Field Validation Rules Documentation

Overview

This document outlines the comprehensive validation rules implemented in the Wedding Services Intake Form. These rules ensure data quality, improve user experience, and maintain consistency across all form submissions.

Validation Types

1. Text Field Validation

Name Fields

Fields: Primary Contact, Secondary Contact, Electronic Signature

```
{
  "validation": {
    "lengthValidation": {
      "minLength": 2,
      "maxLength": 50
    }
  },
  "errorMessage": "Name must be between 2 and 50 characters"
}
```

Email Validation

Fields: Primary Contact Email, Secondary Contact Email

```
{
  "validation": {
    "regexValidation": {
      "pattern": "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$",
      "errorMessage": "Please enter a valid email address"
    }
  }
}
```

Phone Number Validation

Fields: Primary Contact Phone, Secondary Contact Phone

```
{
  "validation": {
    "regexValidation": {
      "pattern": "^[\\d\\s\\(\\)\\-\\+\\.]{10,}$",
      "errorMessage": "Please enter a valid phone number (minimum 10 digits)"
    }
  }
}
```

Accepted Formats:
- (555) 123-4567
- 555-123-4567
- 555.123.4567
- 555 123 4567
- +1 555 123 4567
- 5551234567

Venue Name Validation

```
{
  "validation": {
    "lengthValidation": {
      "minLength": 2,
      "maxLength": 100
    }
  },
  "errorMessage": "Venue name must be between 2 and 100 characters"
}
```

2. Address Validation

Venue Address

```
{
  "validation": {
    "lengthValidation": {
      "minLength": 10,
      "maxLength": 200
    }
  },
  "errorMessage": "Please provide a complete address (minimum 10 characters)"
}
```

Requirements:
- Must include street address
- Should include city and state/province
- Postal code recommended
- Country if international

3. Text Area Validation

Special Requests and Comments

Fields: Special Song Requests, Special Photography Requests, Additional Comments

```
{
  "validation": {
    "lengthValidation": {
      "maxLength": 500
    }
  },
  "errorMessage": "Please limit your response to 500 characters"
}
```

Requirements and Considerations

Fields: Accessibility Requirements, Cultural/Religious Considerations, Venue Restrictions

```
{
  "validation": {
    "lengthValidation": {
      "maxLength": 300
    }
  },
  "errorMessage": "Please limit your response to 300 characters"
}
```

Extended Comments

Fields: Additional Comments (final section)

```
{
  "validation": {
    "lengthValidation": {
      "maxLength": 1000
    }
  },
  "errorMessage": "Please limit your response to 1000 characters"
}
```

4. Date and Time Validation

Wedding Date

```
{
  "validation": {
    "dateValidation": {
      "futureDate": true,
      "minDate": "today",
      "maxDate": "2030-12-31"
    }
  },
  "errorMessage": "Wedding date must be in the future"
}
```

Business Rules:
- Date must be at least 30 days in the future (recommended)
- Maximum 5 years in advance
- Consider seasonal availability

Event Times

Fields: Event Start Time, Expected End Time

```
{
  "validation": {
    "timeValidation": {
      "format": "12-hour",
      "allowedRange": {
        "start": "06:00",
        "end": "23:59"
      }
    }
  },
  "errorMessage": "Please select a valid time between 6:00 AM and 11:59 PM"
}
```

5. Choice Field Validation

Required Single Choice

Fields: Communication Method, Wedding Date Flexibility, Service Packages

```
{
  "validation": {
    "choiceValidation": {
      "required": true,
      "minSelections": 1,
      "maxSelections": 1
    }
  },
  "errorMessage": "Please select one option"
}
```

Required Multiple Choice

Fields: Services Selection

```
{
  "validation": {
    "choiceValidation": {
      "required": true,
      "minSelections": 1,
      "maxSelections": 3
    }
  },
  "errorMessage": "Please select at least one service"
}
```

Optional Multiple Choice

Fields: Music Preferences, Key Photography Moments

```
{
  "validation": {
    "choiceValidation": {
      "required": false,
      "maxSelections": 10
    }
  },
  "errorMessage": "Please select no more than 10 options"
}
```

Advanced Validation Rules

1. Conditional Validation

Service-Dependent Fields

```
// DJ Services fields only required if DJ Services selected
if (servicesSelected.includes('DJ Services')) {
  djPackageField.required = true;
  musicPreferencesField.required = false; // Optional but recommended
}

// Photography fields only required if Photography selected
if (servicesSelected.includes('Photography')) {
  photographyPackageField.required = true;
  photographyStyleField.required = true;
}

// Videography fields only required if Videography selected
if (servicesSelected.includes('Videography')) {
  videographyPackageField.required = true;
  videoStyleField.required = true;
}
```

Budget Validation

```
// Validate budget against selected services
function validateBudget(selectedServices, budgetRange) {
  const minimumBudgets = {
    'DJ Services': 800,
    'Photography': 1200,
    'Videography': 1500
  };

  let minimumRequired = 0;
  selectedServices.forEach(service => {
    minimumRequired += minimumBudgets[service] || 0;
  });

  const budgetRanges = {
    'Under $2,000': 2000,
    '$2,000 - $3,500': 3500,
    '$3,500 - $5,000': 5000,
    '$5,000 - $7,500': 7500,
    '$7,500 - $10,000': 10000,
    'Over $10,000': 15000
  };

  const maxBudget = budgetRanges[budgetRange];

  if (maxBudget < minimumRequired) {
    return {
      valid: false,
      message: `Selected services typically require a minimum budget of $${minimumRe-
quired}. Please adjust your service selection or budget range.`
    };
  }

  return { valid: true };
}
```

2. Cross-Field Validation

Time Validation

```
function validateEventTimes(startTime, endTime) {
  const start = new Date(`2000-01-01 ${startTime}`);
  const end = new Date(`2000-01-01 ${endTime}`);

  if (end <= start) {
    return {
      valid: false,
      message: "End time must be after start time"
    };
  }

  const duration = (end - start) / (1000 * 60 * 60); // hours
  if (duration < 2) {
    return {
      valid: false,
      message: "Event duration must be at least 2 hours"
    };
  }

  if (duration > 12) {
    return {
      valid: false,
      message: "Event duration cannot exceed 12 hours"
    };
  }

  return { valid: true };
}
```

Contact Information Validation

```
function validateContactInfo(primaryEmail, secondaryEmail) {
  if (primaryEmail && secondaryEmail && primaryEmail === secondaryEmail) {
    return {
      valid: false,
      message: "Primary and secondary email addresses must be different"
    };
  }
  return { valid: true };
}
```

3. Business Logic Validation

Date Availability

```
function validateDateAvailability(weddingDate, selectedServices) {
  // Check against booking calendar
  const unavailableDates = getUnavailableDates();
  const dateString = weddingDate.toISOString().split('T')[0];

  if (unavailableDates.includes(dateString)) {
    return {
      valid: false,
      message: "Selected date is not available. Please choose an alternative date."
    };
  }

  // Check seasonal restrictions
  const month = weddingDate.getMonth() + 1;
  if (selectedServices.includes('Videography') && [12, 1, 2].includes(month)) {
    return {
      valid: false,
      message:
"Outdoor videography may be limited during winter months. Please contact us to discuss
options."
    };
  }

  return { valid: true };
}
```

Client-Side Validation Implementation

HTML5 Validation Attributes

```
<!-- Email field -->
<input type="email"
       required
       pattern="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"
       title="Please enter a valid email address">

<!-- Phone field -->
<input type="tel"
       required
       pattern="[\d\s\(\)\-\+\.]{10,}"
       title="Please enter a valid phone number">

<!-- Name field -->
<input type="text"
       required
       minlength="2"
       maxlength="50"
       title="Name must be between 2 and 50 characters">
```

JavaScript Validation

```
function validateForm() {
  const errors = [];

  // Validate required fields
  const requiredFields = document.querySelectorAll('[required]');
  requiredFields.forEach(field => {
    if (!field.value.trim()) {
      errors.push(`${field.labels[0].textContent} is required`);
    }
  });

  // Validate email format
  const emailFields = document.querySelectorAll('input[type="email"]');
  emailFields.forEach(field => {
    if (field.value && !isValidEmail(field.value)) {
      errors.push(`Please enter a valid email address for $
{field.labels[0].textContent}`);
    }
  });

  // Validate phone format
  const phoneFields = document.querySelectorAll('input[type="tel"]');
  phoneFields.forEach(field => {
    if (field.value && !isValidPhone(field.value)) {
      errors.push(`Please enter a valid phone number for ${field.labels[0].textContent}
`);
    }
  });

  // Display errors or submit form
  if (errors.length > 0) {
    displayErrors(errors);
    return false;
  }

  return true;
}

function isValidEmail(email) {
  const regex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
  return regex.test(email);
}

function isValidPhone(phone) {
  const regex = /^[\d\s\(\)\-\+\.]{10,}$/;
  return regex.test(phone);
}
```

Server-Side Validation (Google Apps Script)

Form Submission Validation

```
function validateFormSubmission(formResponse) {
  const errors = [];
  const responses = formResponse.getItemResponses();

  responses.forEach(response => {
    const item = response.getItem();
    const answer = response.getResponse();

    // Validate based on item type and title
    const validation = getValidationRules(item.getTitle());
    if (validation) {
      const result = validateField(answer, validation);
      if (!result.valid) {
        errors.push(`${item.getTitle()}: ${result.message}`);
      }
    }
  });

  // Cross-field validation
  const crossValidationErrors = performCrossFieldValidation(responses);
  errors.push(...crossValidationErrors);

  if (errors.length > 0) {
    // Send validation errors back to user
    sendValidationErrorEmail(formResponse.getRespondentEmail(), errors);
    return false;
  }

  return true;
}

function getValidationRules(fieldTitle) {
  const rules = {
    'Primary Contact Email': {
      type: 'email',
      required: true
    },
    'Primary Contact Phone': {
      type: 'phone',
      required: true
    },
    'Wedding Date': {
      type: 'date',
      required: true,
      futureOnly: true
    }
    // Add more rules as needed
  };

  return rules[fieldTitle];
}
```

Error Handling and User Feedback

Error Message Guidelines

1. **Clear and Specific**: Explain exactly what's wrong
2. **Actionable**: Tell users how to fix the issue
3. **Friendly Tone**: Maintain professional but helpful language
4. **Contextual**: Show errors near the relevant fields

Error Message Examples

```
const errorMessages = {
  email: {
    required: "Email address is required",
    invalid: "Please enter a valid email address (example: name@domain.com)",
    duplicate: "This email address is already in use"
  },
  phone: {
    required: "Phone number is required",
    invalid: "Please enter a valid phone number with at least 10 digits",
    format: "Please use format: (555) 123-4567 or 555-123-4567"
  },
  date: {
    required: "Wedding date is required",
    past: "Wedding date must be in the future",
    tooFar: "Wedding date cannot be more than 2 years in advance",
    unavailable: "Selected date is not available. Please choose another date."
  },
  services: {
    required: "Please select at least one service",
    budget: "Selected services may exceed your budget range. Please adjust your
selections or budget."
  }
};
```

Testing Validation Rules

Automated Testing

```
function testValidationRules() {
  const testCases = [
    {
      field: 'email',
      value: 'invalid-email',
      expected: false
    },
    {
      field: 'email',
      value: 'valid@email.com',
      expected: true
    },
    {
      field: 'phone',
      value: '123',
      expected: false
    },
    {
      field: 'phone',
      value: '(555) 123-4567',
      expected: true
    }
  ];

  testCases.forEach(testCase => {
    const result = validateField(testCase.value, getValidationRules(testCase.field));
    console.assert(result.valid === testCase.expected,
      `Test failed for ${testCase.field}: ${testCase.value}`);
  });
}
```

Manual Testing Checklist

- [ ] Test all required field validations
- [ ] Test email format validation with various inputs
- [ ] Test phone number validation with different formats
- [ ] Test text length limits (minimum and maximum)
- [ ] Test date validation (past dates, future dates)
- [ ] Test conditional field requirements
- [ ] Test cross-field validation rules
- [ ] Test error message display and clarity
- [ ] Test form submission with valid data
- [ ] Test form submission with invalid data

Accessibility Considerations

Screen Reader Support

```
<label for="email">Email Address <span aria-label="required">*</span></label>
<input type="email"
       id="email"
       required
       aria-describedby="email-error"
       aria-invalid="false">
<div id="email-error" role="alert" aria-live="polite"></div>
```

Keyboard Navigation

- Ensure all form fields are accessible via keyboard
- Provide clear focus indicators
- Support tab navigation in logical order
- Allow form submission via Enter key

Color and Contrast

- Don’t rely solely on color to indicate errors
- Use icons and text to supplement color coding
- Ensure sufficient contrast for error messages
- Test with color blindness simulators

This comprehensive validation system ensures data quality while maintaining an excellent user experience throughout the form completion process.