invalid credentials. Please try again.');

}

});

// Serve login.html

app.get('/login', (req, res) => {

res.sendFile(\_\_dirname + '/login.html');

});

// Serve home.html

app.get('/hr.html', (req, res) => {

res.sendFile(\_\_dirname + '/hr.html');

});

**ADD FARMER**

app.post('/addFarmer', async (req, res) => {

const { name, unique\_id, place, age } = req.body;

try {

const farmer = new Farmer({

name,

unique\_id,

place,

age,

});

await farmer.save(); // Save the farmer to the database

res.send('Farmer added successfully.');

} catch (err) {

console.error(err);

res.status(500).send('An error occurred while adding the farmer.');

}

});

**DELETE FARMER**

app.post('/deleteFarmer', async (req, res) => {

const uniqueIdToDelete = req.body.unique\_id;

// Log the unique ID received for debugging

console.log('Unique ID to delete:', uniqueIdToDelete);

try {

const farmer = await Farmer.findOneAndDelete({ unique\_id: uniqueIdToDelete });

if (!farmer) {

console.log('Farmer not found in the database');

res.status(404).send('Farmer not found.');

} else {

console.log('Farmer deleted successfully');

res.send('Farmer deleted successfully.');

}

} catch (err) {

console.error(err);

res.status(500).send('An error occurred while deleting the farmer.');

}

});

**DISPLAY FARMER**

app.get('/displayFarmers', async (req, res) => {

try {

const farmers = await Farmer.find({});

// Convert the data to JSON and send it as a response

res.json(farmers);

} catch (err) {

console.error(err);

// Handle errors here

}

});

**UPDATE FARMER**

app.get('/updateFarmer', (req, res) => {

res.sendFile(\_\_dirname + '/updateFarmer.html');

});

// Assuming you have an instance of Express called 'app'

app.post('/updateFarmer', async (req, res) => {

const { unique\_id, age, place } = req.body;

try {

const farmer = await Farmer.findOne({ unique\_id }).exec();

if (!farmer) {

res.status(404).send('Farmer not found.');

} else {

farmer.age = age;

farmer.place = place;

await farmer.save(); // Save the updated farmer to the database

res.send('Farmer information updated successfully.');

}

} catch (err) {

console.error(err);

res.status(500).send('An error occurred while updating the farmer.');

}

});

**ADD CROP**

app.post('/addCrop', (req, res) => {

const { crop, time, crop\_id, season } = req.body;

// Create a new Crop instance with the form data

const newCrop = new Crop({

crop,

time,

crop\_id,

season,

});

// Save the crop data to the database

newCrop.save()

.then(() => {

res.send('Crop added successfully.');

})

.catch(err => {

console.error(err);

res.status(500).send('An error occurred while adding the crop.');

});

});

// Handle the crop deletion form submission

**DELETE CROP**

// Handle the crop deletion form submission

app.post('/deleteCrop', async (req, res) => {

const cropIdToDelete = req.body.crop\_id;

try {

// Use async/await to find and delete the crop based on cropIdToDelete

const result = await Crop.findOneAndDelete({ crop\_id: cropIdToDelete });

if (!result) {

res.status(404).send('Crop not found.');

} else {

res.send('Crop deleted successfully.');

}

} catch (err) {

console.error(err);

res.status(500).send('Error deleting crop.');

}

});

**DISPLAY CROP**

app.get('/displayCrops', async (req, res) => {

try {

// Fetch the crop data from your database

// Replace 'Crop' with the actual model name you've defined for crops

const crops = await Crop.find({}).exec();

// Send the crop data as a JSON response

res.json(crops);

} catch (error) {

console.error(error);

res.status(500).send('An error occurred while fetching crop data.');

}

});

**PORT**

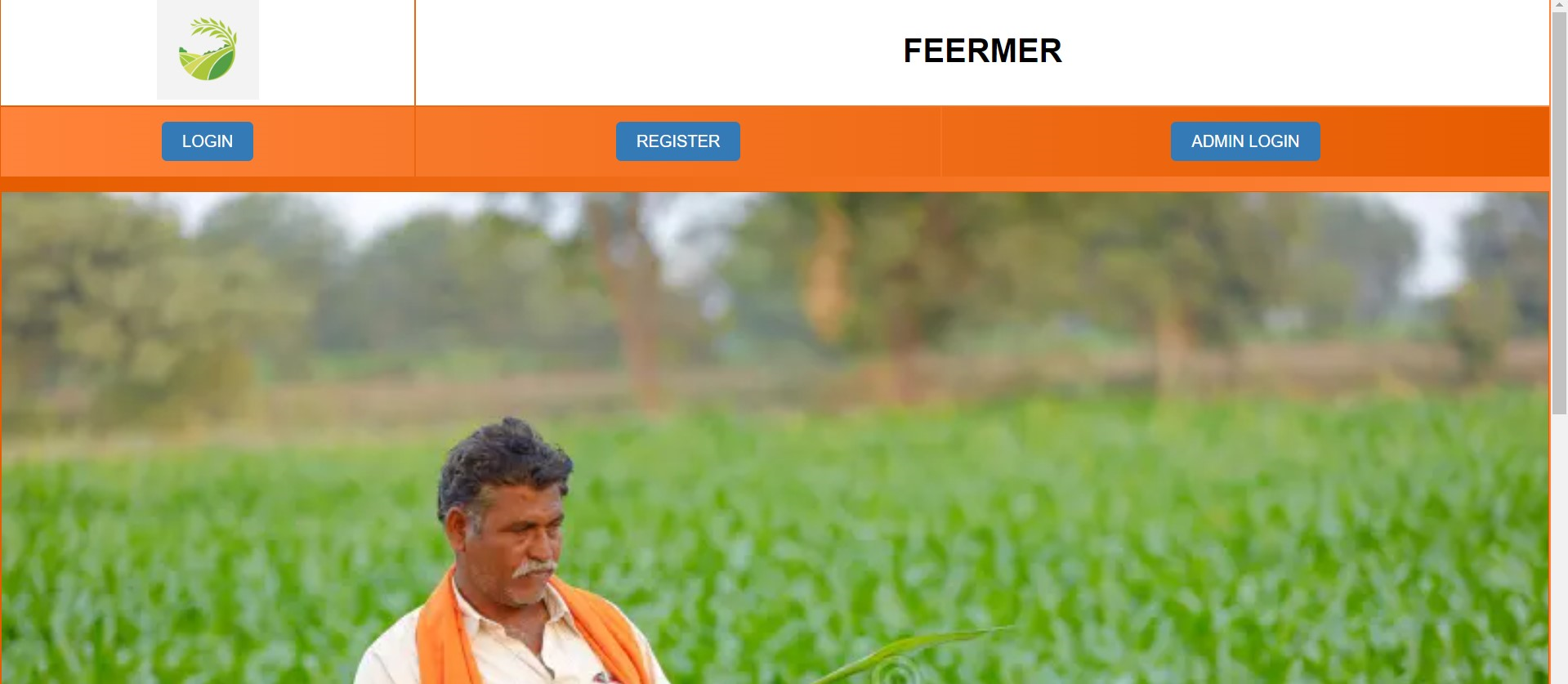
app.listen(port, () => {

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

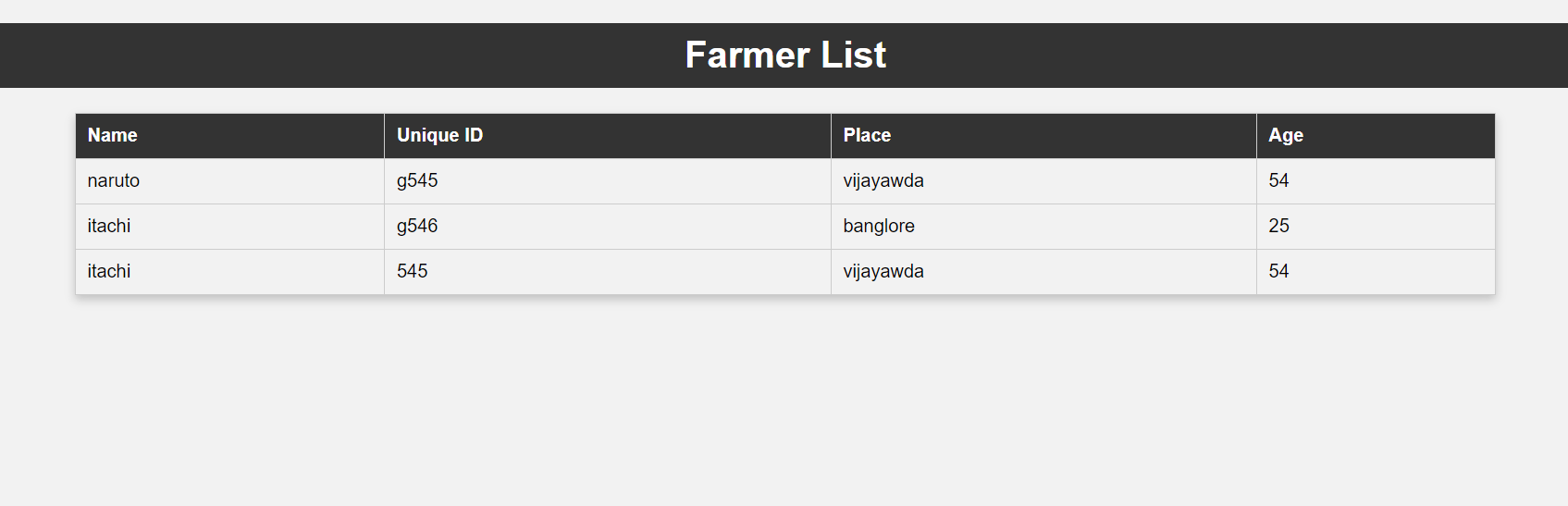
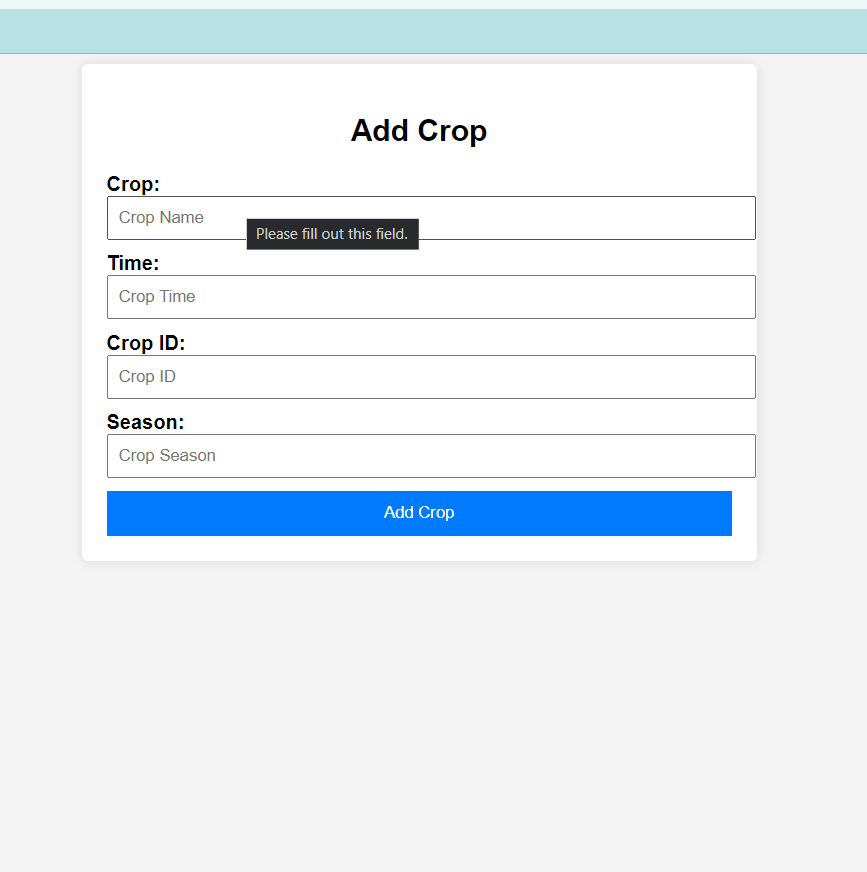
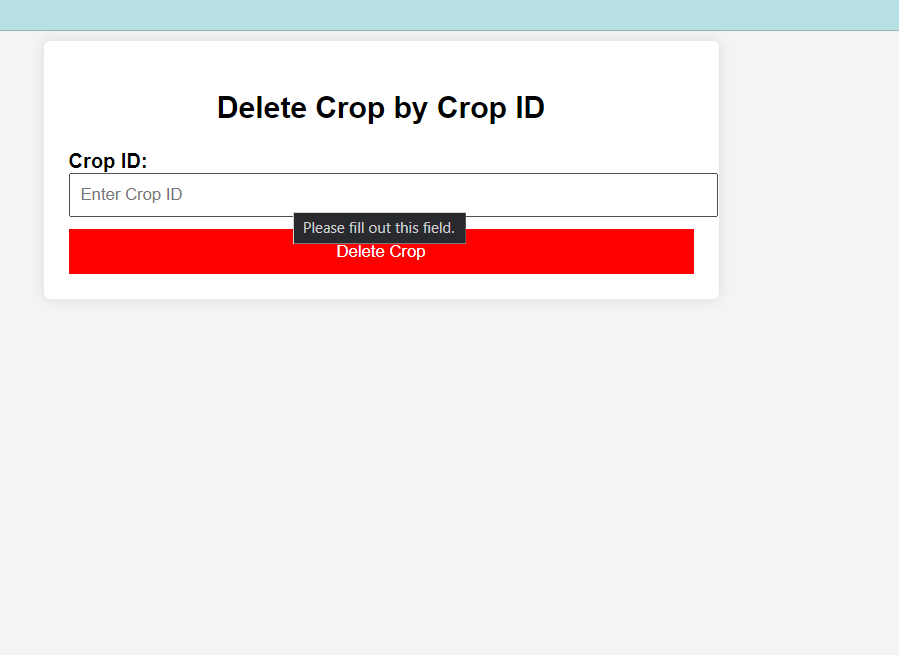
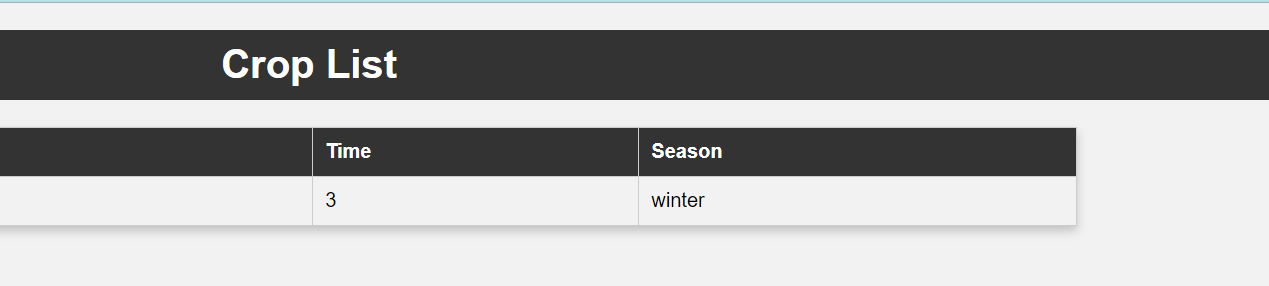
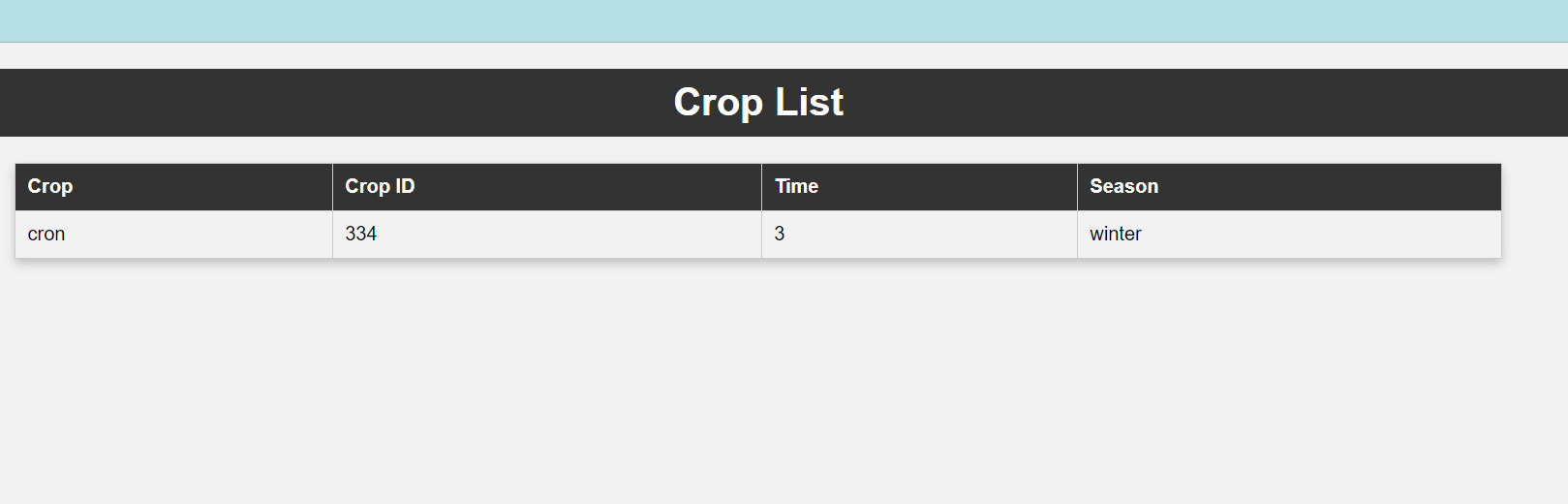
});

**SCREENSHOTS**

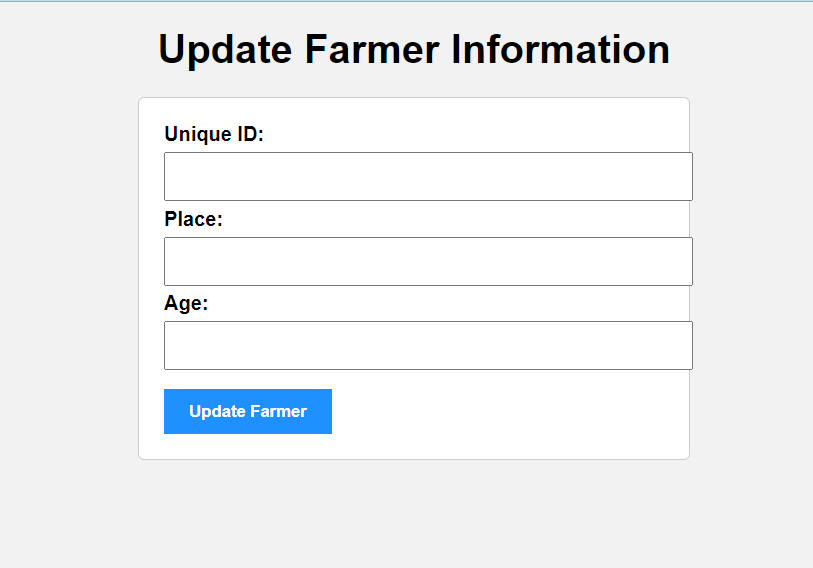
**HOME PAGE:**



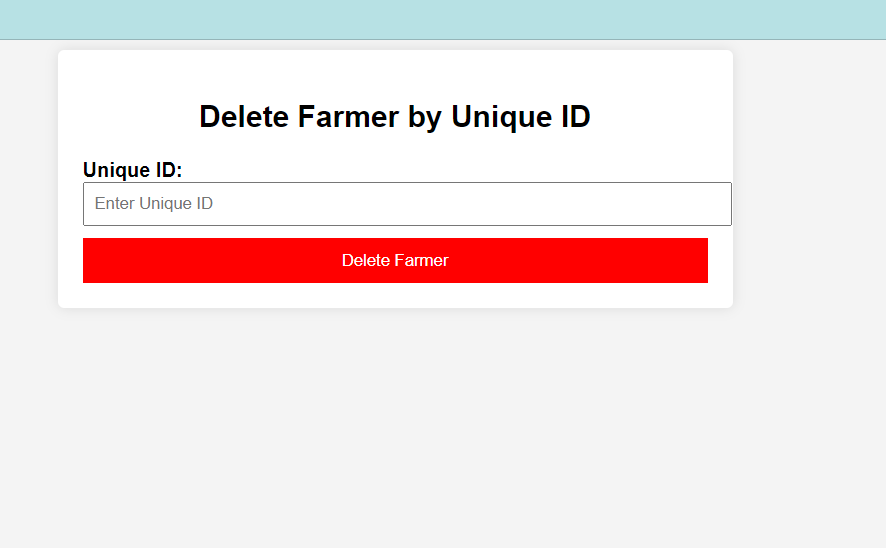
**LOGIN PAGE**



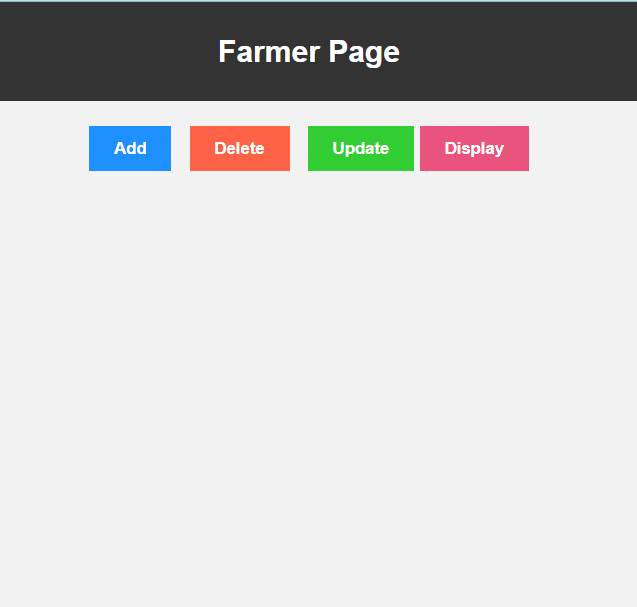
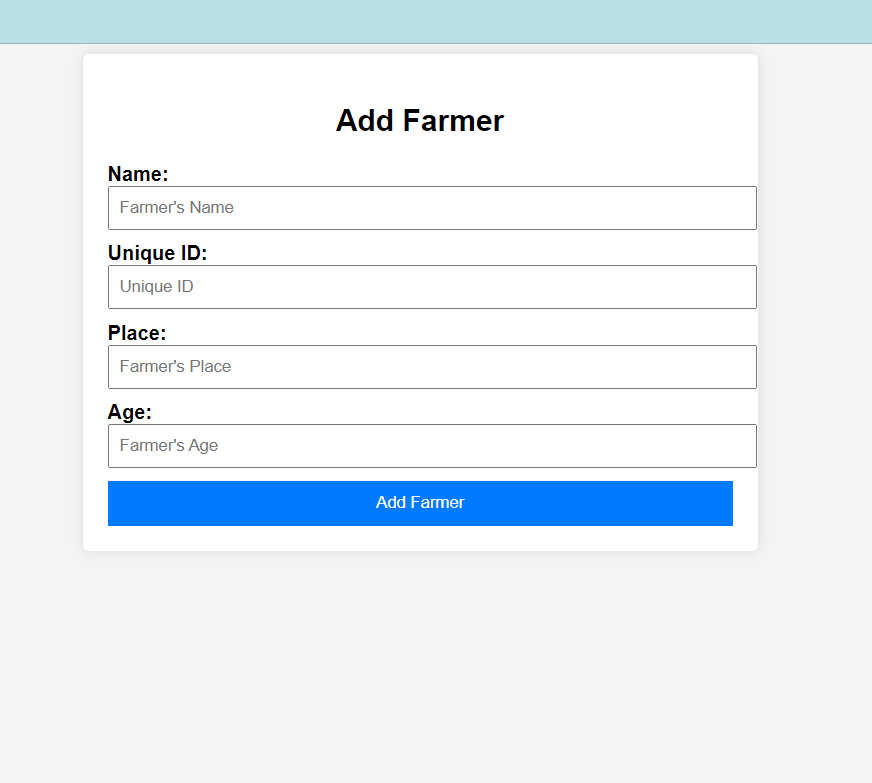
**UPDATE FARMER**



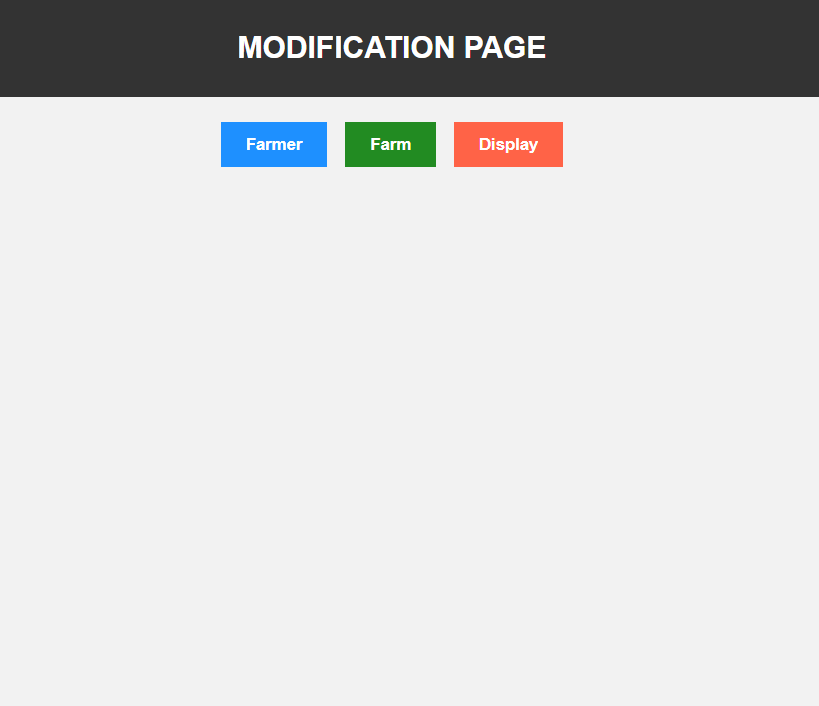
**DELETE FARMER**



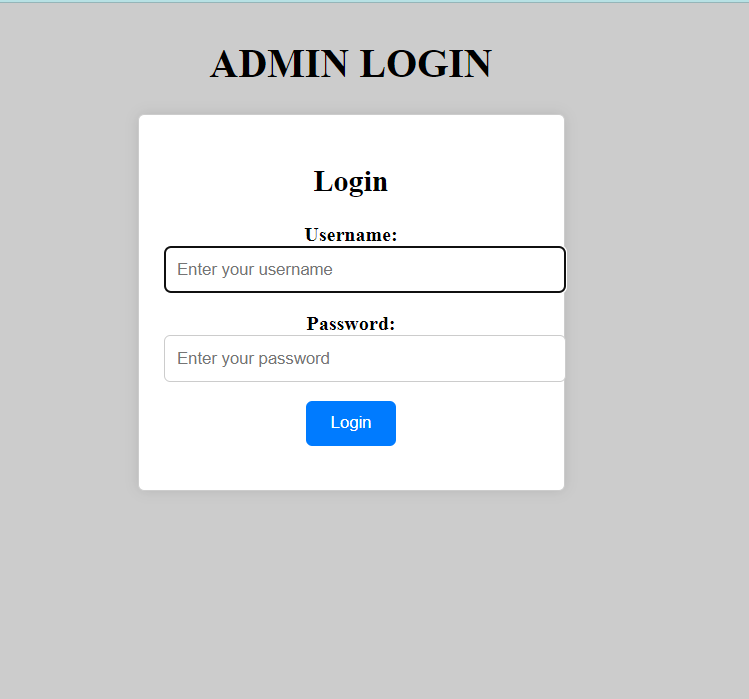
**ADD FARMER**



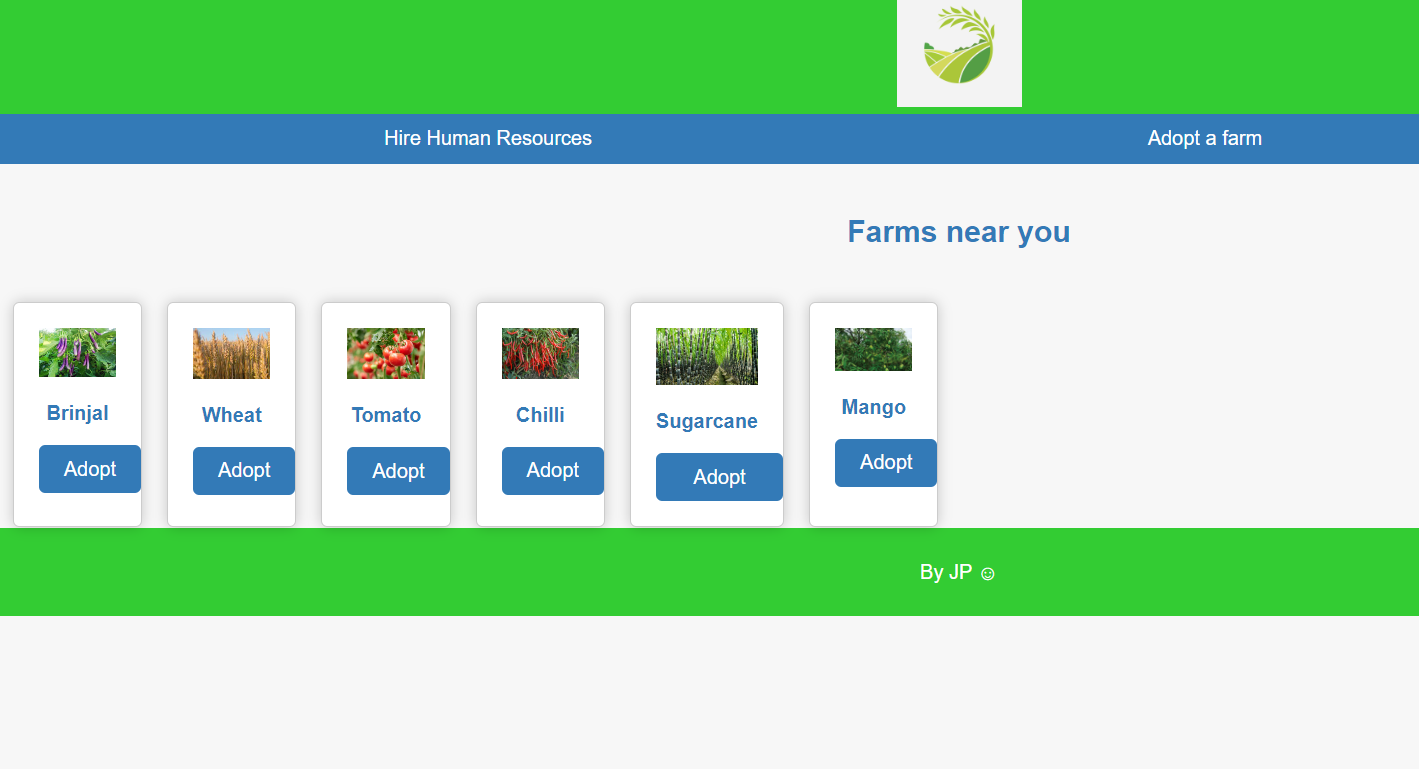
**ADMIN PAGE**



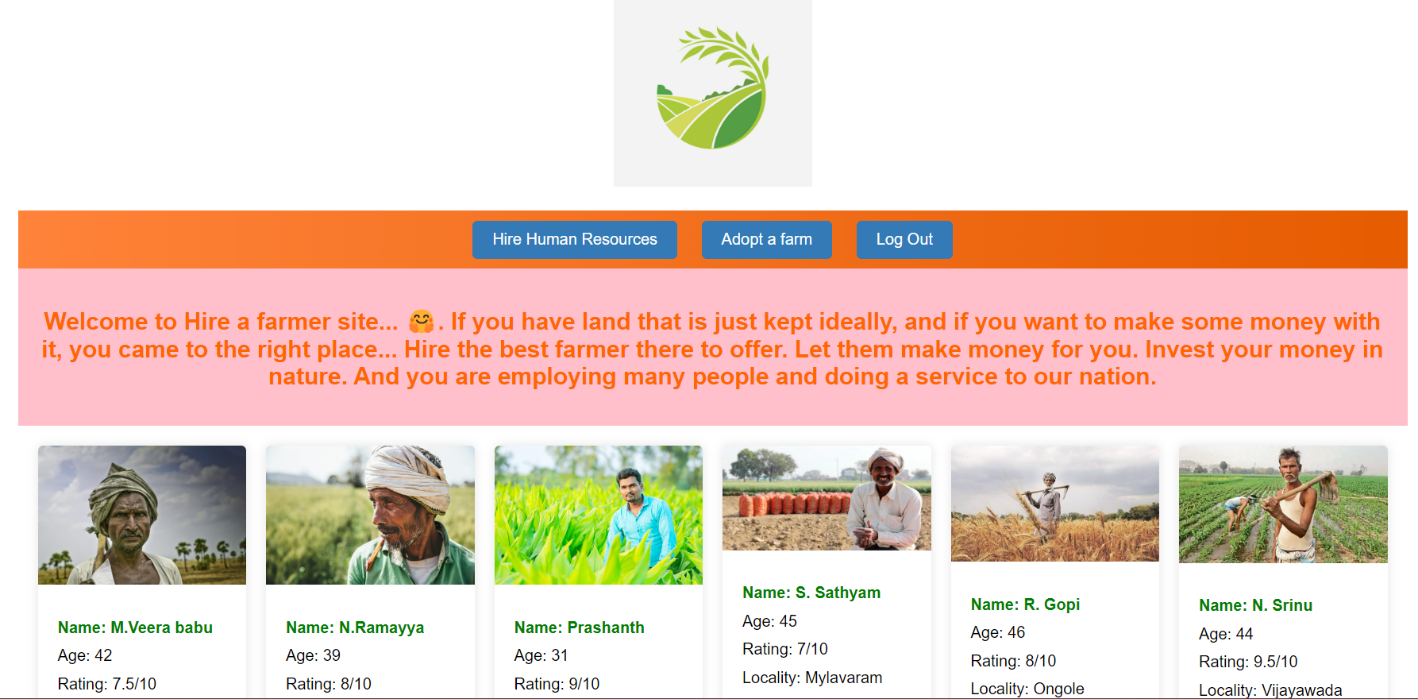
**ADMIN LOGIN**



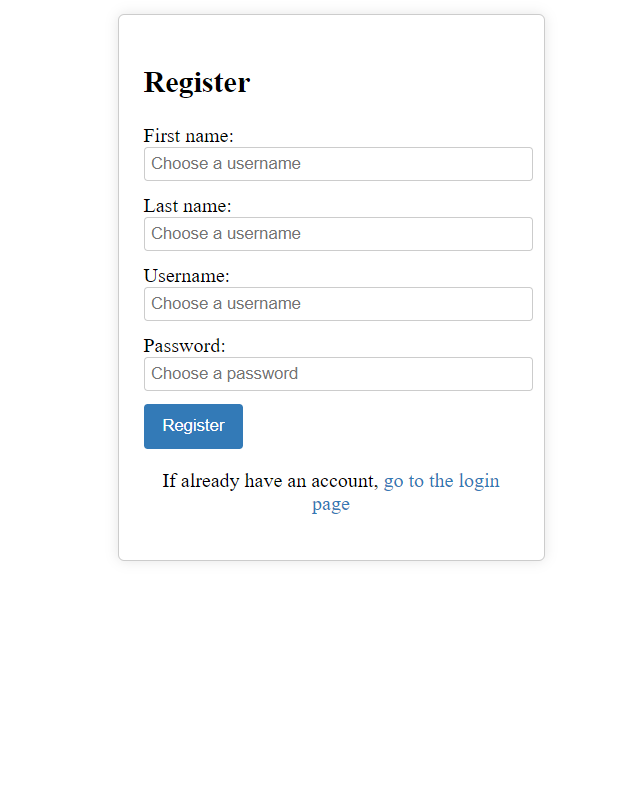
**ADOPT FARM**



**HIRE FARMER**



**REGISTRATION**

**LOGIN**

**CONCLUSION**

The "Adopt-a-Farmer" project represents a pioneering initiative that merges technology with community-driven sustainable agriculture. With a robust backend infrastructure, secure data handling, and seamless user experiences, this platform acts as a bridge connecting urban clients and rural farmers.

By actively participating in the agricultural process, urban clients contribute to sustainable farming practices, enhance food security, and support local communities. The project's backend components, including user registration, geo-location services, a payment gateway, and a messaging system, make this collaboration both effective and user-friendly.

This initiative, supported by scalable and secure cloud-based infrastructure, ensures that it can grow and adapt to an ever-increasing user base. The knowledge-sharing component allows urban users to learn from experienced farmers, strengthening community bonds and agricultural skills.

In conclusion, the "Adopt-a-Farmer" project is poised to reshape the way we engage with and learn from farmers. It leverages technology to bridge the urban-rural divide and promotes sustainability, making it a catalyst for positive change in the world of agriculture. As urban clients virtually adopt farmers and nurture their crops, this project embodies the spirit of collaboration, knowledge sharing, and a sustainable future

**REFERENCE**

References I used in this project are.

### EXTERNAL RESOURCES:

I used HTML tags CSS, JAVASCRIPT, NODE JS & EXPRESS JS &

MONGODB attributes from external sources and used them to develop my website. Google information about HTML, CSS JAVASCRIPT, NODE JS & EXPRESS JS & MONGODB JAVASCRIPT, NODE JS & EXPRESS JS & MONGODB to know

various information about them and how to use them is my website

**WEB RESOURCES:**

**Infosys Springboard**

* HTML[-TOC - HTML5 - The Language | Infosys](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_17739732834840810000_shared/overview) [Springboard(onwingspan.com)](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_17739732834840810000_shared/overview)CSS - [TOC - CSS3 | Infosys Springboard](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_15281000932633230000_shared/overview) [(onwingspan.com)](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_15281000932633230000_shared/overview)
* JavaScript[-TOC - JavaScript | Infosys Springboard](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_18109698366332810000_shared/overview) [(onwingspan.com)](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_18109698366332810000_shared/overview)Node Js[-Infosys Springboard](https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex_19002830632103186000_shared?collectionId=lex_32407835671946760000_shared) [(onwingspan.com)](https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex_19002830632103186000_shared?collectionId=lex_32407835671946760000_shared)
* Express Js[-Infosys Springboard (onwingspan.com)](https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex_19002830632103186000_shared?collectionId=lex_32407835671946760000_shared)

## W3 Schools HTML

* HTML- [HTML Tutorial](https://www.w3schools.com/html/) [(w3schools.com)](https://www.w3schools.com/html/)CSS [-CSS](https://www.w3schools.com/css/) [Tutorial (w3schools.com)](https://www.w3schools.com/css/)
* JavaScript [-JavaScript Tutorial](https://www.w3schools.com/js/) [(w3schools.com)](https://www.w3schools.com/js/)MongoDB[-MongoDB](https://www.w3schools.com/mongodb/) [Tutorial (w3schools.com)](https://www.w3schools.com/mongodb/)

# Self Assessments Certificates

# HTML-5



**FOUNDATION CERTIFICATE**



**JAVA SCRIPT**

