SRINIVAS UNIVERSITY

INSTITUTE OF ENGINEERING & TECHNOLOGY

MANGALURU MUKKA,

HOSPITAL REGISTRATION FORM (USING HTML,CSS&JAVASCRIPT)

For the Academic year 2023-24

Submitted by

1.Anvith - 01SU23EC006

2.Bhoomika N.G– 01SU23EC007

3.Dhyan– 01SU23EC008

4.Gokul– 01SU23EC009

5.Gowtham – 01SU23EC010

Submitted to

MS. Priyanka , Software Technical Trainer

CONTENTS:

Preface:

This project has been composed with the aim of covering a part of B. Tech
(II Sem) under IBM course Cloud Fundamentals as prescribed by Srinivas University Of Engineering And Technology, Mukka, Mangalore. It is with great enthusiasm that we present this report, which represents collective efforts by Anvith, Bhoomika, Dhyan, Gokul and Gowtham . The running project has been presented through Google Chrome .We extend our sincere gratitude to IBM for providing us with the opportunity to present this project.

Welcome to the world of web development, where creativity meets functionality, and innovation knows no bounds. In this preface, we embark on a journey into the realm of building a Hospital Registration Form using the dynamic trio of HTML, CSS, and JavaScript.

As the digital landscape evolves, the need for efficient and user-friendly web interfaces becomes increasingly vital, especially in sectors like healthcare, where seamless communication and data management are paramount.

This project delves into the creation of a registration form tailored specifically for hospitals, catering to both the administrative needs of healthcare providers and the convenience of patients. Through the fusion of HTML, CSS, and JavaScript, we aim to construct a robust and intuitive platform that streamlines the registration process while ensuring data accuracy and security.

HTML, the backbone of web development, provides the structure and semantic markup necessary for organizing the form elements, from basic input fields to complex data validation mechanisms. CSS steps onto the stage to add style and visual appeal, transforming raw HTML elements into polished, aesthetically pleasing components that enhance user experience.

But our journey doesn't end there. Enter JavaScript, the dynamic scripting language that breathes life into our form. With JavaScript, we can introduce interactivity and responsiveness, enabling real-time validation, error handling, and dynamic updates without the need for page reloads.

Throughout this project, we'll explore the intricacies of each language, leveraging their unique capabilities to create a seamless registration experience. From designing a clean and intuitive user interface to implementing backend functionality for data processing, we'll navigate the complexities of web development with diligence and creativity.

By the end of our journey, you'll not only have gained valuable insights into the fundamentals of web development but also acquired the skills to craft sophisticated web applications that bridge the gap between technology and user needs.

So, without further ado, let's embark on this exciting adventure and unlock the limitless possibilities of HTML, CSS, and JavaScript in shaping the future of healthcare registration systems.

AIM OF THE PROJECT:

In the dynamic landscape of healthcare, where efficiency, accuracy, and user experience are paramount, our aim is to develop a comprehensive Hospital Registration Form using HTML, CSS, and JavaScript. Through this project, we strive to achieve the following objectives:

Streamlined Registration Process: Design and implement a user-friendly interface that guides patients through the registration process seamlessly, reducing friction and improving efficiency.
Data Accuracy and Security: Implement robust validation mechanisms to ensure that the data entered by patients is accurate and conforms to predefined standards. Additionally, prioritize data security by employing encryption techniques and best practices to safeguard sensitive information.
Responsive Design: Create a responsive layout that adapts to various screen sizes and devices, ensuring an optimal user experience across desktops, tablets, and smartphones.
Real-time Validation: Leverage JavaScript to provide real-time validation feedback as users input their data, minimizing errors and enhancing usability.
Error Handling: Implement comprehensive error handling mechanisms to gracefully manage invalid inputs, guiding users toward correct data entry and preventing submission of incomplete or erroneous forms.
Accessibility: Ensure that the registration form is accessible to users with disabilities by adhering to web accessibility standards and incorporating features such as keyboard navigation and screen reader compatibility.

Dynamic Updates: Utilize JavaScript to enable dynamic updates and interactions within the form, such as auto-populating fields, displaying conditional sections based on user inputs, and providing live feedback on available appointment slots.

Integration with Backend Systems: Lay the groundwork for seamless integration with backend systems, allowing for secure storage and retrieval of patient data, as well as facilitating communication between the registration form and other healthcare management applications.

Scalability and Maintainability: Design the registration form with scalability and maintainability in mind, employing modular code structures, reusable components, and best practices to facilitate future enhancements and updates.

User Feedback and Iteration: Solicit feedback from users and stakeholders throughout the development process, iterating on the design and functionality of the registration form to ensure alignment with user needs and organizational objectives.

By accomplishing these objectives, we aim to deliver a Hospital Registration Form that not only meets the immediate requirements of healthcare institutions but also lays the foundation for continued innovation and improvement in patient registration processes. Through our commitment to excellence in web development, we seek to empower healthcare providers and enhance the overall quality of care delivery for patients.

Objectives:

1. User-Centric Design: Prioritize user experience by designing an intuitive and visually appealing registration form interface that caters to the diverse needs and preferences of patients.
2. Efficient Data Collection: Develop a structured form layout using HTML to systematically collect essential patient information, including personal details, contact information, medical history, and insurance details.
3. Validation and Error Prevention: Implement client-side validation using JavaScript to ensure the accuracy and completeness of data entered by patients, preventing submission of invalid or incomplete forms and providing immediate feedback on errors.
4. Security Measures: Employ security best practices, such as input sanitization, to mitigate the risk of data breaches and unauthorized access to sensitive patient information. Additionally, integrate encryption techniques to safeguard data transmission over the network.
5. Responsive Layout: Utilize CSS media queries to create a responsive design that adapts seamlessly to different screen sizes and devices, including desktops, laptops, tablets, and smartphones, enhancing accessibility and usability across various platforms.
6. Real-Time Feedback: Leverage JavaScript to provide real-time feedback to users as they interact with the registration form, including dynamic validation messages, autocomplete suggestions, and live updates based on user inputs.
7. Accessibility Compliance: Ensure compliance with web accessibility standards, such as WCAG (Web Content Accessibility Guidelines), by incorporating features like semantic HTML markup,

keyboard navigation, and screen reader compatibility, enabling access for users with disabilities.

8. Conditional Logic: Implement conditional logic using JavaScript to dynamically display or hide form fields and sections based on user selections or responses, optimizing the user experience and streamlining the registration process.
9. Cross-Browser Compatibility: Test the registration form across multiple web browsers, including Chrome, Firefox, Safari, and Edge, to ensure consistent functionality and appearance across different browser environments.
10. Integration with Backend Systems: Develop backend APIs (Application Programming Interfaces) using server-side technologies like Node.js or PHP to handle form submissions, validate data server-side, and store patient information in a secure database.
11. Progressive Enhancement: Employ progressive enhancement techniques to ensure that the registration form remains functional and accessible even in environments with limited JavaScript support, providing a robust user experience for all users.
12. Internationalization and Localization: Support multiple languages and locales by implementing internationalization (i18n) and localization (l10n) features, allowing users to interact with the registration form in their preferred language and format.
13. Performance Optimization: Optimize the performance of the registration form by minimizing file sizes, reducing HTTP requests, and employing techniques like lazy loading for images and asynchronous loading for JavaScript resources, ensuring fast load times and smooth user interactions.
14. Feedback Mechanisms: Incorporate feedback mechanisms, such as surveys or contact forms, to gather input from users regarding their experience with the registration process, enabling continuous improvement and refinement of the form.

Project Code:(Registration.html)

```
<!DOCTYPE html>
<html>
<head><title>Hospital Registration Form Validation</title></head>
<body bgcolor="#E4F0F8">
<script type='text/javascript'>
function formValidator()
{
// Make quick references to our fields
var firstname = document.getElementById('firstname');
var lastname = document.getElementById('lastname');
var email = document.getElementById('email');
var pass = document.getElementById('pass');
var addr = document.getElementById('addr');
var mobileno = document.getElementById('mobileno');
// Check each input in the order that it appears in the form!
if(notEmpty(firstname, "can not be null"))
{
if(isAlphabet(firstname, "Please enter only letters for your Firstname"))
{
if(lengthRestriction(firstname, 6))
```

```
{
if(isAlphabet(lastname, "Please enter only letters for your Lastname"))
{
if(emailValidator(email, "Please enter a valid email address"))
{
if(lengthRestriction(pass, 6))
{
if(isAlphanumeric(pass, "please enter Numbers and Letters Only
{
if(notEmpty(addr, "please enter the address"))
{
if(isNumeric(mobileno, "Please enter a valid mobileno"))
{
if(lengthRestriction1(mobileno, 10 , 10))
{
 return true;
}
}
}
}
}
}
}
}
}
}
return false;
}
function notEmpty(elem, helperMsg)
{
if(elem.value.length == 0)
{
alert(helperMsg);
elem.focus(); // set the focus to this inputreturn false;
}
return true;
}
function isNumeric(elem, helperMsg)
{
var numericExpression = /^[0-9]+$/;
if(elem.value.match(numericExpression))
{
return true;
}
else {
alert(helperMsg);elem.focus();
```

```
return false; }}
function isAlphabet(elem, helperMsg) {
var alphaExp = /^[a
-zA
-Z]+$/;
if(elem.value.match(alphaExp)) {
return true; }
else {
alert(helperMsg);
elem.focus();
return false; }}
function isAlphanumeric(elem, helperMsg) {
var alphaExp = /^[0
-9a
-zA
-Z]+$/;
if(elem.value.match(alphaExp)) {
return true; }
else {
alert(helperMsg);
elem.focus(
)
;
```

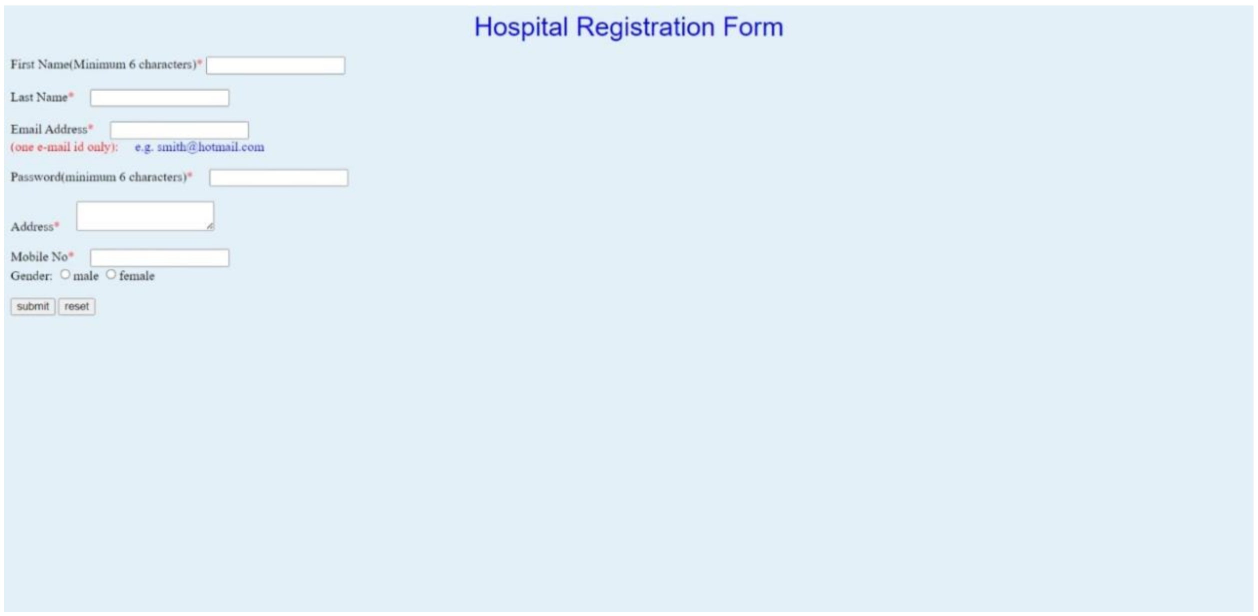
```
return false; }}
function lengthRestriction(elem, min)
{
var uInput = elem.value;
if(uInput.length >= min)
{
return true;
}
else
{
alert("Please enter minimum " +min+ " characters");
elem.focus();
return false;
}
}
function emailValidator(elem, helperMsg)
{
var emailExp = /^[\w\-\.\+]+\@[a-zA-Z0-9\.\-]+\.[a-zA-z0-9]{2,4}$/;
if(elem.value.match(emailExp))
{
return true;
}
else
```

```
{
alert(helperMsg);elem.focus();
return false;
}
}
function lengthRestriction1(elem, min, max)
{
var uInput = elem.value;
if(uInput.length >= min && uInput.length <= max)
{
return true;
}
else
{
alert("Please enter 10 numbers only");
elem.focus();
return false;
}
}
</script>
<center><font color="blue" size="6" face="arial">Hospital Registration
Form</font></center><br />
<form onsubmit='return formValidator()'
action="submitpage.html">First Name(Minimum 6
characters)<font color="red">* </font>
<input type='text' id='firstname' /><br /><br />
Last Name<font color="red"><font color="red">* </font> </font>
&nbsp;&nbsp;&nbsp;
<input type='text' id='lastname' /><br /><br />
Email Address<font color="red">* </font> &nbsp;&nbsp;&nbsp;
<input type='text' id='email' /><br />
<label style="color:red;">(one e-mail id only):</label>
&nbsp;&nbsp;&nbsp;
<label style="color:blue;">e.g. smith@hotmail.com</label><br /><br/>
Password(minimum 6 characters)<label style="color:red;">* </label>
&nbsp;&nbsp;&nbsp;
<input type='password' id='pass'><br /><br/>
Address<label style="color:red;">* </label> &nbsp;&nbsp;&nbsp;
<textarea rows="2" cols="20" id='addr' /></textarea> <br /> <br/>
Mobile No<label style="color:red;">* </label> &nbsp;&nbsp;&nbsp;
<input type='text' id='mobileno' /><br />
Gender: <input type='radio' name="gender">male
<input type='radio' name="gender">female<br/><br />
<input type='Submit' value='submit' />
<input type='Reset' value='reset' />
</form>
</body>
</html>
```

Submit page code(File Name is submitpage.html)

```
<html>
<head>
<style>p
{
font-size:50px;color:green;
text-align:center;padding :50px;
border : 2px double red;margin : 250px;
border-radius:100px;
}
</style>
</head>
<body>
<p> Successfuly Registered !!!!!!!<span style='font-
size:50px;'>&#128077;</span></p>
</body>
</html>
```

Output:


Hospital Registration Form

First Name(Minimum 6 characters)*

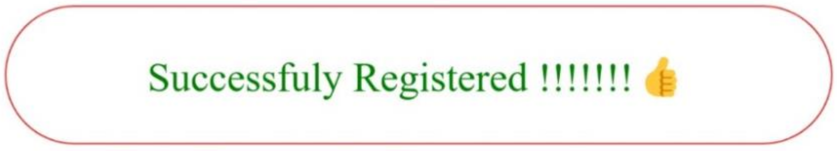
Last Name*

Email Address*
(one e-mail id only): e.g. smith@hotmail.com

Password(minimum 6 characters)*

Address*

Mobile No*
Gender: ○ male ○ female


Successfuly Registered !!!!!!! 👍

Contributions:

1. Anvith(01SU23EC006)– Html code

   Document Structure: The HTML document begins with the <!DOCTYPE html> declaration, followed by the <html> element enclosing the entire document. The <head> section contains metadata, including character encoding, viewport settings, and the page title. External CSS is linked for styling.
   Form Structure: The registration form is wrapped in a <form> element with the action attribute specifying the URL where form data will be submitted, and the method attribute set to "post" for data submission via HTTP POST method.
   Form Fields: The form is divided into fieldsets, each representing a distinct section of information. Within each fieldset, various form elements are included to capture patient details such as name, date of birth, gender, email address, phone number, medical history, insurance provider, and emergency contact information.
   Input Fields: <input> elements of different types (text, date, email, tel) are used to collect textual and numeric data, with the required attribute ensuring that mandatory fields are filled.
   Select Menu: The gender field is implemented using a <select> element with <option> tags for different gender options. It is also marked as required.
   Textarea: A <textarea> element allows for multi-line text input, suitable for capturing medical history.
   Submit Button: An <input type="submit"> element serves as the submission button, allowing users to submit the form data.
2. Bhoomika N.G **(01SU23EC007)** – Html code

Labels and Legends: Each form field is typically accompanied by a corresponding <label> element, providing descriptive text that aids users in understanding the purpose of the field. For grouped inputs, <fieldset> and <legend> elements are used to provide a title or description.

Required Fields: To denote mandatory fields, the required attribute is added to input elements, ensuring that users cannot submit the form without providing essential information.

Validation: HTML5 offers built-in validation features, such as the pattern attribute for defining input formats (e.g., regular expressions for validating email addresses), min and max attributes for setting numerical ranges, and maxlength attribute for limiting text input length.

Submit Button: Including an <input type="submit"> or <button> element allows users to submit the form once all required information has been entered.

Accessibility: Ensuring accessibility involves using semantic HTML elements, providing meaningful labels for form fields, and ensuring proper tab order for keyboard navigation. This ensures that users with disabilities can interact with the form effectively.

Form Structure: Organizing the form logically into sections or fieldsets aids in grouping related information together, enhancing user comprehension and navigation.Inside the form, there are input fields for name, email, phone number, and a dropdown for selecting the course. The "Register" button submits the form data.

3. Dhyan **(01SU23EC008)** – Html code

   Error Messages: Providing clear and informative error messages is crucial for guiding users when they submit invalid or incomplete data. These messages can be displayed using <span> or <div> elements placed near the corresponding form fields. They should convey specific instructions on how to correct the errors and meet the form's requirements.

   Validation Feedback: Beyond just displaying error messages, HTML5's built-in form validation features can dynamically provide feedback to users as they interact with the form. For example, input fields can change color or display icons to indicate valid or invalid entries, helping users correct mistakes in real-time.

   Progressive Disclosure: In cases where the form contains a significant number of fields or sections, consider employing progressive disclosure techniques to present information gradually. This involves initially displaying only essential fields and allowing users to reveal additional sections as needed, minimizing cognitive overload and improving user experience.

   Data Formatting: Use appropriate HTML attributes such as type="tel" for phone numbers, type="email" for email addresses, and type="date" for date inputs to leverage built-in browser

validation and ensure consistent data formatting across different devices and browsers.
Placeholder Text: Utilize the placeholder attribute to provide hints or examples within input fields, guiding users on the expected format or content of their input. Placeholder text can help reduce ambiguity and improve form completion rates.
Autocomplete: Enable autocomplete functionality for input fields whenever possible to assist users in quickly filling out repetitive or commonly used information. This feature leverages the browser's autofill capabilities, enhancing user convenience and efficiency.
Form Accessibility: Ensure that the registration form complies with web accessibility standards (e.g., WCAG) to accommodate users with disabilities. This involves using semantic HTML elements, associating form labels with input fields using for and id attributes, and providing alternative text for non-text content like images or icons.
Multi-step Forms: For complex registration processes, consider breaking the form into multiple steps or pages to prevent overwhelming users with too much information at once. Each step should have clear navigation options and indicate the user's progress through the registration process.
Conditional Logic: Implement conditional logic using JavaScript to dynamically show or hide form fields based on user input or

selections. This helps streamline the form by only presenting relevant fields to the user, reducing clutter and improving usability.
Confirmation Page: After successfully submitting the form, redirect users to a confirmation page or display a confirmation message to acknowledge their submission and provide any additional instructions or next steps. This reassures users that their registration has been received and processed.
Browser Compatibility: Test the registration form across various web browsers (e.g., Chrome, Firefox, Safari, Edge) and devices (desktops, tablets, smartphones) to ensure consistent functionality and appearance. Address any compatibility issues or discrepancies to deliver a seamless experience for all users.

4. Gokul **(01SU23EC009)** – Html code

Dynamic Field Population: Implement functionality to dynamically populate certain form fields based on user input or selections. For example, when a user selects their country, dynamically load a dropdown menu of available states or provinces.
Address Autocomplete: Integrate with address lookup services or APIs to provide autocomplete suggestions as users type their address, reducing the time and effort required to input this information accurately.

Multi-language Support: Design the form to support multiple languages, allowing users to switch between different language options. This involves providing language-specific labels, messages, and instructions while ensuring proper text alignment and formatting for each language.
Client-side Data Encryption: Incorporate client-side encryption techniques to encrypt sensitive data (such as personal or medical information) before it is transmitted over the network. This enhances data security and privacy, protecting user information from unauthorized access or interception.
Data Masking: Implement data masking for fields like phone numbers or social security numbers to obscure sensitive information while maintaining usability. Data masking can help prevent unauthorized access to confidential data, even in cases where the form data is stored or transmitted insecurely.
Session Management: Manage user sessions securely to prevent unauthorized access or tampering with form data during the registration process. Implement session timeouts, CSRF (Cross-Site Request Forgery) protection, and secure session handling techniques to mitigate security risks.
Captcha Verification: Integrate CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) verification to prevent automated bots from submitting the

registration form. CAPTCHA challenges users to complete a task (such as identifying objects in images) to verify that they are human users.

User Consent and Privacy: Include checkboxes or consent checkboxes for users to acknowledge and agree to terms of service, privacy policies, or data processing agreements. Ensure transparency regarding how user data will be used, stored, and shared, and provide options for users to opt out of certain data processing activities if applicable.

Progress Indicators: Display progress indicators or step-by-step navigation to guide users through the registration process and indicate their current position within the form. This helps users understand the overall process and how far they have progressed.

Error Logging and Monitoring: Implement error logging and monitoring mechanisms to track and record any errors or issues encountered during form submissions. Log relevant details such as error messages, user interactions, and timestamps to facilitate troubleshooting and error resolution.

User Feedback Mechanisms: Include feedback mechanisms such as surveys, feedback forms, or rating systems to gather user feedback on their registration experience. Use this feedback to identify areas for improvement and make iterative enhancements to the form.

Browser Autofill Compatibility: Ensure compatibility with browser autofill features to support users who rely on autofill functionality to complete form fields. Test the form with autofill enabled and address any issues or conflicts that may arise with autofill behavior.
GDPR Compliance: Adhere to the General Data Protection Regulation (GDPR) requirements, especially if collecting personal data from users in the European Union. Implement features such as explicit consent for data processing, data access requests, and data deletion mechanisms to comply with GDPR regulations.
HIPAA Compliance: If collecting protected health information (PHI) covered by the Health Insurance Portability and Accountability Act (HIPAA), ensure compliance with HIPAA security and privacy requirements. Implement encryption, access controls, audit trails, and other security measures to safeguard PHI and maintain HIPAA compliance.

5. Goutham**(01SU23EC010)** –Submit page code

HTML Structure: The submit page typically consists of an HTML document structure similar to other web pages, including <!DOCTYPE html>, <html>, <head>, and <body> tags. However, the focus is on processing form submissions rather than displaying content.

Form Data Retrieval: In the <body> section, server-side scripting languages like PHP or Node.js are used to retrieve form data submitted via HTTP POST method. This involves accessing form fields using the corresponding names or IDs specified in the registration form.

Data Validation: Once the form data is retrieved, it is important to validate it to ensure accuracy and integrity. This may involve checking for required fields, validating input formats (e.g., email addresses, phone numbers), and sanitizing inputs to prevent security vulnerabilities such as SQL injection or XSS (Cross-Site Scripting) attacks.

Data Processing: After validation, the form data can be processed according to the requirements of the application. This may include storing the data in a database, sending email notifications, generating reports, or performing any other necessary actions based on the submitted information.

Error Handling: It is essential to handle any errors or exceptions that may occur during the data processing phase. This includes displaying informative error messages to the user if there are validation errors or if the submission process encounters any issues.

Success Message: Upon successful processing of the form data, a confirmation message or redirect can be displayed to the user to indicate that their submission was successful. This helps provide feedback to the user and assures them that their registration has been received and processed.

Security Considerations: Security is paramount when processing form submissions, especially when dealing with sensitive information such as personal or medical data. Implementing measures such as input validation, data encryption, and secure coding practices helps mitigate security risks and protect user privacy.

Logging and Monitoring: It is beneficial to log relevant information about form submissions for auditing, troubleshooting, and monitoring purposes. This may include logging successful submissions, errors, IP addresses, timestamps, and any other relevant details to facilitate analysis and debugging.

GDPR and HIPAA Compliance: If the submitted data includes personal or health information subject to regulations like GDPR or HIPAA, ensure compliance with relevant data protection and privacy requirements. This includes obtaining explicit consent for data processing, implementing data access controls, and maintaining data security measures.

Feedback Mechanisms: Providing feedback to the user about the status of their submission, whether successful or unsuccessful, helps improve the user experience. This may involve displaying success or error messages on the submit page or redirecting users to a dedicated confirmation page.

CONCLUSION

In conclusion, the development of a Hospital Registration Form is a multifaceted endeavor that requires careful consideration of various factors, including usability, security, compliance, and user experience. Throughout this exploration, we have delved into the intricacies of designing both the frontend interface using HTML, CSS, and JavaScript, as well as the backend functionality for processing form submissions. Here, we summarize the key insights and takeaways from this journey:

User-Centric Design: Central to the success of any registration form is a user-centric design approach. By prioritizing user experience and ensuring intuitive navigation, clear instructions, and accessibility features, we aim to create a seamless registration process that caters to the needs of patients and healthcare providers alike.

Data Integrity and Security: Given the sensitive nature of the information collected in a hospital registration form, safeguarding data integrity and security is paramount. Through robust validation mechanisms, encryption techniques, and adherence to regulatory requirements such as GDPR and HIPAA, we strive to protect user privacy and confidentiality.

Compliance and Regulation: Compliance with relevant regulations and standards, such as GDPR for data protection in the European Union and HIPAA for healthcare data privacy in the United States, is essential. By implementing features like explicit consent mechanisms, data access controls, and secure data handling practices, we ensure adherence to legal requirements and industry best practices.

Continuous Improvement: The development of a Hospital Registration Form is an iterative process that requires ongoing feedback, monitoring, and refinement. By soliciting user feedback, monitoring form performance, and staying abreast of emerging technologies and trends, we can continuously enhance the form's functionality, usability, and security.

Collaboration and Communication: Effective collaboration and communication among stakeholders, including healthcare administrators, IT professionals, regulatory experts, and end-users, are critical for the success of the project. By fostering open dialogue, addressing concerns, and aligning goals and objectives, we can ensure that the registration form meets the needs and expectations of all stakeholders involved.

Ethical Considerations: Throughout the development process, it is essential to uphold ethical principles and values, particularly concerning data privacy, consent, and transparency. By prioritizing ethical considerations and putting the interests of users first, we can build trust and credibility in the healthcare system and promote positive patient experiences.

Empowering Innovation: The development of a Hospital Registration Form presents an opportunity to leverage technology and innovation to improve healthcare delivery and patient outcomes. By harnessing the capabilities of HTML, CSS, JavaScript, and other web technologies, we can create dynamic, user-friendly solutions that streamline administrative processes and enhance the overall quality of care.

In essence, the development of a Hospital Registration Form is not merely about collecting data—it is about empowering patients, optimizing healthcare workflows, and fostering trust and confidence in the healthcare system. Through a combination of technical expertise, regulatory compliance, and a deep commitment to user-centric design, we can create registration forms that not only meet the functional requirements of healthcare institutions but also uphold the highest standards of security, privacy, and usability. As we continue to advance in the field of web development, let us remain steadfast in our dedication to building solutions that make a positive impact on the lives of patients and healthcare providers around the world.

References:

Bootstrap.com

W3schools.com