Conditional Logic and Branching Documentation

Overview

This document outlines the conditional logic implementation for the Wedding Services Intake Form. The branching logic ensures users only see relevant questions based on their service selections, creating a personalized and streamlined experience.

Conditional Logic Architecture

Primary Branching Point

The main conditional logic is triggered by the **“Which services are you interested in?”** question, which allows multiple selections:
- DJ Services
- Photography
- Videography

Branching Structure

```
Services Selection (Checkbox)
├── DJ Services Selected
│   └── DJ Services Details Section
│       ├── DJ Service Package
│       ├── Music Preferences
│       ├── Special Song Requests
│       └── MC Services Needed
├── Photography Selected
│   └── Photography Services Details Section
│       ├── Photography Package
│       ├── Photography Style Preference
│       ├── Key Photography Moments
│       └── Special Photography Requests
└── Videography Selected
    └── Videography Services Details Section
        ├── Videography Package
        ├── Video Style Preference
        ├── Key Video Moments
        └── Drone Footage Interest
```

Implementation Methods

Method 1: Google Forms Native Branching

Limitations of Native Branching

Google Forms native conditional logic has limitations:
- Only works with **Multiple Choice (Radio)** and **Dropdown** questions
- Does not support **Checkbox** questions for branching
- Cannot create complex multi-path logic

Workaround for Service Selection

Since we need checkbox functionality for service selection, we implement a hybrid approach:

1. **Service Selection Question**: Use checkbox for user experience
2. **Hidden Routing Questions**: Use multiple choice questions for branching logic

```
// Google Apps Script implementation
function createServiceRoutingLogic() {
  const form = FormApp.openById('YOUR_FORM_ID');

  // Create hidden routing questions based on checkbox selections
  const djRouting = form.addMultipleChoiceItem();
  djRouting.setTitle('DJ Services Routing (Hidden)');
  djRouting.setChoices([
    djRouting.createChoice('Show DJ Section', FormApp.PageNavigationType.GO_TO_PAGE, dj
Section),
    djRouting.createChoice('Skip DJ Section', FormApp.PageNavigationType.GO_TO_PAGE, ph
otographySection)
  ]);

  // Similar for Photography and Videography
}
```

Method 2: Google Apps Script Advanced Logic

Dynamic Form Generation

```
function createDynamicForm() {
  const form = FormApp.create('Wedding Services Intake - Dynamic');

  // Base sections that always appear
  createClientInfoSection(form);
  createEventDetailsSection(form);
  createServiceSelectionSection(form);

  // Conditional sections created based on responses
  // These will be shown/hidden via script processing
  createConditionalSections(form);
}

function createConditionalSections(form) {
  // DJ Services Section
  const djSection = form.addPageBreakItem();
  djSection.setTitle('DJ Services Details');

  const djPackage = form.addMultipleChoiceItem();
  djPackage.setTitle('DJ Service Package');
  djPackage.setChoices([
    'Basic Package - 4 hours, basic sound system, music only',
    'Standard Package - 6 hours, professional sound system, MC services',
    'Premium Package - 8 hours, premium sound system, MC services, lighting',
    'Deluxe Package - Full day, premium equipment, MC, lighting, special effects'
  ].map(choice => djPackage.createChoice(choice)));

  // Photography Section
  const photoSection = form.addPageBreakItem();
  photoSection.setTitle('Photography Services Details');

  // Videography Section
  const videoSection = form.addPageBreakItem();
  videoSection.setTitle('Videography Services Details');
}
```

Response-Based Form Modification

```
function onFormSubmit(e) {
  const responses = e.values;
  const servicesSelected = parseServiceSelections(responses);

  // Modify form based on selections
  modifyFormSections(servicesSelected);
}

function parseServiceSelections(responses) {
  // Find the services selection response (adjust index based on form structure)
  const servicesResponse = responses[15]; // Example index
  return servicesResponse ? servicesResponse.split(', ') : [];
}

function modifyFormSections(services) {
  const form = FormApp.openById('YOUR_FORM_ID');
  const items = form.getItems();

  // Show/hide sections based on selections
  services.forEach(service => {
    switch(service) {
      case 'DJ Services':
        showDJSection(form, items);
        break;
      case 'Photography':
        showPhotographySection(form, items);
        break;
      case 'Videography':
        showVideographySection(form, items);
        break;
    }
  });
}
```

Method 3: Multi-Form Approach

Separate Forms for Each Service

Create separate forms for detailed service information:

1. **Main Intake Form**: Basic info + service selection
2. **DJ Services Form**: Detailed DJ requirements
3. **Photography Form**: Detailed photography requirements
4. **Videography Form**: Detailed videography requirements

```
function createMultiFormSystem() {
  // Main form
  const mainForm = FormApp.create('Wedding Services - Main Intake');
  setupMainForm(mainForm);

  // Service-specific forms
  const djForm = FormApp.create('Wedding Services - DJ Details');
  const photoForm = FormApp.create('Wedding Services - Photography Details');
  const videoForm = FormApp.create('Wedding Services - Videography Details');

  // Link forms together
  linkForms(mainForm, djForm, photoForm, videoForm);
}

function linkForms(mainForm, djForm, photoForm, videoForm) {
  // Add links to service-specific forms based on selections
  const serviceSelection = mainForm.getItems().find(item =>
    item.getTitle().includes('Which services are you interested in?')
  );

  // Create follow-up section with links
  const followUpSection = mainForm.addPageBreakItem();
  followUpSection.setTitle('Complete Service Details');

  const instructions = mainForm.addTextItem();
  instructions.setTitle('Next Steps');
  instructions.setHelpText(`
    Based on your service selections, please complete the following detailed forms:

    DJ Services: ${djForm.getPublishedUrl()}
    Photography: ${photoForm.getPublishedUrl()}
    Videography: ${videoForm.getPublishedUrl()}
  `);
}
```

Advanced Conditional Logic Patterns

1. Package-Based Conditional Questions

DJ Package Conditional Logic

```
function setupDJPackageLogic(form) {
  const djPackageQuestion = form.addMultipleChoiceItem();
  djPackageQuestion.setTitle('DJ Service Package');

  // Create sections for each package level
  const basicSection = form.addPageBreakItem().setTitle('Basic Package Details');
  const standardSection = form.addPageBreakItem().setTitle('Standard Package Details');
  const premiumSection = form.addPageBreakItem().setTitle('Premium Package Details');
  const deluxeSection = form.addPageBreakItem().setTitle('Deluxe Package Details');

  // Set up branching
  djPackageQuestion.setChoices([
    djPackageQuestion.createChoice('Basic Package', Form-
App.PageNavigationType.GO_TO_PAGE, basicSection),
    djPackageQuestion.createChoice('Standard Package', Form-
App.PageNavigationType.GO_TO_PAGE, standardSection),
    djPackageQuestion.createChoice('Premium Package', Form-
App.PageNavigationType.GO_TO_PAGE, premiumSection),
    djPackageQuestion.createChoice('Deluxe Package', Form-
App.PageNavigationType.GO_TO_PAGE, deluxeSection)
  ]);

  // Package-specific questions
  setupBasicPackageQuestions(form, basicSection);
  setupStandardPackageQuestions(form, standardSection);
  setupPremiumPackageQuestions(form, premiumSection);
  setupDeluxePackageQuestions(form, deluxeSection);
}

function setupPremiumPackageQuestions(form, section) {
  // Questions specific to premium package
  const lightingOptions = form.addCheckboxItem();
  lightingOptions.setTitle('Lighting Preferences');
  lightingOptions.setChoices([
    'Uplighting',
    'Dance floor lighting',
    'Ambient lighting',
    'Color-changing lights',
    'Spotlights for special moments'
  ].map(choice => lightingOptions.createChoice(choice)));

  const specialEffects = form.addCheckboxItem();
  specialEffects.setTitle('Special Effects');
  specialEffects.setChoices([
    'Fog machine',
    'Bubble machine',
    'Confetti cannons',
    'Cold sparklers',
    'Laser lights'
  ].map(choice => specialEffects.createChoice(choice)));
}
```

2. Budget-Based Conditional Logic

Budget Validation and Recommendations

```
function setupBudgetConditionalLogic(form) {
  const budgetQuestion = form.addMultipleChoiceItem();
  budgetQuestion.setTitle('Total Budget Range');

  // Create different recommendation sections based on budget
  const lowBudgetSection = form.addPageBreakItem().setTitle('Budget-Friendly Options');
  const midBudgetSection = form.addPageBreakItem().setTitle('Standard Packages');
  const highBudgetSection = form.addPageBreakItem().setTitle('Premium Packages');

  budgetQuestion.setChoices([
    budgetQuestion.createChoice('Under $2,000', FormApp.PageNavigationType.GO_TO_PAGE,
lowBudgetSection),
    budgetQuestion.createChoice('$2,000 - $5,000', Form-
App.PageNavigationType.GO_TO_PAGE, midBudgetSection),
    budgetQuestion.createChoice('$5,000 - $10,000', Form-
App.PageNavigationType.GO_TO_PAGE, highBudgetSection),
    budgetQuestion.createChoice('Over $10,000', FormApp.PageNavigationType.GO_TO_PAGE,
highBudgetSection)
  ]);
}

function validateBudgetAgainstServices(responses) {
  const services = parseServiceSelections(responses);
  const budget = responses.find(r => r.includes('Budget Range'));

  const recommendations = generateBudgetRecommendations(services, budget);

  if (recommendations.warnings.length > 0) {
    // Send budget warning email
    sendBudgetWarningEmail(responses, recommendations);
  }
}
```

3. Venue-Based Conditional Logic

Venue Type Conditional Questions

```
function setupVenueConditionalLogic(form) {
  const venueTypeQuestion = form.addMultipleChoiceItem();
  venueTypeQuestion.setTitle('Venue Type');

  const indoorSection = form.addPageBreakItem().setTitle('Indoor Venue Details');
  const outdoorSection = form.addPageBreakItem().setTitle('Outdoor Venue Details');
  const mixedSection = form.addPageBreakItem().setTitle('Mixed Venue Details');

  venueTypeQuestion.setChoices([
    venueTypeQuestion.createChoice('Indoor - Banquet Hall', Form-
App.PageNavigationType.GO_TO_PAGE, indoorSection),
    venueTypeQuestion.createChoice('Indoor - Hotel', Form-
App.PageNavigationType.GO_TO_PAGE, indoorSection),
    venueTypeQuestion.createChoice('Outdoor - Garden/Park', Form-
App.PageNavigationType.GO_TO_PAGE, outdoorSection),
    venueTypeQuestion.createChoice('Outdoor - Beach', Form-
App.PageNavigationType.GO_TO_PAGE, outdoorSection),
    venueTypeQuestion.createChoice('Mixed - Indoor/Outdoor', Form-
App.PageNavigationType.GO_TO_PAGE, mixedSection)
  ]);

  // Venue-specific questions
  setupIndoorVenueQuestions(form, indoorSection);
  setupOutdoorVenueQuestions(form, outdoorSection);
}

function setupOutdoorVenueQuestions(form, section) {
  const weatherBackup = form.addMultipleChoiceItem();
  weatherBackup.setTitle('Weather Backup Plan');
  weatherBackup.setChoices([
    'Tent/Marquee available',
    'Indoor backup location',
    'Covered pavilion',
    'No backup plan needed',
    'Need assistance planning backup'
  ].map(choice => weatherBackup.createChoice(choice)));

  const powerAccess = form.addMultipleChoiceItem();
  powerAccess.setTitle('Power Access');
  powerAccess.setChoices([
    'Standard electrical outlets available',
    'Generator required',
    'Limited power access',
    'No power access',
    'Unsure about power availability'
  ].map(choice => powerAccess.createChoice(choice)));
}
```

Dynamic Question Generation

Service-Specific Question Sets

```
const questionSets = {
  djServices: {
    required: ['DJ Service Package'],
    optional: ['Music Preferences', 'Special Song Requests', 'MC Services Needed'],
    conditional: {
      'Premium Package': ['Lighting Preferences', 'Special Effects'],
      'Deluxe Package': ['Lighting Preferences', 'Special Effects', 'Additional Equip-
ment']
    }
  },
  photography: {
    required: ['Photography Package', 'Photography Style Preference'],
    optional: ['Key Photography Moments', 'Special Photography Requests'],
    conditional: {
      'Complete Package': ['Album Preferences', 'Print Selections'],
      'Premium Package': ['Additional Print Options']
    }
  },
  videography: {
    required: ['Videography Package', 'Video Style Preference'],
    optional: ['Key Video Moments', 'Drone Footage Interest'],
    conditional: {
      'Cinematic Package': ['Same-Day Edit Preferences', 'Special Video Effects'],
      'Premium Package': ['Drone Footage Options']
    }
  }
};

function generateDynamicQuestions(selectedServices, packageSelections) {
  const questions = [];

  selectedServices.forEach(service => {
    const serviceKey = service.toLowerCase().replace(' ', '');
    const questionSet = questionSets[serviceKey];

    if (questionSet) {
      // Add required questions
      questions.push(...questionSet.required);

      // Add optional questions
      questions.push(...questionSet.optional);

      // Add conditional questions based on package selection
      const selectedPackage = packageSelections[service];
      if (selectedPackage && questionSet.conditional[selectedPackage]) {
        questions.push(...questionSet.conditional[selectedPackage]);
      }
    }
  });

  return questions;
}
```

Response Processing and Logic

Post-Submission Logic Processing

```
function processConditionalLogic(formResponse) {
  const responses = formResponse.getItemResponses();
  const processedData = {};

  // Extract service selections
  const servicesResponse = responses.find(r =>
    r.getItem().getTitle().includes('Which services are you interested in?')
  );

  if (servicesResponse) {
    const selectedServices = servicesResponse.getResponse();
    processedData.services = Array.isArray(selectedServices) ? selectedServices : [se-
lectedServices];

    // Process each selected service
    processedData.services.forEach(service => {
      processedData[service] = extractServiceData(responses, service);
    });

    // Generate recommendations
    processedData.recommendations = generateRecommendations(processedData);

    // Calculate pricing
    processedData.pricing = calculatePricing(processedData);

    // Create follow-up tasks
    createFollowUpTasks(processedData);
  }

  return processedData;
}

function extractServiceData(responses, service) {
  const serviceData = {};
  const servicePrefix = service.toLowerCase().replace(' ', '');

  responses.forEach(response => {
    const title = response.getItem().getTitle().toLowerCase();

    if (title.includes(servicePrefix)) {
      const key = title.replace(servicePrefix, '').trim();
      serviceData[key] = response.getResponse();
    }
  });

  return serviceData;
}
```

Intelligent Recommendations

```
function generateRecommendations(processedData) {
  const recommendations = {
    packages: [],
    addOns: [],
    warnings: [],
    suggestions: []
  };

  // Analyze service combinations
  if (processedData.services.includes('DJ Services') &&
      processedData.services.includes('Photography')) {
    recommendations.suggestions.push('Consider our DJ + Photography combo package for
additional savings');
  }

  // Budget-based recommendations
  const budget = processedData.budget;
  const services = processedData.services;

  if (budget === 'Under $2,000' && services.length > 1) {
    recommendations.warnings.push('Your budget may be tight for multiple services. Con-
sider prioritizing your most important service.');
  }

  // Venue-based recommendations
  const venueType = processedData.venueType;
  if (venueType && venueType.includes('Outdoor')) {
    if (services.includes('DJ Services')) {
      recommendations.suggestions.push('Weather-resistant equipment recommended for
outdoor venues');
    }
    if (services.includes('Photography')) {

recommendations.suggestions.push('Golden hour photography opportunities available at
outdoor venues');
    }
  }

  return recommendations;
}
```

Testing Conditional Logic

Automated Testing Framework

```
function testConditionalLogic() {
  const testScenarios = [
    {
      name: 'DJ Services Only',
      services: ['DJ Services'],
      expectedSections: ['DJ Services Details'],
      expectedQuestions: ['DJ Service Package', 'Music Preferences']
    },
    {
      name: 'All Services',
      services: ['DJ Services', 'Photography', 'Videography'],
      expectedSections: ['DJ Services Details', 'Photography Services Details', 'Video-
graphy Services Details'],
      expectedQuestions: ['DJ Service Package', 'Photography Package', 'Videography
Package']
    },
    {
      name: 'Photography + Videography',
      services: ['Photography', 'Videography'],
      expectedSections: ['Photography Services Details', 'Videography Services De-
tails'],
      expectedQuestions: ['Photography Package', 'Videography Package']
    }
  ];

  testScenarios.forEach(scenario => {
    const result = simulateFormSubmission(scenario.services);
    validateTestResult(result, scenario);
  });
}

function simulateFormSubmission(services) {
  // Create mock form response
  const mockResponse = {
    services: services,
    // Add other mock data as needed
  };

  return processConditionalLogic(mockResponse);
}

function validateTestResult(result, expected) {
  console.log(`Testing: ${expected.name}`);

  // Validate services were processed correctly
  console.assert(
    JSON.stringify(result.services.sort()) === JSON.stringi-
fy(expected.services.sort()),
    `Services mismatch for ${expected.name}`
  );

  // Validate expected sections are present
  expected.expectedSections.forEach(section => {
    console.assert(
      result.sectionsShown && result.sectionsShown.includes(section),
      `Missing section: ${section} for ${expected.name}`
    );
  });

  console.log(`✓ Test passed: ${expected.name}`);
}
```

Manual Testing Checklist

Service Selection Testing

- [ ] Select DJ Services only → Shows DJ section only
- [ ] Select Photography only → Shows Photography section only
- [ ] Select Videography only → Shows Videography section only
- [ ] Select DJ + Photography → Shows both sections
- [ ] Select DJ + Videography → Shows both sections
- [ ] Select Photography + Videography → Shows both sections
- [ ] Select all three services → Shows all sections

Package-Based Testing

- [ ] DJ Basic Package → Shows basic questions only
- [ ] DJ Premium Package → Shows lighting and effects questions
- [ ] Photography Complete Package → Shows album and print questions
- [ ] Videography Cinematic Package → Shows advanced video questions

Budget-Based Testing

- [ ] Low budget + multiple services → Shows warning message
- [ ] High budget → Shows premium options
- [ ] Budget mismatch → Generates recommendations

Venue-Based Testing

- [ ] Outdoor venue → Shows weather backup questions
- [ ] Indoor venue → Skips outdoor-specific questions
- [ ] Mixed venue → Shows both indoor and outdoor questions

This comprehensive conditional logic system ensures that users have a personalized, relevant experience while maintaining data quality and completeness for business processing.