Software Requirements Specification for Sapphire Sounds

Lab 2 - Version 1.0

Prepared by: Zach Boudreaux, Wes Brown, Priscilla Cook, Zach Hendrickson, Shawn Maybush, Yahyah Mohamed, Christiney Ponton, Alisa Smanpongse, Brian Stiles

Old Dominion University

October 31, 2025

3.0 System Requirements

3.1 System Features

3.1.1 - User Interfaces (O: Boudreaux)

The Sapphire Sound system must provide a responsive, web-based graphical user interface accessible through modern browsers such as Google Chrome, Microsoft Edge, and Mozilla Firefox. The interface must be implemented using the React.js framework.

The system must present tenants and property managers with unique dashboards. Tenants should be presented with personalized dashboards that display current and historical noise level data, recent noise events, and their compliance status. The tenant dashboard will provide notifications when noise thresholds are exceeded for a configured duration, and tenants must be able to download or export summary reports for personal record keeping. The property manager dashboard must display a consolidated view of all connected units, with the ability to show noise event history for each tenant and unit. The property manager interface should include administrative features to configure threshold levels, manager accounts, and update tenant information. Generally, the design will employ consistent color schemes, fonts, and layout. The system should provide responsive feedback to the user, and all sensitive data fields will be masked during input and transmission.

3.1.2 - Hardware Interfaces (O: Maybush)

The Sapphire Sounds system requires specific hardware components and interfaces for noise level measurement and communication.

Acoustic Sensor Interface

- The sensor must interface with the controller using the I2C Bus (I2C protocol).
- The hardware component is the I2C Decibel Sound Level Meter Module.
- The sensor's purpose is to track how loud sounds are (decibel levels) without recording any audio.

Controller Hardware

- The controller is the Raspberry Pi Zero 2 W.
- The controller must be capable of processing sensor readings and sending them to the application server.
- The controller requires an operating system that supports the i2c-tools Linux Package to function as the Sensor Driver.

Communication Hardware

- The system requires a Wi-Fi connectivity module.
- The connectivity module is the Infineon CYW43439, which is built into the Raspberry Pi Zero 2 W.
- This module is used to transmit data in real-time using a cloud server.

3.1.3 Software Interfaces (O: Stiles)

Frontend - Backend Interface: The React.js web app interacts with the backend via REST API endpoints consisting of:

- User authentication and session management
- Retrieval and upload of sensor data
- Management and retrieval of Property Data for managers, including associated Unit Data, Sensor Data, and Tenant Data
- Retrieval of Unit and Sensor Data for Tenants

Backend - Database Interface: SQL queries for reading/writing sensor data, user data, property data, and rewards data.
Backend - Notification Interface: Decibel monitoring system evaluates sensor data for noise violations and delivers an alert to tenants using SMS or SMTP.

3.1.4 Communication Interfaces (O: Mohamed)

- The sensor device connects to the system through a secure wireless connection.
- The sensor transmits collected decibel readings to the application server for processing and storage.
- Data is transmitted automatically when noise thresholds are exceeded.
- The mobile and web applications communicate with the system backend to retrieve user data, noise history, and reports.
- The backend sends notifications to tenants when noise levels remain high for an extended duration.
- Property managers access incident reports and tenant information through a dedicated web interface that connects to the same backend system used by tenants.
- The system supports real-time data exchange between the sensor, server, and applications to maintain current information.
- All transmitted data includes time stamps to ensure accuracy and event traceability of events.
- The backend continuously updates the database to manage and store recorded sound level data for later reference.

3.2 Functional Requirements

3.2.1 - Authentication and Authorization (O: Boudreaux)

3.2.1.1 Stimulus

The Sapphire Sound application must implement a secure authentication and role-based authorization system to ensure that only authorized users gain access to the system, and they are only presented with the correct endpoints and dashboards.

3.2.1.2 Response

The system must require all users to authenticate with a unique username and password combination, it will store credentials using salted hashing consistent with current security best practices. The system must also support secure password reset via the verified email address, and must terminate sessions automatically after a defined period of inactivity. User roles must be associated with each user after successful authentication. The two possible roles are Tenant and Property Manager. Tenants will have

read-only access to their personal noise reports and profiles, while property managers will have read-only access to the reports for all their tenants, as well as administrative functions. Unauthorized attempts to access resources will be logged and denied with a standardized error response.

3.2.2 Tenant Rewards (O: Brown)

3.2.2.1 Stimulus

The user must initiate the “View Rewards” action by selecting the rewards menu option in the application’s navigation menu.

3.2.2.2 Response

The application must display a pop-up window listing all available rewards. Each reward entry in the pop-up menu must include: the reward title, a brief description of the reward, as well as the number of points required to redeem the reward. The user must be able to select a reward from the list displayed in the pop-up window to initiate the redemption process.

3.2.3 Tenant Management (O: Stiles)

3.2.3.1 Stimulus

A property manager clicks on the “Manage Tenants” button within the Unit interface.

3.2.3.2 Response

The application will open a new component in a pop-up, which will allow the manager to add a new tenant via user ID or remove an existing tenant.

3.2.4 Tenant Complaints (O: Maybush)

3.2.4.1 Stimulus

A tenant clicks a button to create a complaint in the UI, enters information, and saves the complaint.

3.2.4.2 Response

The complaint is recorded, and a notification is sent to the property manager(s).

3.2.5 Real-Time Notification (O: Hendrickson)

3.2.5.1 Stimulus

The noise level exceeds a predefined decibel threshold for longer than the configured duration.

3.2.5.2 Response

The application must send tenants a real-time notification through the mobile interface.

3.2.6 Property Manager's Tenant List View (O: Smanpongse)

3.2.6.1 Stimulus

A property manager clicks the dropdown button next to registered property names.

3.2.6.2 Response

The dropdown menu will show the property manager's tenants registered under the selected property.

3.2.7 Tenant Noise Data View (O: Cook)

3.2.7.1 Stimulus

A tenant logs in successfully.

3.2.7.2 Response

The system must let tenants view a list of their unit's noise levels with times and dates.

3.2.8 Logging Noise Levels Set By Property Manager (O: Ponton)

3.2.8.1 Stimulus

The property manager clicks the decibel level tab and enters a numerical value starting at 70 decibels.

3.2.8.2 Response

The system must log noise levels that meet or exceed the decibel threshold set by the user.

3.2.9 Historical Noise Logs for Dispute Resolution (O: Mohamed)

3.2.9.1 Stimulus

A property manager clicks the dropdown button next to registered property names.

3.2.9.2 Response

The dropdown menus will show historical noise logs for any unit to support dispute resolution.

3.3 Performance Requirements

3.3.1 Notification Alert (O: Hendrickson)

The system must send a noise alert to the tenant within 15 seconds after a noise event is detected.

3.3.2 Dashboard Updates (O: Hendrickson)

The system must show new noise events on the tenant dashboard within 20 seconds after they are logged.

3.3.3 Sensor Readings (O: Hendrickson)
The system must collect and evaluate decibel readings every 1 second to determine if a noise event is occurring.

3.3.4 Report Generation (O: Hendrickson)
The system must generate summary noise reports in under 5 seconds after the user requests them.

3.3.5 Claim Rewards (O: Hendrickson)
The system must allow tenants to view and claim eligible rewards through the application interface.

3.3.6 Historical Data (O: Mohamed)
The system must allow property managers to access historical noise logs for any unit to support dispute resolution.

3.4 Design Constraints

3.4.1 Compatibility Requirement (O: Brown)
The application must be compatible with all major web browsers, including Mozilla Firefox, Microsoft Edge, and Google Chrome. For the native versions of Sapphire Sounds, the application must support all major product lines of Apple iOS and macOS, as well as Android devices that use the Google Play Store.

3.4.2 Scalability Requirement (O: Brown)
The system architecture must scale to accommodate increasing user load demand and increased data and processing for noise detection during peak hours without service interruptions.

3.4.3 Hardware Requirement (O: Maybush)
The software must interface exclusively with the specified Raspberry Pi/Infineon/I2C hardware stack.

3.4.4 Privacy Requirement (O: Maybush)
The system must strictly adhere to a no-audio-recording policy.

3.5 Software System Attributes

3.5.1 Reliability (O: Ponton)
The system must provide a decibel reading once every second with an accuracy of no more than 2 decibels per second. The system must be resistant to sensor failures.

3.5.2 Availability (O: Ponton)
The application server and database hosting must maintain an uptime of 99.9% to ensure continuous alert processing and report generation.

3.5.3 Security (O: Ponton)
The system must store user passwords and Emails securely in the database without loss. All recordings of sound will not be stored, and only the report of the noise event.

3.5.4 Maintainability (O: Ponton)
The system must be properly organized and documented to allow for scheduled maintenance and regular updates.

3.5.5 User Portability (O: Ponton)

The software interface must be easy to access, traverse, and be viewable on different types of devices.

3.6 Other

N/A