Google Forms Wedding Intake Setup Instructions

Prerequisites

Required Google Workspace Access

- Google Forms access (free Google account sufficient)
- Google Sheets access for data processing
- Google Docs access for contract generation
- Google Drive for file storage and sharing

Recommended Additional Tools

- Google Apps Script for automation
- Google Calendar for scheduling
- Gmail for client communication
- Google Sites for form embedding (optional)

Step 1: Create the Base Form

1.1 Initial Form Creation

1. Go to forms.google.com (https://forms.google.com)
2. Click the "+" button to create a new form
3. Title: "Wedding Services Intake Form"
4. Description: "Complete this form to request our DJ, Photography, and Videography services for your special day."

1.2 Form Settings Configuration

1. Click the gear icon (Settings) in the top right
2. Configure the following settings:
   - **General Tab**:
     - ✅ Collect email addresses
     - ✅ Allow response editing
     - ✅ See summary charts and text responses
     - **Presentation Tab**:
     - ✅ Show progress bar
     - ✅ Shuffle question order: OFF
     - Confirmation message: "Thank you for your submission! We'll review your information and contact you within 24 hours with your custom quote."
     - **Quizzes Tab**:
     - ❌ Make this a quiz (leave unchecked)

Step 2: Import Form Structure

2.1 Using Google Apps Script (Recommended)

1. Open script.google.com (https://script.google.com)
2. Create a new project
3. Replace the default code with the following:

```
function createWeddingIntakeForm() {
  // Read the JSON configuration
  const formConfig = {
    // Paste the content from intake_form.json here
  };

  // Create the form
  const form = FormApp.create(formConfig.info.title);
  form.setDescription(formConfig.info.description);

  // Configure settings
  form.setCollectEmail(formConfig.settings.collectEmail);
  form.setAllowResponseEdits(formConfig.settings.allowResponseEditing);
  form.setConfirmationMessage(formConfig.settings.confirmationMessage);

  // Add items from configuration
  formConfig.items.forEach(itemConfig => {
    addItemToForm(form, itemConfig);
  });

  Logger.log('Form created: ' + form.getEditUrl());
  return form;
}

function addItemToForm(form, itemConfig) {
  if (itemConfig.pageBreakItem) {
    const pageBreak = form.addPageBreakItem();
    pageBreak.setTitle(itemConfig.title);
    if (itemConfig.description) {
      pageBreak.setHelpText(itemConfig.description);
    }
  } else if (itemConfig.textItem) {
    const textItem = form.addTextItem();
    textItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      textItem.setHelpText(itemConfig.description);
    }
  } else if (itemConfig.questionItem) {
    addQuestionItem(form, itemConfig);
  }
}

function addQuestionItem(form, itemConfig) {
  const question = itemConfig.questionItem.question;

  if (question.textQuestion) {
    const textItem = form.addTextItem();
    textItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      textItem.setHelpText(itemConfig.description);
    }
    textItem.setRequired(question.required);

    // Add validation if specified
    if (question.textQuestion.validation) {
      addTextValidation(textItem, question.textQuestion.validation);
    }
  } else if (question.paragraphTextQuestion) {
    const paragraphItem = form.addParagraphTextItem();
    paragraphItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      paragraphItem.setHelpText(itemConfig.description);
    }
    paragraphItem.setRequired(question.required);
  } else if (question.choiceQuestion) {
    addChoiceQuestion(form, itemConfig, question);
  } else if (question.dateQuestion) {
    const dateItem = form.addDateItem();
    dateItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      dateItem.setHelpText(itemConfig.description);
    }
    dateItem.setRequired(question.required);
    dateItem.setIncludesYear(question.dateQuestion.includeYear);
  } else if (question.timeQuestion) {
    const timeItem = form.addTimeItem();
    timeItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      timeItem.setHelpText(itemConfig.description);
    }
    timeItem.setRequired(question.required);
  }
}

function addChoiceQuestion(form, itemConfig, question) {
  const choiceConfig = question.choiceQuestion;

  if (choiceConfig.type === 'RADIO') {
    const multipleChoiceItem = form.addMultipleChoiceItem();
    multipleChoiceItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      multipleChoiceItem.setHelpText(itemConfig.description);
    }
    multipleChoiceItem.setRequired(question.required);

    const choices = choiceConfig.options.map(option =>
      multipleChoiceItem.createChoice(option.value)
    );
    multipleChoiceItem.setChoices(choices);

    if (choiceConfig.hasOtherOption) {
      multipleChoiceItem.showOtherOption(true);
    }
  } else if (choiceConfig.type === 'CHECKBOX') {
    const checkboxItem = form.addCheckboxItem();
    checkboxItem.setTitle(itemConfig.title);
    if (itemConfig.description) {
      checkboxItem.setHelpText(itemConfig.description);
    }
    checkboxItem.setRequired(question.required);

    const choices = choiceConfig.options.map(option =>
      checkboxItem.createChoice(option.value)
    );
    checkboxItem.setChoices(choices);

    if (choiceConfig.hasOtherOption) {
      checkboxItem.showOtherOption(true);
    }
  }
}

function addTextValidation(textItem, validation) {
  if (validation.regexValidation) {
    const regexValidation = FormApp.createTextValidation()
```

```
      .setHelpText(validation.regexValidation.errorMessage)
      .requireTextMatchesPattern(validation.regexValidation.pattern)
      .build();
    textItem.setValidation(regexValidation);
  } else if (validation.lengthValidation) {
    if (validation.lengthValidation.minLength &&
validation.lengthValidation.maxLength) {
      const lengthValidation = FormApp.createTextValidation()
        .requireTextLengthBetween(
          validation.lengthValidation.minLength,
          validation.lengthValidation.maxLength
        )
        .build();
      textItem.setValidation(lengthValidation);
    }
  }
}
```

1. Run the `createWeddingIntakeForm()` function
2. Check the logs for the form edit URL

2.2 Manual Form Creation (Alternative)

If you prefer to create the form manually:

1. Follow the structure outlined in `intake_form.json`
2. Create each section as a page break
3. Add questions according to the specifications
4. Configure validation rules for each field
5. Set up conditional logic (see Step 4)

Step 3: Apply Branding and Styling

3.1 Theme Customization

1. In the form editor, click the palette icon (Theme)
2. **Header**: Upload your company logo
3. **Theme color**: Set to #8B4B8C (elegant purple)
4. **Background color**: Set to #FEFEFE (off-white)
5. **Font**: Choose a professional font combination

3.2 Custom Header Image

Create a header image with:
- Dimensions: 1600x400 pixels
- Company logo and branding
- Elegant wedding-themed background
- Contact information (optional)

3.3 Section Styling

For each page break (section):
1. Add descriptive titles
2. Include helpful descriptions
3. Use consistent formatting
4. Add progress indicators where helpful

Step 4: Configure Conditional Logic

4.1 Service-Based Branching

Set up conditional logic to show relevant sections based on service selection:

1. **Services Selection Question**: “Which services are you interested in?”
2. **Configure branching**:
   - If “DJ Services” selected → Show DJ Services Details section
   - If “Photography” selected → Show Photography Services Details section
   - If “Videography” selected → Show Videography Services Details section

4.2 Implementation Steps

1. Select the service selection question
2. Click the three dots menu → “Go to section based on answer”
3. Configure each answer option to direct to appropriate section
4. Set up “Continue to next section” for multi-service selections

4.3 Advanced Logic Setup

For complex conditional logic, use Google Apps Script:

```
function setupConditionalLogic() {
  const form = FormApp.openById('YOUR_FORM_ID');
  const items = form.getItems();

  // Find the service selection question
  const serviceQuestion = items.find(item =>
    item.getTitle().includes('Which services are you interested in?')
  );

  if (serviceQuestion && serviceQuestion.getType() === FormApp.ItemType.CHECKBOX) {
    // Set up conditional logic based on checkbox selections
    // This requires more complex handling in the response processing
  }
}
```

Step 5: Connect to Google Sheets

5.1 Create Response Spreadsheet

1. In the form editor, click the “Responses” tab
2. Click the green Sheets icon to create a spreadsheet
3. Name it “Wedding Services Intake Responses”
4. The spreadsheet will automatically populate with form responses

5.2 Set Up Data Processing

Create additional sheets for:
- **Calculations**: Pricing calculations based on selections
- **Client Summary**: Formatted client information
- **Contract Data**: Data formatted for contract generation
- **Follow-up Tasks**: Action items and scheduling

5.3 Automated Processing Script

```javascript
function onFormSubmit(e) {
  const responses = e.values;
  const sheet = e.range.getSheet();

  // Process the response data
  processClientData(responses);
  generateQuote(responses);
  sendConfirmationEmail(responses);
  createFollowupTasks(responses);
}

function processClientData(responses) {
  // Extract and format client information
  const clientData = {
    name: responses[2], // Adjust indices based on your form
    email: responses[3],
    phone: responses[4],
    weddingDate: responses[8],
    venue: responses[11],
    services: responses[15].split(', '),
    budget: responses[25]
  };

  // Save to processing sheet
  const processingSheet = SpreadsheetApp.getActiveSpreadsheet()
    .getSheetByName('Client Summary');
  processingSheet.appendRow([
    new Date(),
    clientData.name,
    clientData.email,
    clientData.phone,
    clientData.weddingDate,
    clientData.venue,
    clientData.services.join(', '),
    clientData.budget
  ]);
}
```

Step 6: Set Up Automated Workflows

6.1 Email Notifications

```
function sendConfirmationEmail(responses) {
  const clientEmail = responses[3]; // Adjust index
  const clientName = responses[2];

  const subject = 'Thank you for your wedding services inquiry!';
  const body = `
    Dear ${clientName},

    Thank you for submitting your wedding services intake form. We're excited about the opportunity to be part of your special day!

    We'll review your information and contact you within 24 hours with a personalized quote.

    In the meantime, feel free to browse our portfolio at [website] or contact us with any questions.

    Best regards,
    [Your Company Name]
  `;

  GmailApp.sendEmail(clientEmail, subject, body);
}
```

6.2 Internal Notifications

```
function sendInternalNotification(responses) {
  const adminEmail = 'admin@yourcompany.com';
  const clientName = responses[2];
  const services = responses[15];
  const budget = responses[25];

  const subject = `New Wedding Inquiry: ${clientName}`;
  const body = `
    New wedding services inquiry received:

    Client: ${clientName}
    Services: ${services}
    Budget: ${budget}

    View full response: [SPREADSHEET_URL]

    Action required: Generate quote within 24 hours
  `;

  GmailApp.sendEmail(adminEmail, subject, body);
}
```

6.3 Calendar Integration

```
function createFollowupTasks(responses) {
  const calendar = CalendarApp.getDefaultCalendar();
  const clientName = responses[2];
  const followupDate = new Date();
  followupDate.setDate(followupDate.getDate() + 1); // Next day

  calendar.createEvent(
    `Follow up: ${clientName} - Wedding Quote`,
    followupDate,
    new Date(followupDate.getTime() + 60*60*1000), // 1 hour duration
    {
      description: 'Generate and send wedding services quote',
      guests: 'admin@yourcompany.com'
    }
  );
}
```

Step 7: Testing and Quality Assurance

7.1 Form Testing Checklist

- [ ] All required fields properly marked
- [ ] Validation rules working correctly
- [ ] Conditional logic functioning as expected
- [ ] Mobile responsiveness verified
- [ ] Email notifications sending
- [ ] Spreadsheet integration working
- [ ] Branding applied consistently

7.2 Test Scenarios

1. **Complete Form Submission**: Fill out entire form with valid data
2. **Partial Submissions**: Test required field validation
3. **Invalid Data**: Test validation rules (email, phone, etc.)
4. **Service Combinations**: Test different service selections
5. **Mobile Testing**: Complete form on mobile device
6. **Email Testing**: Verify all automated emails

7.3 User Acceptance Testing

1. Have team members complete the form
2. Gather feedback on user experience
3. Test with actual clients (beta testing)
4. Make adjustments based on feedback

Step 8: Deployment and Monitoring

8.1 Form Sharing Options

1. **Direct Link**: Share the form URL directly
2. **Website Embedding**: Embed in your website

3. **QR Code**: Generate QR code for print materials
4. **Social Media**: Share on social platforms

8.2 Embedding in Website

```
<iframe
  src="https://docs.google.com/forms/d/e/YOUR_FORM_ID/viewform?embedded=true"
  width="100%"
  height="2000"
  frameborder="0"
  marginheight="0"
  marginwidth="0">
  Loading…
</iframe>
```

8.3 Monitoring and Analytics

1. **Response Tracking**: Monitor submission rates
2. **Completion Rates**: Track form abandonment
3. **Error Monitoring**: Watch for validation issues
4. **Performance**: Monitor form loading times

8.4 Regular Maintenance

- **Monthly**: Review and update pricing options
- **Quarterly**: Update service packages and options
- **Annually**: Refresh branding and design
- **As needed**: Add new services or modify existing ones

Troubleshooting Common Issues

Form Not Submitting

- Check required field validation
- Verify internet connection
- Clear browser cache
- Test in different browsers

Conditional Logic Not Working

- Verify section names match exactly
- Check question types (only multiple choice and dropdown support branching)
- Test all possible paths

Email Notifications Not Sending

- Check Gmail API permissions
- Verify email addresses are correct
- Check spam folders
- Review Apps Script execution logs

Spreadsheet Integration Issues

- Verify spreadsheet permissions
- Check column mapping

- Review Apps Script triggers
- Test with manual form submission

Support and Resources

Google Forms Documentation

- Google Forms Help Center (https://support.google.com/docs/topic/9055404)
- Google Apps Script Documentation (https://developers.google.com/apps-script)
- Google Sheets API (https://developers.google.com/sheets/api)

Additional Resources

- Form design best practices
- Wedding industry specific considerations
- GDPR compliance for client data
- Accessibility guidelines for forms

For technical support, contact your Google Workspace administrator or refer to the Google Workspace support resources.