**Software Requirements Specification (SRS) for Comprehensive Form**

**1. Introduction**

**1.1. Purpose**

The purpose of this document is to specify the requirements for a web-based form that includes a variety of input types with functional and data validation features. The form will be used to collect and validate user information in compliance with predefined rules.

**1.2. Scope**

The comprehensive form will allow users to input details such as username, phone number, email, password, age, event date, address (conditionally), and CAPTCHA. The form will perform both client-side and server-side validations to ensure data integrity.

**1.3. Definitions, Acronyms, and Abbreviations**

* **SRS:** Software Requirements Specification
* **UI:** User Interface
* **UX:** User Experience

**1.4. References**

* HTML5 and JavaScript validation best practices
* W3C Web Content Accessibility Guidelines

**2. Overall Description**

**2.1. Product Perspective**

This web form is part of a larger web application that collects user data for authentication and event registration purposes. It will be a standalone component for the collection of validated input data.

**2.2. User Classes and Characteristics**

* **Registered Users**: Users who have a profile and need to update their information.
* **Guest Users**: Unregistered users filling in the form for the first time.

**2.3. Assumptions and Dependencies**

* The form will assume that JavaScript is enabled for client-side validation.
* The server-side will be responsible for storing validated data and further processing.

**3. Functional Requirements**

**3.1. Field Requirements**

| **Field** | **Type** | **Functional Description** | **Validation Rules** |
| --- | --- | --- | --- |
| Username | Text | Collects the user’s unique username. This is a mandatory field. | - Mandatory  - Default value: JohnDoe  - Cannot be empty or contain special characters |
| Phone Number | Text | An optional field to collect the user’s phone number. | - Optional  - Format should be: +1234567890 or 123-456-7890 |
| Email | Email | A required field for collecting a valid email address. | - Mandatory  - Must be a valid email format (e.g., user@example.com) |
| Password | Password | Password input with a minimum length requirement. | - Mandatory  - Minimum 6 characters  - Should include alphanumeric characters for strong passwords |
| Age | Number | A required numeric field that restricts user input between a minimum and maximum age range. | - Mandatory  - Minimum age: 18  - Maximum age: 100 |
| Event Date | Date | A date field with default value, restricted within a specific time range. | - Mandatory  - Default value: 2024-01-01  - Date range: 2024-01-01 to 2024-12-31 |
| Address | Text (Conditional) | Conditional text field for user address; only visible if the user selects the "Include Address" checkbox. | - Optional  - Cannot be empty if the "Include Address" checkbox is selected |
| CAPTCHA | Text | A CAPTCHA input field to ensure that the form is being submitted by a human. | - Mandatory  - User must input 1234 correctly |

**3.2. Functional Features**

* **Form Submission**:
  + The form will collect user input and, upon successful validation, trigger the form submission process.
  + The form will not be submitted if any validation fails.
* **Form Reset**:
  + A reset button will clear all input fields and error messages.
* **Dynamic Field Display**:
  + The "Address" field will be shown only if the user checks the "Include Address" checkbox.
* **Auto-Save Feature**:
  + A simulation of an auto-save feature will save the input of the Auto-Save Field into local storage every 5 seconds.
  + When the page is reloaded, the saved value will be populated in the field from local storage.

**4. Non-Functional Requirements**

**4.1. Performance Requirements**

* The form should load and validate data instantaneously (within 2 seconds on an average connection).
* Form validation (both on key-up and on submission) should occur without noticeable delay.

**4.2. Usability Requirements**

* The UI must be intuitive, with labels clearly indicating which fields are mandatory or optional.
* Clear error messages must be displayed beneath each field that fails validation.
* Fields should have appropriate default values where applicable.

**4.3. Security Requirements**

* The password input must mask characters for security purposes.
* Client-side validation should not be relied upon as the only defense against malformed data; server-side validation will be required for final verification.

**4.4. Scalability**

* The form should support multiple users concurrently accessing and submitting the form without performance degradation.

**4.5. Compatibility Requirements**

* The form must be compatible with all modern browsers, including Chrome, Firefox, Edge, and Safari.
* The form must be responsive and display properly on mobile, tablet, and desktop devices.

**5. Data Validation Requirements**

**5.1. Client-Side Validation**

* The form will perform client-side validation using JavaScript before submitting to the server. This includes checking for:
  + Empty fields (mandatory fields only)
  + Format constraints (email, phone, CAPTCHA)
  + Numeric constraints (age)
  + Conditional fields (address)

**5.2. Server-Side Validation**

* All client-side validations must also be enforced on the server to ensure data integrity.
* The server must reject any data that fails validation and return appropriate error messages to the client.

**5.3. Error Messages**

* Error messages should be specific to each field and guide the user on how to correct their input.
* Error messages must appear in red text beneath the associated field.

**6. Form Flow**

**6.1. Normal Flow**

1. The user fills out the form.
2. Upon clicking the "Submit" button, all input fields are validated.
3. If validation passes, the form is submitted successfully, and a success message is displayed.
4. If validation fails, the form remains on the same page, and error messages are shown below the invalid fields.

**6.2. Alternative Flow**

* If the form reset button is pressed, all fields are cleared, and error messages are reset.

**7. UI Mockups and Design**

**7.1. Form Layout**

* The form will have a simple, clean layout with clear labels and responsive input fields.
* Buttons will have hover effects to enhance user experience.

**8. Conclusion**

This document outlines the requirements for a comprehensive web form that ensures user input meets strict validation rules. The combination of dynamic field behavior, auto-save simulation, and a CAPTCHA placeholder ensures robust form functionality with good UX and security considerations.

This SRS covers both functional and validation requirements, ensuring that the form works properly under multiple scenarios.