1. **Introduction**

**Overview of the Project**

The proposed database design enhances the management and operation of a smart home ecosystem, enabling seamless interaction between users and connected devices. This system is organized to store and manage essential data related to users, devices, rooms, and sensor readings while keeping logs of device actions and automation rules.

Key features of the design include:

- User authentication

- Device categorization by rooms or zones

- Real-time storage of sensor data

- Control logs for auditing purposes

- Management of automation rules

Additionally, the design includes an alert system to notify users of significant events, such as intrusions or low battery levels. This relational database schema provides a scalable and efficient foundation for supporting smart home operations, offering improved usability, data integrity, and future extensibility for advanced functionalities.

**1.1 Purpose and Scope of the database**

The main goal of the smart home database is to efficiently manage the data from various smart home devices and their interactions. It serves key functions such as device management, data collection, automation, user access management, alerts and notifications, and maintaining operational logs for accountability and troubleshooting.

**The Scope of the database Involves:**

* Information Management:

- This entails the preservation of vital data regarding smart gadgets, occupants, and the distinct areas or zones within the residence.

- The system captures immediate sensor information, such as temperature, movement, and humidity metrics.

- It systematically archives device activities and automation occurrences for future analysis.

* Automation and Oversight:

- The repository facilitates the formulation and implementation of automation guidelines and timetables, fostering seamless operation of the intelligent home.

- Users can command devices instantaneously through straightforward directives, increasing ease of use.

* User Roles and Privileges:

- The architecture distinguishes between administrative and visitor roles, offering tailored access according to user requirements.

- Safeguarding data security is emphasized through robust authentication protocols.

* Alerts and Announcements:

- Users are notified of critical situations, such as potential security breaches, battery depletion, or unusual temperature shifts.

- The framework provides actionable recommendations via alerts to keep users well-informed.

* Growth and Flexibility:

- The repository is structured to embrace additional devices, areas, and automation capabilities, facilitating expansion as the smart home evolves.

- Integration with external platforms, including cloud services and third-party applications, is encouraged to amplify functionality.

* Support for Insights:

- It allows for the examination of historical sensor data, enabling users to optimize energy usage and discern trends in device activity.

- The system endorses proactive maintenance for smart devices, ensuring they function efficiently over time.

**1.2 Problem statement**

The rapid adoption of smart home technologies is poised to increase the number of interconnected devices and the data they produce, but challenges lie ahead:

1. Data Fragmentation: Managing diverse device data without a centralized system will become increasingly complex.

2. Scalability Issues: As homes expand, existing solutions may face performance bottlenecks.

3. Real-Time Processing: Delays in critical operations, like security alerts, may arise without improved infrastructure.

4. Security Risks: Vulnerabilities related to sensitive user information will require stronger data management.

5. Inefficient Automation: Synchronizing automation across devices may prove challenging without a unified platform.

6. Analytics Limitations: There will be a greater need for effective logging and historical data querying.

7. Alert Management: Enhancing alert systems to customize notifications based on user preferences will be essential.

**1.3 Objectives**

* Establish a comprehensive system to register, organize, and manage all smart home devices, such as lights, thermostats, cameras, and sensors.
* Enable the storage and retrieval of real-time data from sensors and devices to support immediate actions, such as triggering automation rules or sending alerts.
* Design the database to accommodate the increasing number of devices, users, and automation rules as the smart home ecosystem grows.
* Enable users to create, store, and execute custom automation rules and schedules for smooth device interaction and synchronization.
* Establish a strong access control system to distinguish between user roles (such as admin and guest) and offer tailored interactions with devices.
* Maintain historical logs of device actions, user interactions, and sensor data to support analytics, troubleshooting, and performance optimization.
* Safeguard sensitive information about users and devices by using encryption, implementing access controls, and adhering to data privacy regulations.
* Support different types of devices and communication methods to ensure they work together in a smart home with products from various brands.
* Enable the analysis of energy usage patterns to optimize power consumption and reduce costs.

**2.0 Requirement Analysis**

**2.1 Data Requirement**

The main entities for this smart home design is as follows:

* User (Stores information about the user of the smart home)
  + UserID (Primary Key)
  + UserName
  + Email
  + Role (admin, guest)
  + Password
  + Created at
* Room (Defining an area within the smart home)
  + RoomID (Primary Key)
  + RoomName
  + Floor
* Devices (Stores information about device connected to the system)
  + DeviceID (Primary Key)
  + DeviceName
  + DeviceType (light, thermostat, camera, sensor, other)
  + Status (on, off, idle)
  + RoomID (Foreign key references RoomID)
  + Created at (date of creation)
* Control Unit ( control the entire smart system)
  + ControlUnitID (Primary key)
  + Action type ( Execute rule, send commands, generate alert)
  + DeviceID (Foreign Key references DeviceID)
  + SensorDataID(Foreign key references SensorDataID)
  + RulesID ( Foreign key references RulesID)
  + TriggeredBy (user, sensor, automation rule)
  + Timestamp ( When the action occurred)
* Voice Assistant
  + Voice Assistant ID (Primary key)
  + Voice Assistant Name
  + DeviceID( Foreign Key references DeviceID)
  + Created at
* Sensor Data (Collect real time sensor data)
  + SensorDataID (Primary key)
  + DeviceID (Foreign key references DeviceID)
  + Sensor Type
  + Value
  + Timestamp
* ControlLog (Keeps up-to-date record performed on the device)
  + LogID (Primary key)
  + DeviceID ( Foreign key references DeviceID)
  + UserID (Foreign key references UserID)
  + Action
  + Timestamp
* AutomationRules (Stores user-defines automation rules)
  + RuleID (Primary key)
  + RuleName
  + Condition
  + Action
  + DeviceID (Foreign key references DeviceID)
  + IsActive (true or false)
* Alert ( logs alert and notification for users)
  + AlertID (Primary Key)
  + DeviceID ( Foreign key references DeviceID)
  + Alert type ( overheating, intrusion, low battery, others)
  + Resolved status ( true or false)

**3.0 Conceptual Design**

#### Entity-Relationship Diagram (ERD)

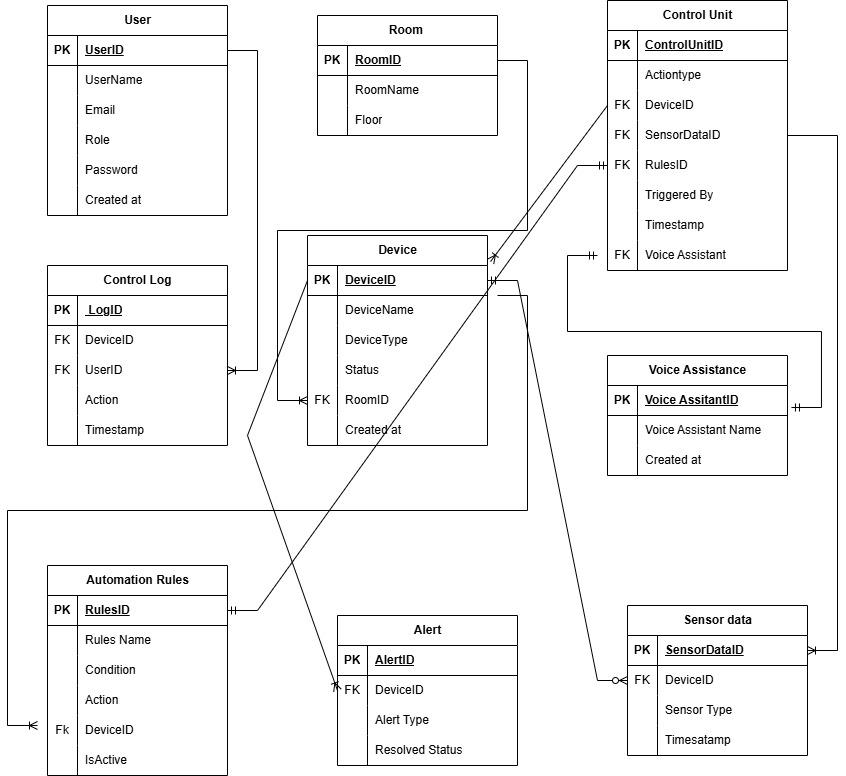
In the ERD, we will represent key entities, their attributes, and the relationships between them. Below is a textual representation of the entities and relationships in the ride-hailing app. An actual ERD diagram will be generated using diagramming Draw.io.

* Entities
  + User (General for both admin and guest)
  + Room(Link to devices)
  + Devices (it can be controlled centrally)
  + Control Unit (Has access to all devices)
  + Remote Control (controls one device per time)
  + Voice Assistant (Accesses the functionality of all devices)
  + Sensor data ( has access to most informations)
  + Control logs ( responsible for role-based access)
  + Automated Rules (Associated with the sensor data)
  + Alert ( makes signal sending possible for the control unit)

Relationships:

* User “performs” action on devices.
  + A User (admin or guest) can use more than one devices.
  + Cardinality: 1 User can make use of 0 or more devices per time.(1:N)
* Control Log “Shows” User actions.
  + A User can have access to more than one control log statements.
  + Cardinality: 1 User can perform 0 or more actions per time. (1: N)
* Room “contains” devices:
  + Each room can accomodate more than one devices.
  + One device belongs to one Room.
  + Cardinality: 1 Room to 0 or more devices (1:N) or one-to-many relationship, one device for a room is a (1:1) or one-to-one relationship.
* Device “generate” sensor data:
  + Device like sensors generate periodic data.
  + Cardinality: 1 device (Sensor) can produce 0 or more data, making a one-to-many relationship (1:N) and each periodic data is connected to one sensor device.
* Device “has” automated rules:
  + Each device can have one automated rule attached to it.
  + Cardinality: 1 device to 0 or one automated rule (1:1)
* Device “trigger” alert:
  + One device can trigger alerts based on certain condition like status or sensor data.
  + Cardinality: 1 device to 0 or more alerts (1:N).
* Control Unit “ assigns” User role:
  + One or more roles are being assigned by the control unit.
  + Cardinality: 1 control unit to 0ne or more User role (1:N).
* Voice assistant “control” device functionality
  + Voice assistant can give access to more than of features of different devices.
  + 1 voice assistant to one or many devices (1:N)

**SMART HOME ERD**



Entity descriptions and Attributes

* User:
  + UserID: A unique identifier for each user (Primary Key).
  + UserName: full name of user.
  + Email: email of the user.
  + Role:The different roles of the user indicating whether (“Admin” or “Guest”)
  + Password: A unique mixture of characters.
  + Created at: The time and date the user was registered.
* Room:
  + RoomID: A unique identifier for each room (Primary Key).
  + RoomName: The name of the room.
  + Floor: The floor number where the room is located.
* Devices:
  + DeviceID: A unique identifier (Primary Key).
  + DeviceName: The name of device.
  + DeviceType: Types of device available like (light, thermostat, camera, sensor, other).
  + Status: Device current state (on, off, idle).
  + RoomID: Links the device to a specific Room (Foreign key).
  + Created at: Date and Time the device was installed.
* Control Unit:
  + ControlUnitID: Unique identifier for the Control Unit (Primary key)
  + Action type: Action taken by the Control Unit per time ( Execute rule, send commands, generate alert).
  + DeviceID: Links the control unit to the device (Foreign Key)
  + SensorDataID: Links the control unit to the Sensor data(Foreign key).
  + RulesID: Links the control unit to the automated rules ( Foreign key).
  + TriggeredBy: The device being per time like (user, sensor, automation rule).
  + Timestamp: The time and date the action occurred.
* Voice Assistant:
  + VoiceAssitantID: A unique identifier for Voice Assistant (Primary key).
  + VoiceAssitantName: Name of voice assistant e.g Alexa.
  + DeviceID: Links Voice assistant to the device ( Foreign Key).
  + Created at: Date of installation.
* Sensor Data:
  + SensorDataID: A unique identifier for Sensor data (Primary key).
  + DeviceID: Links Sensor data to the device (Foreign key).
  + Sensor Type: Type of Sensor such as heat sensor, security breach etc
  + Value: Value of the Sensor data.
  + Timestamp: Date and Time the data was collected.
* ControlLog:
  + LogID: A unique identifier for ControlLog (Primary key).
  + DeviceID: Links ControlLog to a Device ( Foreign key).
  + UserID: Links ControlLog to the User (Foreign).
  + Action: Current action that took place.
  + Timestamp: Date and Time of log.
* AutomationRules:
  + RuleID: Unique identifier for Automation Rules (Primary key)
  + RuleName: Name of rule.
  + Condition: The condition before the rule triggers.
  + Action: Action to be want the rule is triggered.
  + DeviceID: Links Automated Rule to the Device (Foreign key).
  + IsActive: States whether the rule has been triggered or not (true or false)
* Alert:
  + AlertID: Unique identifier for Alert (Primary Key).
  + DeviceID: Links the Alert to a device ( Foreign key).
  + Alert type: The actual type of alert ( overheating, intrusion, low battery, others).
  + Resolved status: States if the reason for the alert has been resolved ( true or false).

**4. Logical Design**

The logical design phase involves converting the conceptual design into a detailed relational schema. This schema defines the tables, their fields, data types, and relationships between them. We’ll also normalize the tables to ensure the database is efficient, avoids redundancy, and maintains data integrity.

#### Relational Schema

The relational schema outlines how data will be stored in tables. Below are the main tables derived from the entities in the conceptual design:

* User Table
  + UserID: INT (Primary Key)
  + UserName: VARCHAR(100)
  + Email: VARCHAR(100)
  + Role: ENUM (‘Admin’, ‘ Guest’)
  + Password: VARCHAR(100)
  + Created at: TIMESTAMP
* Room Table
  + RoomID: (Primary Key)
  + RoomName: VARCHAR(100)
  + Floor: INT
* Device Table
  + DeviceID: INT (Primary Key)
  + DeviceName: VARCHAR(100)
  + DeviceType: ENUM (“Light”, “thermostat”, “camera”, “sensor”, “other”)
  + Status: ENUM (‘On’, ‘Off’, ‘ Idle’)
  + RoomID: INT (Foreign key)
  + Created at: TIMESTAMP
* Control Unit Table
  + ControlUnitID: INT (Primary key)
  + Action type:ENUM ( “Execute rule”, “Send Commands”, “ Generate Alert”)
  + DeviceID: INT (Foreign Key)
  + SensorDataID: INT(Foreign key)
  + RulesID: INT ( Foreign key)
  + TriggeredBy: ENUM (“user”, “ sensor”, “automation rule”)
  + Timestamp: TIMESTAMP
* Voice Assistant Table
  + VoiceAssitantID: INT (Primary key)
  + VoiceAssitantName: VARCHAR(100)
  + DeviceID: INT( Foreign Key)
  + Created at: TIMESTAMP
* Sensor Data Table
  + SensorDataID: INT (Primary key)
  + DeviceID: INT (Foreign key)
  + Sensor Type:VARCHAR(50)
  + Value: FLOAT
  + Timestamp: TIMESTAMP
* ControlLog Table
  + LogID: INT (Primary key)
  + DeviceID: INT ( Foreign key)
  + UserID: INT (Foreign key)
  + Action : VARCHAR(100)
  + Timestamp: TIMESTAMP
* AutomationRules Table
  + RuleID: INT (Primary key)
  + RuleName: VARCHAR(50)
  + Condition: VARCHAR(100)
  + Action: VARCHAR(100)
  + DeviceID: INT (Foreign key)
  + IsActive: BOOLEAN
* Alert Table
  + AlertID: INT (Primary Key)
  + DeviceID: INT ( Foreign key)
  + Alert type: ENUM ( ‘Overheating’, ‘Intrusion’, ‘Low battery’, ‘Others’)
  + Resolved status: BOOLEAN

#### Tables, Fields, and Data Types

Let's break down the tables further, including their fields and data types.

* User Table:
  + Fields: UserID, UserName, Email, Role, password, Created at
  + Data types:
    - UserID: INT (Primary Key)
    - UserName: VARCHAR(100)
    - Email: VARCHAR(100)
    - Role: ENUM (‘Admin’, ‘ Guest’)
    - Password: VARCHAR(100)
    - Created at: TIMESTAMP

| **User** | **Data Type** | **Constraints** |
| --- | --- | --- |
| UserID | SERIAL | PRIMARY KEY, AUTO\_INCREMENT |
| UserName | VARCHAR(100) | NOT NULL |
| Email | VARCHAR(100) | UNIQUE, NOT NULL |
| Role | ENUM(‘Admin’, ‘Guest’) | DEFAULT ‘Guest’ |
| Password | VARCHAR(100) | NOT NULL |
| Created at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* Room Table:
  + Fields: RoomID, RoomName, FloorName
  + Data Type:
    - RoomID: (Primary Key)
    - RoomName: VARCHAR(100)
    - Floor: INT

| **Room** | **Data Type** | **Constraints** |
| --- | --- | --- |
| RoomID | SERIAL | PRIMARY KEY, AUTO\_INCREMENT |
| RoomName | VARCHAR(100) | NOT NULL |
| Floor | INT | NOT NULL |

* Device Table:
  + Fields: DeviceID, DeviceName, DeviceType, Status, RoomID, ControlUnitID, RemoteControlID, VoiceAssitstantID, Created at
  + Data Type:
    - DeviceID: INT (Primary Key)
    - DeviceName: VARCHAR(100)
    - DeviceType: ENUM (“Light”, “thermostat”, “camera”, “sensor”, “other”)
    - Status: ENUM (‘On’, ‘Off’, ‘ Idle’)
    - RoomID: INT (Foreign key)
    - Created at: TIMESTAMP

| **Device** | **Data Type** | **Constraints** |
| --- | --- | --- |
| DeviceID | INT | PRIMARY KEY |
| DeviceName | VARCHAR(100) | NOT NULL |
| DeviceType | ENUM('Light', 'Thermostat', 'Camera', 'Sensor', 'Other') | NOT NULL |
| Status | ENUM('ON', 'OFF', 'IDLE') | DEFAULT ‘OFF’ |
| RoomID | INT | FOREIGN KEY |
| Created at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* Control Unit Table:
  + Fields: ControlUnitID, Action type, DeviceID, SensorDataID, RulesID, TriggeredBy, Timestamp
  + Data Type:
    - ControlUnitID: INT (Primary key)
    - Action type:ENUM ( “Execute rule”, “Send Commands”, “ Generate Alert”)
    - DeviceID: INT (Foreign Key)
    - SensorDataID: INT(Foreign key)
    - RulesID: INT ( Foreign key)
    - TriggeredBy: ENUM (“user”, “ sensor”, “automation rule”)
    - Timestamp: TIMESTAMP

| **Control Unit** | **Data Type** | **Constraints** |
| --- | --- | --- |
| Control UnitID | INT | PRIMARY KEY |
| Action Type | ENUM(‘Execute rule’, ‘Send Command’, ‘Generate Alert’) | NOT NULL |
| DeviceID | INT | FOREIGN KEY |
| SensorDataID | INT | FOREIGN KEY |
| RulesID | INT | FOREIGN KEY |
| TriggeredBy | ENUM(‘User’, ‘sensor’, ‘automation rule’ | NOT NULL |
| Timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* Voice Assistant Table:
  + Fields: VoiceAssistantID, VoiceAssistantName, DeviceID
  + Data Type:
    - VoiceAssitantID: INT (Primary key)
    - VoiceAssitantName: VARCHAR(100)
    - DeviceID: INT( Foreign Key)
    - Created at: TIMESTAMP

| **Voice Assistant** | **Data Type** | **Constraints** |
| --- | --- | --- |
| Voice AssistantID | INT | PRIMARY KEY |
| Voice AssitantName | VARCHAR(100) | NOT NULL |
| DeviceID | INT | FOREIGN KEY |
| Created at | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* Sensor Data Table:
  + Fields: SensorDataID, DeviceID, Sensor Type, Value, Timestamp
  + Data Type:
    - SensorDataID: INT (Primary key)
    - DeviceID: INT (Foreign key)
    - Sensor Type:VARCHAR(50)
    - Value: FLOAT
    - Timestamp: TIMESTAMP

| **Sensor Data** | **Data Type** | **Constraints** |
| --- | --- | --- |
| SensorDataID | INT | PRIMARY KEY |
| DeviceID | INT | FOREIGN KEY |
| Sensor Type | VARCHAR(50) | NOT NULL |
| Value | FLOAT | NOT NULL |
| Timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* ControlLog Table:
  + Fields: LogID, DeviceID, UserID, Action, Timestamp
  + Data Type:
    - LogID: INT (Primary key)
    - DeviceID: INT ( Foreign key)
    - UserID: INT (Foreign key)
    - Action : VARCHAR(100)
    - Timestamp: TIMESTAMP

| **ControlLog** | **Data Type** | **Constraints** |
| --- | --- | --- |
| LogID | INT | PRIMARY KEY |
| DeviceID | INT | FOREIGN KEY |
| UserID | INT | FOREIGN KEY |
| Action | VARCHAR(100) | NOT NULL |
| Timestamp | TIMESTAMP | DEFAULT CURRENT\_TIMESTAMP |

* Automation Rules table:
  + Fields: RuleID, RuleName, Condition, Action, DeviceID, IsActive
  + Data Type:
    - RuleID: INT (Primary key)
    - RuleName: VARCHAR(50)
    - Condition: VARCHAR(100)
    - Action: VARCHAR(100)
    - DeviceID: INT (Foreign key)
    - IsActive: BOOLEAN

| **Automation Rules** | **Data Type** | **Contraints** |
| --- | --- | --- |
| RuleID | INT | PRIMARY KEY |
| RuleName | VARCHAR(50) | NOT NULL |
| Condition | VARCHAR(100) | NOT NULL |
| Action | VARCHAR(100) | NOT NULL |
| DeviceID | INT | FOREIGN KEY |
| IsActive | BOOLEAN | DEFAULT TRUE |

* Alert Table:
  + Fields: AlertID, DeviceID, Alert Type, Resolved Status
  + Data Type:
    - AlertID: INT (Primary Key)
    - DeviceID: INT ( Foreign key)
    - Alert type: ENUM ( ‘Overheating’, ‘Intrusion’, ‘Low battery’, ‘Others’)
    - Resolved status: BOOLEAN

| **Alert** | **Data Type** | **Constraints** |
| --- | --- | --- |
| AlertID | INT | PRIMARY KEY |
| DeviceID | INT | FOREIGN KEY |
| Alert Type | ENUM('Intrusion', 'LowBattery', 'Overheat', 'Other') | NOT NULL |
| Resolved status | BOOLEAN | DEFAULT FALSE |