✦ Great. Now that generateFormBuilderQuestions is updated, let's work on the logic that uses it. Here's what you need to do next:

1. Create the System Prompt

This is the most important step. The system prompt will guide the AI's behavior. You should store this prompt in your app\_localizations files to support multiple

languages.

Here is the recommended system prompt:

1 You are an intelligent form building assistant. Your goal is to help the user create a form by asking clarifying questions.

2

3 The user will start by describing the form they want to create.

4

5 Your task is to guide the user through the process of creating the form. You should ask questions one at a time to understand the requirements for each form

field.

6

7 For each field, you need to determine:

8 1. The question to ask in the form (e.g., "What is your name?").

9 2. The type of input field (e.g., "text", "multiple choice", "checkboxes", "dropdown").

10 3. The options for multiple choice, checkboxes, or dropdown fields.

11

12 Once you have gathered all the necessary information and the user confirms that the form is complete, you must output the form definition as a single JSON

object. Do not include any other text in your response, only the JSON object.

13

14 The JSON object should have the following structure:

15 {

16 "title": "The title of the form",

17 "fields": [

18 {

19 "question": "The question for the first form field",

20 "type": "The type of the first form field",

21 "options": ["Option 1", "Option 2"]

22 },

23 {

24 "question": "The question for the second form field",

25 "type": "The type of the second form field"

26 }

27 ]

28 }

29

30 Here are the supported field types: "text", "textarea", "multiple-choice", "checkboxes", "dropdown", "date", "number".

31

32 Let's start. What is the title of your form and what is the first question you would like to ask?

2. Update ai\_form\_builder\_controller.dart

Now, you need to modify the addAiFormBuilderChat function in your AiFormBuilderChatController to pass the chat history and the new system prompt to the

generateFormBuilderQuestions function. You also need to handle the AI's response.

Here's the conceptual logic for your addAiFormBuilderChat function:

1 // In ai\_form\_builder\_controller.dart

2

3 Future<void> addAiFormBuilderChat(String usersText) async {

4 // ... (your existing code to add the user's message)

5

6 try {

7 final mistralService = ref.read(mistralServiceProvider);

8

9 // 1. Get the chat history from the state

10 final chatHistory = state.value?.map((chat) {

11 return [

12 {"role": "user", "content": chat.chatTextBody ?? ""},

13 {"role": "assistant", "content": chat.replyText ?? ""}

14 ];

15 }).expand((element) => element).toList() ?? [];

16

17 // 2. Get the system prompt from your localizations

18 final systemPrompt =

19 AppLocalizations.of(context)!.formBuilderSystemPrompt; // You'll need to add this

20

21 // 3. Call the new generateFormBuilderQuestions function

22 final aiReplyText = await mistralService.generateFormBuilderQuestions(

23 usersText,

24 chatHistory,

25 systemPrompt,

26 );

27

28 // 4. Handle the AI's response

29 try {

30 // Try to decode the response as JSON

31 final decodedJson = jsonDecode(aiReplyText);

32 if (decodedJson is Map<String, dynamic>) {

33 // It's a form!

34 // Here you will parse the JSON and save it to Hive

35 // For example:

36 final form = AiGeneratedFormModel.fromJson(decodedJson);

37 // Now save the form using your repository

38 // ...

39

40 // You can then generate a link and display it to the user

41 final formLink = "https://your-app.com/form/${form.id}";

42 final updatedMessage = usersMessage.copyWith(

43 replyText: "Your form has been created! You can access it here: $formLink",

44 isReplied: true,

45 isSeen: true,

46 );

47 await \_repo.updateAiFormBuilderChat(usersMessage.id!, updatedMessage);

48 // ... (update state)

49 }

50 } catch (e) {

51 // It's not JSON, so it's a follow-up question

52 final updatedMessage = usersMessage.copyWith(

53 replyText: aiReplyText,

54 isReplied: true,

55 isSeen: true,

56 );

57 await \_repo.updateAiFormBuilderChat(usersMessage.id!, updatedMessage);

58 // ... (update state)

59 }

60 } catch (e, s) {

61 // ... (your error handling)

62 }

63 }

3. Generating the Form Link

Once the form is saved in Hive, you will have a unique ID for it. You can use this ID to create a unique URL for the form. For example:

https://your-app.com/form/<form-id>.

You will need to set up a new page in your app that takes a form ID as a parameter, fetches the form data from Hive, and dynamically builds and displays the form. You

can use the go\_router package to manage the routing to this new page.

These changes will enable a conversational form-building experience in your app. Let me know if you have any questions.