**System Architecture Details**

**Hardware Components Used**

* Raspberry pi 4 with Camera Module: captures Images and handles Facial Recognition
* PIR Motion Sensor: Detects motion at the door.
* Doorbell Button: Allows the user to initiate interactions, it’s like a doorbell button which gives the user the feel for the regular bells, acts as a trigger for the camera sensor.
* Speaker: provides Audio Feedback that the user has intended to output through the App
* LED: Turns on when the camera turns on to help the user know when the camera is going to take an image.
* Power Supply: The power is supplied from the raspberry pi.

**Web Server:**

* Flask server: Hosting the web application and the API endpoints.

1. Frontend Hosting: Flask serves as the web server to host the frontend of your application. The frontend is developed using Kotlin (Android app) and provides the user interface for interacting with the doorbell system. Users can receive real-time notifications, view images, and control the doorbell.
2. Flask exposes RESTful API endpoints that enable communication between the frontend (Kotlin app) and the backend (AWS instances). These API endpoints facilitate actions like sending image capture requests to the Raspberry Pi, retrieving captured images from the database, user authentication, and more.
3. Flask also handles user Authentication, ensuring only users can access certain functionality and moreover it helps with the Request Processing in a way that the frontend sends a HTTPS request to the flask , Flask then processes the requests and then communicates it to the Backed , i.e. the **AWS** instances and then initiate the actions like retrieving an image from the database , authentication etc .

* Kotlin: Using Jetpack Compose build an App for the frontend client user.

**IOT Elements**

Apart from all the sensors and hardware we also have the following things we are going to be using.

* PubNub : This helps real time communication between Pi and the frontend App , the app will be subscribed to the pubnub which is going to output all the data
* Raspberry Pi: It is going to be the software for the bell that we are building.
* AWS IOT Core : During research came the fact that this provides a secure management and MQTT communication . This is optional as its not been discussed yet with the project Manger / John

**AWS Instances**

In our Project we are using AWS EC2 to host the backend services

* AWS EC2 instances host the Flask web server, which serves as the backend of your application. These instances are responsible for processing requests from the frontend and interacting with various components of the system, including the Raspberry Pi and the SQL database.
* As described earlier this is used for Request Handling
* **Security**: AWS instances can be configured with security groups and access control policies to restrict access and enhance security. For example, you can allow access to API endpoints only from trusted sources, enhancing the security of your IoT system.
* **Database Integration**: In some cases, AWS EC2 instances may also interact with AWS RDS (Relational Database Service) to access the SQL database. This provides a managed and scalable database solution, further enhancing data reliability and security.

**Database**

* SQL Database: Stores user data, camera sensor data, and authentication information. May be hosted on AWS RDS for scalability and reliability.
* Reference from <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html>
* Encrypt data at rest using database-level encryption features.

**References**

[**https://www.simplilearn.com/tutorials/aws-tutorial/aws-ec2**](https://www.simplilearn.com/tutorials/aws-tutorial/aws-ec2)

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html>