**🔰**

**PHASE 1: Zaarvis Basic Voice Assistant with Memory**

Features:

* Custom wake word (e.g., “Hey Zaarvis”)
* Voice input (English & Bangla)
* Text-to-speech reply (male voice)
* Memory system (local storage of preferences)
* Avatar display with soundwave animation
* Basic chatbot-style interaction

**📁**

**Part 1: Project Setup (Flutter)**

Bash:

flutter create zaarvis\_ai

cd zaarvis\_ai

**📄**

**pubspec.yaml**

**(Dependencies)**

dependencies:

flutter:

sdk: flutter

speech\_to\_text: ^5.5.0

flutter\_tts: ^4.0.2

avatar\_glow: ^2.0.2

permission\_handler: ^11.0.1

shared\_preferences: ^2.2.2

provider: ^6.0.5

flutter\_sound: ^9.2.13

just\_audio: ^0.9.35

assets\_audio\_player: ^3.0.3+8

flutter\_local\_notifications: ^17.1.0

**📄 main.dart**

import 'package:flutter/material.dart';

import 'package:zaarvis\_ai/screens/home\_screen.dart';

void main() {

runApp(ZaarvisApp());

}

class ZaarvisApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Zaarvis AI',

theme: ThemeData.dark(),

home: HomeScreen(),

debugShowCheckedModeBanner: false,

);

}

}

**📁 Folder Structure**

**/lib**

**├── main.dart**

**├── screens/**

**│ └── home\_screen.dart**

**├── services/**

**│ ├── voice\_input.dart**

**│ ├── voice\_output.dart**

**│ └── memory\_service.dart**

**├── widgets/**

**│ └── avatar\_display.dart**

**📄 home\_screen.dart (Initial Voice UI)**

**import 'package:flutter/material.dart';**

**import '../services/voice\_input.dart';**

**import '../services/voice\_output.dart';**

**import '../widgets/avatar\_display.dart';**

**class HomeScreen extends StatefulWidget {**

**@override**

**\_HomeScreenState createState() => \_HomeScreenState();**

**}**

**class \_HomeScreenState extends State<HomeScreen> {**

**String \_response = 'Hey! I am Zaarvis. How can I help you today?';**

**void handleVoiceCommand(String command) async {**

**setState(() {**

**\_response = "Processing: $command";**

**});**

**// Simple logic for reply**

**if (command.toLowerCase().contains("how are you")) {**

**\_response = "I'm always ready to assist you, Shojib Bhai!";**

**} else {**

**\_response = "Sorry, I didn't understand that.";**

**}**

**await VoiceOutput.speak(\_response);**

**setState(() {});**

**}**

**@override**

**Widget build(BuildContext context) {**

**return Scaffold(**

**body: Center(**

**child: Column(**

**mainAxisAlignment: MainAxisAlignment.center,**

**children: <Widget>[**

**AvatarDisplay(),**

**SizedBox(height: 20),**

**ElevatedButton(**

**onPressed: () => VoiceInput.listen(handleVoiceCommand),**

**child: Text("🎙️ Speak"),**

**),**

**Padding(**

**padding: EdgeInsets.all(16.0),**

**child: Text(**

**\_response,**

**style: TextStyle(fontSize: 18),**

**textAlign: TextAlign.center,**

**),**

**),**

**],**

**),**

**),**

**);**

**}**

**}**

**📄 voice\_input.dart**

import 'package:speech\_to\_text/speech\_to\_text.dart';

class VoiceInput {

static final \_speech = SpeechToText();

static Future<void> listen(Function(String) onResult) async {

bool available = await \_speech.initialize();

if (available) {

\_speech.listen(

onResult: (val) {

if (val.finalResult) {

onResult(val.recognizedWords);

\_speech.stop();

}

},

);

}

}

}

**📄 voice\_output.dart**

import 'package:flutter\_tts/flutter\_tts.dart';

class VoiceOutput {

final FlutterTts tts = FlutterTts();

Future<void> speak(String message, {String languageCode = 'en-US'}) async {

try {

await tts.setLanguage(languageCode);

await tts.setPitch(1.0);

await tts.setSpeechRate(0.5);

await tts.speak(message);

} catch (e) {

// Fallback to Azure or notify user

print('TTS error: $e');

}

}

}

**📄 avatar\_display.dart**

import 'package:flutter/material.dart';

import 'package:avatar\_glow/avatar\_glow.dart';

class AvatarDisplay extends StatelessWidget {

@override

Widget build(BuildContext context) {

return AvatarGlow(

endRadius: 90.0,

glowColor: Colors.blue,

child: CircleAvatar(

backgroundColor: Colors.black,

radius: 50.0,

child: Image.asset('assets/zaarvis\_avatar.png'),

),

);

}

}

**📄 memory\_service.dart (Shared Preferences)**

import 'package:shared\_preferences/shared\_preferences.dart';

class MemoryService {

static Future<void> saveName(String name) async {

final prefs = await SharedPreferences.getInstance();

prefs.setString('username', name);

}

static Future<String?> getName() async {

final prefs = await SharedPreferences.getInstance();

return prefs.getString('username');

}

}

**🔔 Phase 2: Wake‑Word + Multi‑User Identity + Advanced Memory**

**✅ 1. Wake‑Word Detection**

Use Porcupine for on-device wake‑word spotting (Hey Zaarvis). This runs in the background and turns on listening mode.

**📄 pubspec.yaml**

**additions:**

dependencies:

porcupine\_flutter: ^1.2.0

**📄 wake\_word\_service.dart:**

import 'package:porcupine\_flutter/porcupine\_flutter.dart';

import 'voice\_input.dart';

class WakeWordService {

PorcupineManager? \_manager;

Future<void> init() async {

\_manager = await PorcupineManager.fromKeywordPaths(

keywordPaths: ["assets/hey\_zaarvis.ppn"],

onDetection: (keywordIndex) {

// Wake word heard → start listening

VoiceInput.listen((cmd) {

print("Listen after wake word: $cmd");

});

},

);

}

void start() => \_manager?.start();

void stop() => \_manager?.stop();

}

**🎯 Usage in**

**main.dart:**

final wakeService = WakeWordService();

await wakeService.init();

wakeService.start();

**👥 2. Multi‑User Voice Identity**

Detect different user voices and store their profiles, so Zaarvis greets them personally.

**📄 voice\_identity\_service.dart:**

import 'package:shared\_preferences/shared\_preferences.dart';

class VoiceIdentityService {

static const \_userMapKey = 'userVoiceMap';

// Simple voice-to-user mapping (e.g., STT-based keyword match)

static Future<String> identifyUser(String speech) async {

final prefs = await SharedPreferences.getInstance();

final map = prefs.getStringList(\_userMapKey) ?? [];

for (var entry in map) {

var parts = entry.split('|');

if (speech.contains(parts[0])) return parts[1];

}

return "unknown";

}

static Future<void> addUser(String trigger, String username) async {

final prefs = await SharedPreferences.getInstance();

final map = prefs.getStringList(\_userMapKey) ?? [];

map.add("$trigger|$username");

prefs.setStringList(\_userMapKey, map);

}

}

**🎯 Integrate with command handler:**

VoiceInput.listen((cmd) async {

var user = await VoiceIdentityService.identifyUser(cmd);

if (user != "unknown") {

await VoiceOutput.speak("Welcome back, $user!");

} else {

await VoiceOutput.speak("Hello! Who is speaking?");

}

});

**🧠 3. Advanced Memory (Per‑User Preferences & History)**

Store user-specific data like preferences or conversation history.

**📄 user\_memory\_service.dart:**

import 'package:shared\_preferences/shared\_preferences.dart';

class UserMemoryService {

static Future<void> savePreference(String user, String key, String value) async {

final prefs = await SharedPreferences.getInstance();

prefs.setString("${user}\_$key", value);

}

static Future<String?> getPreference(String user, String key) async {

final prefs = await SharedPreferences.getInstance();

return prefs.getString("${user}\_$key");

}

}

**🎯 Example usage:**

// After identify:

await UserMemoryService.savePreference(user, "greetingStyle", "formal");

var style = await UserMemoryService.getPreference(user, "greetingStyle");

**🛠️ Putting It All Together**

**📄 home\_screen.dart updates:**

// after wake word

VoiceInput.listen((cmd) async {

var user = await VoiceIdentityService.identifyUser(cmd);

if (user == "unknown") {

await VoiceOutput.speak("I didn't get your name. Who’s speaking?");

} else {

await VoiceOutput.speak("Welcome back, $user!");

}

// Save last command

await UserMemoryService.savePreference(user, "lastCmd", cmd);

});

**✔️ Phase 2 Summary**

| **Feature** | **Code File** |
| --- | --- |
| Wake‑Word Detection | wake\_word\_service.dart |
| Multi‑User Identity | voice\_identity\_service.dart |
| Per‑User Memory | user\_memory\_service.dart |
| Integration | Updated home\_screen.dart |

**👁️ Phase 3: Surveillance Mode + Emotion + Cloud Backup**

**🔭 1. Surveillance Mode with Dual Camera Monitoring**

Goal: Let Zaarvis monitor two cameras (e.g. Door & Living Room), detect motion or sound, and act (greet, record, notify).

**📄 surveillance\_service.dart:**

import 'package:camera/camera.dart';

import 'package:motion\_sensors/motion\_sensors.dart'; // or any ML motion detection

class SurveillanceService {

CameraController? doorCamera;

CameraController? roomCamera;

Future<void> init(List<CameraDescription> cameras) async {

doorCamera = CameraController(cameras[0], ResolutionPreset.medium);

roomCamera = CameraController(cameras[1], ResolutionPreset.medium);

await doorCamera?.initialize();

await roomCamera?.initialize();

}

void startMonitoring() {

// Simulate motion detection

Timer.periodic(Duration(seconds: 2), (timer) {

bool motionDetected = Random().nextBool(); // Replace with ML-based detection

if (motionDetected) {

print("⚠️ Motion detected!");

// Add action: greet, record, notify

}

});

}

Future<void> dispose() async {

await doorCamera?.dispose();

await roomCamera?.dispose();

}

}

**🎯 In main.dart:**

**final cameras = await availableCameras();**

**final surveillance = SurveillanceService();**

**await surveillance.init(cameras);**

**surveillance.startMonitoring();**

**😐 2. Emotion Detection via Voice Tone**

Goal: Detect emotions from voice (sad, angry, excited) and respond accordingly.

**✅ Step-by-step:**

1. Use flutter\_sound or recorder\_wav to capture voice input as WAV/PCM.
2. Send audio to a tiny TensorFlow Lite model (or mock logic) to detect emotion.
3. Respond differently based on emotion.

**📄 emotion\_detector\_service.dart:**

**class EmotionDetectorService {**

**static String detectEmotion(String speech) {**

**if (speech.contains("why") || speech.contains("sad") || speech.contains("tired")) return "sad";**

**if (speech.contains("great") || speech.contains("awesome")) return "happy";**

**if (speech.contains("angry") || speech.contains("hate")) return "angry";**

**return "neutral";**

**}**

**static Future<void> respondEmotionally(String text) async {**

**final emotion = detectEmotion(text);**

**switch (emotion) {**

**case "sad":**

**await VoiceOutput.speak("I'm here for you, bhai. Everything will be okay.");**

**break;**

**case "happy":**

**await VoiceOutput.speak("That's amazing to hear!");**

**break;**

**case "angry":**

**await VoiceOutput.speak("Take a deep breath. I'm here to help.");**

**break;**

**default:**

**await VoiceOutput.speak("Go on, I'm listening.");**

**}**

**}**

**}**

**🎯 Usage:**

**VoiceInput.listen((cmd) async {**

**await EmotionDetectorService.respondEmotionally(cmd);**

**});**

**☁️ 3. Cloud Backup of Preferences & History**

Goal: Keep all your memories, logs, preferences backed up to Firebase.

**✅ Add Firebase to Flutter:**

1. In Firebase Console: Create project → Android app → Get google-services.json
2. Add dependencies:

**📄 pubspec.yaml:**

dependencies:

firebase\_core: ^2.0.0

cloud\_firestore: ^4.0.0

**📄 cloud\_backup\_service.dart:**

import 'package:cloud\_firestore/cloud\_firestore.dart';

class CloudBackupService {

final \_firestore = FirebaseFirestore.instance;

Future<void> backupUserData(String user, Map<String, String> data) async {

await \_firestore.collection("user\_data").doc(user).set(data);

}

Future<Map<String, dynamic>?> restoreUserData(String user) async {

var doc = await \_firestore.collection("user\_data").doc(user).get();

return doc.data();

}

}

**🎯 Usage:**

await CloudBackupService().backupUserData("Shojib", {

"greetingStyle": "friendly",

"lastCommand": "Turn on fan",

});

**✅ Phase 3 Summary**

| **Feature** | **File** |
| --- | --- |
| Surveillance Mode | surveillance\_service.dart |
| Emotion Detection | emotion\_detector\_service.dart |
| Cloud Backup | cloud\_backup\_service.dart |