**SOME PROBABILITY ASSKED QNS ON OUR 1ST PRESENTATION**

**✅ 1. What is Agile Development Approach?**

**Definition:**  
Agile is a software development approach where the system is built in small parts (called iterations or sprints) and improved step by step based on feedback.

**Why I chose it in my project:**  
Because our system is built step by step — we first work on frontend, then backend, then devices. Agile allows us to:

* Test each part early
* Improve based on user/supervisor feedback
* Work flexibly as a student group
* Your team is working in stages (e.g., frontend first, then backend, then hardware).
* It's flexible — if you face issues or get feedback from your supervisor, you can easily improve the system.
* It's perfect for small student teams working on real, evolving projects.

✅ **Summary answer:**“Agile is good for us because we can build the system in parts, test them early, and improve them step by step. This fits our project workflow well.

**Bonus Comparison:**

Waterfall needs everything to be planned from the start and is hard to change. But Agile is flexible — that’s why we preferred it.

## ✅ 2. ****What is Alpha Testing?****

**Definition:**  
Alpha Testing is the **first level of testing** done by the developers themselves to check the system's functionality **before** it’s given to real users.

**Why I chose it in my project:**  
Because we’re still developing the system and only testing it ourselves for now — to catch errors early before showing it to others like lecturers.

* Since your group will test the system yourselves before demo day.
* You want to make sure it works properly **before** others (like the supervisor or demo panel) use it.

✅ **Summary answer:**

“We chose Alpha Testing because we are testing the system ourselves first — to make sure it works well before showing it to others.”

**Bonus Comparison:**

Beta Testing is done by real users after the system is complete. But our system is still in progress, so Alpha Testing is more suitable.

## ✅ 3. ****What is Object-Oriented Design?****

**Definition:**  
Object-Oriented Design is a way of designing software where you think in terms of **real-world objects** (like User, Device, Admin). Each object has its **data** and **functions**.

**Why You Chose It for Your Project:**

* Your system has clear objects: **Users**, **Devices**, **Logs**, **Admins**.
* It helps you organize your system using Django models and reusable code.

**Example:**  
Each device has properties (name, status) and actions (turn on/off), which fits OOD perfectly.

✅ **Summary answer:**

“We used Object-Oriented Design because our system has clear objects like User and Device. It helps us design clean and organized code.”

## ✅ 4. ****What is a Smart Switching System?****

**Definition:**  
A system that lets users **turn electrical devices ON or OFF remotely**, often using a **web or mobile interface**, and sometimes also voice or automation.

**Why It’s Useful:**

* Saves electricity
* Adds convenience
* Enables remote control
* Makes homes smarter

**Why I chose it in my project:**  
Because many users forget to turn off fans/lights. This system will help them control devices from anywhere — saving energy and adding convenience.

## ✅ 5. ****Why Web-Based and Not Mobile App?****

**Definition:**  
Web-based systems work in a browser (on phone or laptop), while mobile apps require installation and often need separate versions for Android/iOS.

**Why I chose it in my project:**

* A web-based system is easier and faster to build for a student project.
* It works on both phones and laptops without needing to install an app.
* Adding voice control using JavaScript (Web Speech API) is easier than using Android or iOS voice systems.

**Summary answer:**

* “Web-based control is faster to build and works on any device without installation. Also, voice control using Web Speech API is easy to integrate into web pages.”

## ✅ 6. ****What is Voice Control and Why Did You Use It?****

**Definition:**  
Voice control allows users to give commands (like “Turn on fan”) using speech, and the system converts that to an action.

**Why I chose it in my project:**

* Makes the system more modern and user-friendly
* Helps users who may find clicking buttons harder
* Easy to add using the Web Speech API in JavaScript

## ✅ 7. ****What is IoT (Internet of Things)?****

**Definition:**  
IoT is a technology where **devices like lights, fans, sensors** are connected to the **internet**, and can be **monitored or controlled remotely**.

**Why I used it in this project:**

* It allows real hardware (fans/lights) to be controlled over Wi-Fi
* ESP8266 microcontroller helps make this possible
* Makes the system smarter and more useful

## ✅ 8. ****What Did You Learn from Your Literature Review?****

**Definition:**  
A literature review is reading past research papers to understand what other people did and what was missing in their systems.

**What I read:**

* Smart Switch Using IoT – Found they lacked multi-user access and device naming
* Smart Home with ESP8266 – They had no voice control or usage logs
* Bluetooth-Based Systems – Had short range and no remote access

**Why this helped:**  
It helped us **identify gaps**, like:

* No dashboards
* No voice command
* No ability to name devices

So, we added those in our system.

## ✅ 9. ****What Did You Learn from the Google Form?****

**Definition:**  
We used a Google Form to gather real user needs for our project.

**Findings:**

* Most users forget to turn off lights or fans
* Many want remote access from phone
* Voice control was highly requested
* Some asked for device history (ON/OFF time)

**Why it's important:**  
This helped us design the system **based on real user needs**, not just assumptions

## ✅ 10. ****Bonus Trick Question: How is Your System Better Than Others?****

**Answer:**

“Our system improves on existing ones by allowing both voice and web control, showing usage history, supporting multiple users, and allowing custom device names. Most past systems did not offer all of these together.”

another question may be asked is to give overview of our system and how it gonna work based on user as well as it implementation

**✅ 11. What is Automation?**

**Definition:**  
Automation is the ability of a system to **perform actions automatically without human input**, often based on conditions (like time, motion, or sensor input).

**Why it's relevant to your project:**  
Even though your current project focuses on manual control (via web or voice), it’s the **foundation for future automation** — for example, turning off lights if no one is in the room.

✅ **Answer:**

“Automation means the system can control devices without needing user commands. While our current system is manual (voice/web), we plan to add automation in the future like turning off a fan after a set time.”

**✅ 1. What are the challenges you expect to face in this project?**

**Answer:**

“Some of the challenges we might face include:

* 🔌 **Connecting hardware to the web interface** — getting Django (Python) to talk to ESP8266 reliably.
* 🌐 **Delay in communication** — sometimes commands might take time to reach the device.
* 👨‍💻 **Learning curve** — since we’re using multiple technologies (Django, HTML, JavaScript, Arduino).
* 🎙 **Voice command accuracy** — it may not understand some accents or background noise.”

### ✅ 2. **How are you planning to solve those challenges?**

**Answer:**

* For hardware–web communication, we’ll use **HTTP requests** with the ESP8266 acting as a web server to receive commands.
* To reduce delays, we’ll keep the system simple and ensure **efficient code and lightweight data**.
* For learning challenges, we’ve divided tasks and are focusing on what’s necessary for the project — not trying to learn everything at once.
* For voice issues, we’ll use the **Web Speech API**, which is supported by most browsers and works well in quiet environments. We'll also provide a button alternative if voice fails.

### ✅ 3. **What Challenges Did Other Existing Projects Face That We Want to Solve?**

* **Answer with article examples included:**

| **Project** | **Challenges Found** |
| --- | --- |
| Smart Switch Using IoT (ResearchGate) | No multi-user support, no device naming |
| Smart Home with ESP8266 (IJSTR, IEEE) | No voice control, no device history/logs |
| Bluetooth Smart Home Projects | Short-range only (no internet), not scalable, no web or voice interface |

* ✅ **How we're solving them:**
* “Our system supports multiple users, lets each user name their devices (e.g., Kitchen Fan), and adds voice control and usage logs — which these systems didn’t include.”

### ✅ 4. **What Features Could You Add Later to Improve the Project? (Future Work)**

✅ **Answer:**

“In future, we could add:

* Energy usage tracking per device
* Automatic schedules (e.g., turn off at midnight)
* Notification alerts if a device is on too long
* Support for sensors (motion, light) for full automation
* A mobile app version for better user experience”

### ✅ 5. **Can You Give an Overview of Your System – How It Works for a User?**

✅ **Answer:**

“After registration and login, the user will see a dashboard. There, they can:

* Add their devices (like Bedroom Light, Kitchen Fan)
* Turn devices ON or OFF using buttons or voice commands
* See the current status (ON/OFF) of each device
* Check history to see when a device was last used

All this is made possible by linking the web interface (Django) to the ESP8266 microcontroller, which physically controls the device using a relay module.”

### ✅ 6. **Can You Explain the Technical Flow of the System?**

✅ **Answer:**

“Yes, here’s how the system works behind the scenes:

1. The user clicks a button or uses a voice command on the website.
2. Django sends a signal to the backend.
3. The backend sends a request to the ESP8266 (via HTTP).
4. ESP8266 receives it and activates the correct GPIO pin.
5. The relay module switches the connected device ON/OFF.
6. The system updates the database with the device’s new status

### ✅ Final Question: **The Articles You Mentioned – Are They Real? What Are They?**

Great question! Here's how to answer this:

✅ **Answer:**

“Yes, those are real papers and online research documents. Some of them are from:

* ResearchGate (community where people publish projects)
* IJSTR (International Journal of Scientific & Tech Research)
* IEEE Access (technical journal for research papers)  
  We reviewed them to understand what was missing in existing projects, and how to improve ours.”

For example:

* Smart Switch Using IoT (ResearchGate) – No user access control
* ESP8266 Smart Home (IJSTR) – Didn’t have voice control or history
* Bluetooth Smart Control – Only worked in short distance (not internet-based)

### ✅ 3. **What happens if the internet disconnects? Will the system still work?**

**Answer:**

“Since we’re using a web-based system with IoT, internet is required for remote control. However, the ESP8266 can still be programmed to respond to local network commands or fallback modes — which we could explore in future updates.”

**✅ 4. What database are you using and why?**

**Answer:**

“We are using **SQLite** because it's lightweight, easy to set up, and fully supported by Django. For a student project like ours, it is fast and sufficient. But later, we can switch to **MySQL** if we need more performance or users.”

**✅ 5. What’s the difference between frontend and backend in your system?**

**Answer:**

* **Frontend**: What the user sees (webpages, buttons, voice input) – made with HTML, CSS, JavaScript.
* **Backend**: The logic behind the system – processes commands, talks to the database, sends commands to devices – made with Django (Python).

### ✅ 6. **Can the user add more than one device? How is it handled in the system?**

**Answer:**

“Yes. Each user can add multiple devices (e.g., Kitchen Light, Bedroom Fan). When added, the device is stored in the database, and the user sees it on their dashboard with an ON/OFF switch. Each device is controlled separately.”

**✅ 7. What security features does your system have?**

**Answer:**

“Our system includes:

* User authentication (login and registration)
* Admin control for managing users and device types
* Devices are only accessed by their respective users
* We avoid exposing device control URLs to public”

### ✅ 8. **What makes your system better than existing ones?**

**Answer:**

“Most existing systems use Bluetooth (short-range), don’t support voice control or multi-user features.  
Our system allows:

* Internet-based remote control
* Voice and button control
* Personalized device naming
* User-specific dashboards and history logs  
  All these make it more interactive and scalable.”

### ✅ 9. **What technologies are you using and what is their role?**

| **Technology** | **Purpose** |
| --- | --- |
| HTML, CSS | Frontend (webpage layout and styling) |
| JavaScript | Interactivity + Voice control (Web Speech API) |
| Django (Python) | Backend logic, API endpoints |
| SQLite | Store users, devices, usage logs |
| ESP8266 | Wi-Fi enabled microcontroller to switch real devices |
| Relay Module | Turns the actual device ON/OFF |
| Arduino IDE | Write and upload code to ESP8266 |

**✅ 10. If you had more time, what would you improve in your system?**

**Answer:**

* Add scheduling (e.g., auto turn-off at night)
* Add sensor integration (motion or temperature)
* Show real-time energy consumption
* Build a mobile app version
* Add notifications for forgotten devices

### ✅ **1. What does “Web UI Mockups and Design Diagrams” mean?**

* **Web UI Mockups** = Visual sketches of what your webpages will look like (like login page, dashboard, device page).
  + Example: A drawing of how your dashboard will look, with buttons for turning devices on/off.
  + Tools used: Figma, Draw.io, Pen & Paper
* **Design Diagrams** = Visual representations of how parts of your system will interact.
  + Can include things like:
    - **Use Case Diagram**
    - **Activity Diagram**
    - **System Architecture Diagram**

✅ **When to use it?**  
Use this when you are talking about **how the system will look and behave from the user’s view** — **UI/UX design**.

**✅ 2. What is “Object-Oriented Design Model”?**

* This is about the **technical structure** of your system using objects/classes.
* In Django, it means how you model:
  + User class → with name, email, password
  + Device class → with name, status, type
  + UsageLog class → with time ON/OFF, etc.

✅ **When to use it?**  
Use this when you are talking about the **backend structure**, especially in Django or Python — **database design and logic**.

**✅ Which one is better to write in Design section?**

✔ **Both are useful**, but they serve different parts:

| **If you're writing about…** | **Use This** |
| --- | --- |
| How users will interact with the system (pages) | ✅ Web UI Mockups and Design Diagrams |
| How the backend is structured (classes/models) | ✅ Object-Oriented Design Model |

**✅ Suggested Final Answer:**

“For the frontend and system behavior, we used **Web UI mockups and system diagrams** to design the look and navigation of the system.  
For the backend and logic, we used **Object-Oriented Design** to model users, devices, and device logs.”