SQL Schema Overview - Subject to Change - 29-03-25 - Draft 1.0 - Darsh K

Before delving into the schema, we should first consider our setup on the pi itself via the following manner:

- Install MariaDB (or MySQL, PostgreSQL, SQLite).
- Create a database + tables.
- Insert roles and permissions.

We may integrate accordingly, most-likely using ***our API.***

- The Pi application (Python, Node.js, etc.) handles the logic for reading from and writing to this DB.
- The Android app communicates with the Pi (via REST, MQTT, or WebSocket) to manage face data, open/close door, retrieve logs, etc.
- Also need to fill in the actual permission sets per role in `roles_permissions`.

I have *attempted* to create an SQL schema which hopes to address most of these requirements, and is maximally coherent with our most-recent sketch of the app's UI and flows (*as shared by Lavanya*).

1. User and role management - this would also include per-doorbell entries.
2. Doorbell registration (name/password/settings).
3. Face profiles (with blacklisting at the doorbell level if required).
4. App-side user settings (which would include 2FA toggles, face-login for the mobile app, and so on).
5. Device tracking (for "remember" me sessions, device info etc.).
6. Logs & Recordings (event logs, video references for the footage (ids and the like)).

Please also see below, a high-level entity/relationship outline (on the basis of tables) of the proposed schema:

A. We have users and roles here:

- `users` would contain global user accounts with username, email, password, etc.
- `user_settings` may include per-user toggles for 2FA, face login for the app, notifications, etc.
- `devices (we may consider this as an optional implementation)` would entail per-user device info (e.g., phone-models, MAC/IP, last-login and more if needed.).
- `roles` would include names roles such as admin, owner, adult, child, guest, etc.
- `permissions` would as agreed upon, include the fine-grained permissions per user, which would perhaps be (manage*user_profiles, open_door, etc.).*
  _Just sharing some potential options as suggested by Baron: ***(please note that this is verbatim as highlighted in the shared google-docs)***

1. Potential permissions for users:

- Doorbell:

1. Open/close door (toggle for auto close also but ehh)
2. Check live feed
3. Store recordings

4. See doorbell logs (unsure about this one)

- Management

5. Manage user profiles
6. Manage face profiles
7. Manage blacklist
8. Manage roles

- Alerts

1. Person at door
2. Blocked person at door
3. Break in detected
4. New user added to doorbell
5. New face added to doorbell

- Presets: Admin: Everything, Adult: Everything, Children: Doorbell - (2,3,4), Alerts - (1,2,3), Guest: None
- `roles_permissions` would require a mapping table linking each role to a set of permissions as specified above.
- `user_permissions_override (this might be optional)` in order to allow one to tweak a single user's permission(s) beyond their role-default.

B. Doorbells

- `doorbells` is likely to include each doorbell's name, password, owner, and doorbell-wide settings (e.g., face recognition on/off, recordings after 72h, and so on as required.)
- `user_doorbell` is in place to determine which users have access to which doorbell, to ensure the apt users have access to the apt doorbells, and at what role-level (so _we must consider that the same user can be an "owner" on doorbell #1, only to be a "guest" on doorbell #2 etc.).

C. Facial recognition

- `face_profiles` would probably include the actual face-embeddings and/or image-references.
- `doorbell_face_profiles (possibly optional)` in order to consider cases where doorbell blacklisting is *doorbell-specific*. If we proceed with the former consideration in mind, we may store a link here with the `is_blocked` flag and also reference the face. Otherwise, if the very procession of "blocking" is global, we can store it in `face_profiles`.

D. Logging & Recordings

- `doorbell_logs` likely consists of detailed doorbell events as suggested in the app-sketches (new-user-added, face-recognized, door-opened, break-in, plentiful possible combinations or options here.).
- `recordings` would likely feature the video-references on the pi (filename, date/time)--which may assist syncing, and automatically be purged after 72-hours in relation to each particular file's date-and-time of addition.

E (which might be optional) - Sensor Data

- `sensor_data` if we decide to store temperature/motion/light logs, can reference `doorbell_id` and keep historical data for future-reference.

Schema in greater detail:

The sections below show a visual-representation of the table's columns with some notes on purpose and below each, the SQL code that *might* be used to create the tables as specified above.

1. User & Authentication

1.1 `users`

Column	Type	Constraints	Notes
user_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique global user identifier
username	VARCHAR(50)	NOT NULL, UNIQUE	Login username
email	VARCHAR(100)	NOT NULL, UNIQUE	User email
password_hash	VARCHAR(255)	NOT NULL	Hashed password
display_name	VARCHAR(100)	(nullable)	User-friendly name
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Account creation timestamp

```
CREATE TABLE users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(50) NOT NULL UNIQUE,
  email VARCHAR(100) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  display_name VARCHAR(100),
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

1.2 `user_settings`

Column	Type	Constraints	Notes
user_id	INT	PRIMARY KEY, FOREIGN KEY (users.user_id)	Matches a user from the `users` table
enable_2fa	BOOLEAN	NOT NULL, DEFAULT FALSE	Two-factor authentication toggle
face_login_app	BOOLEAN	NOT NULL, DEFAULT FALSE	Enable face login on the mobile app
enable_in_app_alerts	BOOLEAN	NOT NULL, DEFAULT TRUE	In-app alert toggle
enable_push_notifications	BOOLEAN	NOT NULL, DEFAULT TRUE	Push notification toggle
notify_new_device_login	BOOLEAN	NOT NULL, DEFAULT TRUE	Notify on new device login
notify_new_user_to_doorbell	BOOLEAN	NOT NULL, DEFAULT TRUE	Alert when a new user is added to a doorbell
notify_familiar_user_entered	BOOLEAN	NOT NULL, DEFAULT TRUE	Alert for familiar user presence
notify_unfamiliar_guest	BOOLEAN	NOT NULL, DEFAULT TRUE	Alert for an unfamiliar guest

Column	Type	Constraints	Notes
notify_blacklisted_user	BOOLEAN	NOT NULL, DEFAULT TRUE	Alert for a blacklisted face

```sql
CREATE TABLE user_settings (
  user_id INT PRIMARY KEY,
  enable_2fa BOOLEAN NOT NULL DEFAULT FALSE,
  face_login_app BOOLEAN NOT NULL DEFAULT FALSE,
  enable_in_app_alerts BOOLEAN NOT NULL DEFAULT TRUE,
  enable_push_notifications BOOLEAN NOT NULL DEFAULT TRUE,
  notify_new_device_login BOOLEAN NOT NULL DEFAULT TRUE,
  notify_new_user_to_doorbell BOOLEAN NOT NULL DEFAULT TRUE,
  notify_familiar_user_entered BOOLEAN NOT NULL DEFAULT TRUE,
  notify_unfamiliar_guest BOOLEAN NOT NULL DEFAULT TRUE,
  notify_blacklisted_user BOOLEAN NOT NULL DEFAULT TRUE,

  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

1.3 `devices`

Column	Type	Constraints	Notes
device_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique device identifier
user_id	INT	NOT NULL, FOREIGN KEY (users.user_id)	Owner of the device
device_name	VARCHAR(100)	(nullable)	Name given to the device
device_model	VARCHAR(100)	(nullable)	Model information
mac_address	VARCHAR(50)	(nullable)	Device MAC address
ip_address	VARCHAR(50)	(nullable)	Last known IP address
last_login	DATETIME	(nullable)	Timestamp of last login
remember_me_expires	DATETIME	(nullable)	Expiry for “remember me” sessions

```sql
CREATE TABLE devices (
  device_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  device_name VARCHAR(100),
  device_model VARCHAR(100),
  mac_address VARCHAR(50),
  ip_address VARCHAR(50),
  last_login DATETIME,
  remember_me_expires DATETIME,

  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

2. Roles & Permissions

2.1 `roles`

Column	Type	Constraints	Notes
role_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique role identifier
role_name	VARCHAR(50)	NOT NULL	e.g., 'Admin', 'Owner', 'Adult', etc.

```
CREATE TABLE roles (
  role_id INT AUTO_INCREMENT PRIMARY KEY,
  role_name VARCHAR(50) NOT NULL
);
```

2.2 `permissions`

Column	Type	Constraints	Notes
permission_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique permission identifier
permission_name	VARCHAR(100)	NOT NULL	e.g., 'manage_user_profiles', 'open_door', etc.

```
CREATE TABLE permissions (
  permission_id INT AUTO_INCREMENT PRIMARY KEY,
  permission_name VARCHAR(100) NOT NULL
);
```

2.3 `roles_permissions`

(Mapping table for many-to-many relationship between roles and permissions)

Column	Type	Constraints	Notes
role_id	INT	NOT NULL, part of PRIMARY KEY, FK to roles	Role identifier
permission_id	INT	NOT NULL, part of PRIMARY KEY, FK to permissions	Permission identifier

```
CREATE TABLE roles_permissions (
  role_id INT NOT NULL,
  permission_id INT NOT NULL,
  PRIMARY KEY (role_id, permission_id),
  FOREIGN KEY (role_id) REFERENCES roles(role_id)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (permission_id) REFERENCES permissions(permission_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

2.4 `user_permissions_override`

(Override table for per-user permission adjustments)

Column	Type	Constraints	Notes
user_id	INT	NOT NULL, part of PRIMARY KEY, FK to users	User identifier
permission_id	INT	NOT NULL, part of PRIMARY KEY, FK to permissions	Permission identifier
is_granted	BOOLEAN	NOT NULL	True if permission is explicitly granted for the user

```
CREATE TABLE user_permissions_override (
  user_id INT NOT NULL,
  permission_id INT NOT NULL,
  is_granted BOOLEAN NOT NULL,

  PRIMARY KEY (user_id, permission_id),
  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (permission_id) REFERENCES permissions(permission_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

---

3. Doorbells & Per–Doorbell Roles

3.1 `doorbells`

Column	Type	Constraints	Notes
doorbell_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique doorbell identifier
doorbell_name	VARCHAR(100)	NOT NULL	Name of the doorbell
doorbell_password	VARCHAR(255)	NOT NULL	Doorbell authentication password
owner_user_id	INT	NOT NULL, FK to users	The owner of the doorbell
enable_facial_recognition	BOOLEAN	NOT NULL, DEFAULT TRUE	Toggle for facial recognition
auto_delete_recordings	BOOLEAN	NOT NULL, DEFAULT TRUE	Automatically purge recordings after 72h

Column	Type	Constraints	Notes
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Doorbell registration timestamp

```sql
CREATE TABLE doorbells (
  doorbell_id INT AUTO_INCREMENT PRIMARY KEY,
  doorbell_name VARCHAR(100) NOT NULL,
  doorbell_password VARCHAR(255) NOT NULL,
  owner_user_id INT NOT NULL,

  enable_facial_recognition BOOLEAN NOT NULL DEFAULT TRUE,
  auto_delete_recordings BOOLEAN NOT NULL DEFAULT TRUE,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (owner_user_id) REFERENCES users(user_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

3.2 `user_doorbell`

(Associates users with doorbells and their roles on each)

Column	Type	Constraints	Notes
user_id	INT	NOT NULL, part of PRIMARY KEY, FK to users	User identifier
doorbell_id	INT	NOT NULL, part of PRIMARY KEY, FK to doorbells	Doorbell identifier
role_id	INT	NOT NULL, FK to roles	User’s role on the doorbell (e.g., Owner, Guest)

```sql
CREATE TABLE user_doorbell (
  user_id INT NOT NULL,
  doorbell_id INT NOT NULL,
  role_id INT NOT NULL,

  PRIMARY KEY (user_id, doorbell_id),
  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (doorbell_id) REFERENCES doorbells(doorbell_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (role_id) REFERENCES roles(role_id)
    ON DELETE RESTRICT
    ON UPDATE CASCADE
);
```

Note: This structure allows a user to have different roles on different doorbells (e.g., “Owner” on one and “Guest” on another).

4. Face Profiles & Blacklist Handling

4.1 `face_profiles`

Column	Type	Constraints	Notes
face_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique face profile identifier
user_id	INT	(nullable), FK to users	Optional: links a face to a known user
face_data	TEXT	NOT NULL	Raw face embedding data or image reference
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Timestamp when face data was created

```
CREATE TABLE face_profiles (
  face_id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT,
  face_data TEXT NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);
```

4.2 `doorbell_face_profiles`

(Links face profiles to specific doorbells, with a block flag)

Column	Type	Constraints	Notes
doorbell_id	INT	NOT NULL, part of PRIMARY KEY, FK to doorbells	Doorbell identifier
face_id	INT	NOT NULL, part of PRIMARY KEY, FK to face_profiles	Face profile identifier
is_blocked	BOOLEAN	NOT NULL, DEFAULT FALSE	Indicates if the face is blocked for that doorbell

```
CREATE TABLE doorbell_face_profiles (
  doorbell_id INT NOT NULL,
  face_id INT NOT NULL,
  is_blocked BOOLEAN NOT NULL DEFAULT FALSE,

  PRIMARY KEY (doorbell_id, face_id),
  FOREIGN KEY (doorbell_id) REFERENCES doorbells(doorbell_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (face_id) REFERENCES face_profiles(face_id)
    ON DELETE CASCADE
```

```
    ON UPDATE CASCADE
);
```

Note: This allows a face to be allowed on one doorbell but blocked on another if needed.

---

5. Doorbell Logs & Recordings

5.1 `doorbell_logs`

Column	Type	Constraints	Notes
log_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique log identifier
doorbell_id	INT	NOT NULL, FK to doorbells	Doorbell where the event occurred
user_id	INT	(nullable), FK to users	User who triggered the event (if applicable)
event_type	VARCHAR(100)	NOT NULL	Type of event (e.g., FACE_RECOGNIZED, DOOR_OPEN)
event_detail	VARCHAR(255)	(nullable)	Additional event details
event_timestamp	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Timestamp of the event

```
CREATE TABLE doorbell_logs (
  log_id INT AUTO_INCREMENT PRIMARY KEY,
  doorbell_id INT NOT NULL,
  user_id INT NULL,
  event_type VARCHAR(100) NOT NULL,
  event_detail VARCHAR(255),
  event_timestamp DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (doorbell_id) REFERENCES doorbells(doorbell_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);
```

Example event_types include "NEW_USER_ADDED", "FACE_RECOGNIZED", "DOOR_OPEN", etc.

5.2 `recordings`

Column	Type	Constraints	Notes
recording_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique recording identifier
doorbell_id	INT	NOT NULL, FK to doorbells	Doorbell from which the recording was captured

Column	Type	Constraints	Notes
file_path	VARCHAR(255)	NOT NULL	Path or filename of the stored video/audio clip
created_at	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Recording creation time
duration_seconds	INT	(nullable)	Duration of the recording (optional)

```
CREATE TABLE recordings (
  recording_id INT AUTO_INCREMENT PRIMARY KEY,
  doorbell_id INT NOT NULL,
  file_path VARCHAR(255) NOT NULL,
  created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  duration_seconds INT,

  FOREIGN KEY (doorbell_id) REFERENCES doorbells(doorbell_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

---

6. (Optional) Sensor Data

`sensor_data`

Column	Type	Constraints	Notes
sensor_data_id	INT	AUTO_INCREMENT, PRIMARY KEY	Unique sensor data identifier
doorbell_id	INT	NOT NULL, FK to doorbells	Associated doorbell
sensor_type	VARCHAR(50)	NOT NULL	Type of sensor data (e.g., 'motion', 'temp', 'light')
sensor_value	VARCHAR(50)	NOT NULL	Recorded sensor value
reading_timestamp	DATETIME	NOT NULL, DEFAULT CURRENT_TIMESTAMP	Timestamp for the sensor reading

```
CREATE TABLE sensor_data (
  sensor_data_id INT AUTO_INCREMENT PRIMARY KEY,
  doorbell_id INT NOT NULL,
  sensor_type VARCHAR(50) NOT NULL,
  sensor_value VARCHAR(50) NOT NULL,
  reading_timestamp DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,

  FOREIGN KEY (doorbell_id) REFERENCES doorbells(doorbell_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

> Considerations as to what we might do next:

1. Set-up the pi as specified at the beginning.
2. Decide which parts of this schema we might truly need on the Pi versus what can be handled by the app.
3. Implement our server code (e.g., in Python or Node.js) to handle authentication, doorbell management, logging, etc.
4. Use scheduled scripts or cron jobs for tasks like auto-deleting recordings older than 72 hours.

Hope this clarifies certain things better. Feel free to let me know if you believe I missed something, or for further clarifications/amendments.