SmartNest: Smart Home System

System Design Document

Team: Group B
Team Members: Heba Azeef & Harman Sandhu

Table of Contents

Introduction

Purpose of System

In the world of smart home automation, several smart devices with numerous functionalities are now available to everyday consumers. These devices allow users to control their homes remotely through smartphones, tablets, computers, and laptops, whether at home or away on vacation. The current market for smart home technology is rapidly expanding, with numerous companies developing various products, saturating the market with their unique ecosystems that use various communication protocols. These unique ecosystems pose issues to consumers in integrating their smart homes, as they require consumers to have multiple control consoles or to limit them to devices within the same brand umbrella. Most companies offer cloud-based server services with home automation products, allowing for remote control anywhere. However, this architecture's dependency on cloud servers poses limitations in cases of the loss of internet connectivity, which renders remote control impossible for the users. To address this, we have developed a multi-mode system that allows users to choose between a cloud-based server and a standalone mode that allows user devices and smart devices within the same local network to communicate. This system aims to transform how home environments are managed, by leveraging innovative technologies and industry best practices, to create a unified platform that seamlessly integrates smart devices, optimizes resource utilization, enhances user experience, and ensures compatibility and scalability for future enhancements.

Design Goals

Design goal	Traceability to non-functional requirements from the SmartNest RAD Document.
Scalability in terms of adding new users, devices, and rooms to the system/home unit.	- "The system should be scalable and capable of accommodating up to 25 connected devices and 5 concurrent users in a single home without compromising performance or functionality." - "The system should accommodate up to 200 online users creating new accounts or registering devices concurrently."
Strong **user support** facilitating simple, fast diagnosis and recovery from errors and potential user issues	"The system should provide informative notifications of the occurrence of an error to a user, less than one minute after the error occurs (e.g. loss of connectivity to specific devices, low battery warning on devices)"
Learnability: ensure an intuitive and easy-to-use user interface	"The system should be intuitive and easy to use, requiring minimal training for homeowners to operate effectively."

Usability: seamless integration facilitating use from different devices	"The system should be forward and backward-compatible with existing and new smart devices and technology."
Performance and **responsiveness**: the system should have fast responses to the client	- "The system should be responsive, with response times under 20 seconds for device commands and scheduled triggers" - "The system should be able to process the creation of a new home template (without devices) in under 1 minute."
Security: provide strict access control and secure storage of customer information.	- "SmartNest must enforce encryption for all stored user data to prevent unauthorized access by individual employees" - SmartNest will use 2FA to provide robust security safeguards, with an access control mechanism.

The Smart Nest system design aims to prioritize the user experience, performance, and ease of use to best meet customer needs. Trade-offs are made with other NFRs, such as system availability. While all non-functional requirements remain, the outlined design goals indicate which ones Smart Nest will prioritize and use to make trade-offs that maximize goals.

As the SmartNest system deals with sensitive user data and additionally has components related to Home Security, SmartNest prioritizes the security of user data as well as of the home system itself- in terms of access permissions regarding security devices, device control, remote access, etc.

Please refer to the SmartNest Requirements Analysis Document to view the entirety of non-functional requirements set out in the Smart Nest system. Please note that this document has been updated to better incorporate feedback for the non-functional requirements.

Definitions, Acronyms, and Abbreviations

RAD: Requirements Analysis Document

SDD: System Design Document

SHS: Smart Home Systems - Mostly about those in the current market.

MVC: Model, View, Controller- a system architecture

UI: User Interface

NFRs: Non-functional Requirements

RDBMS: Relationship Database Management System

2FA - Two Factor Authentication

Client: end-user device or software application that requests services or resources from a server.

Referenced Documents

Name	Link to document	Date Issued	Notes
System Proposal	SmartNest Proposal	January 25th, 2024	The design goals outlined in this System Design Document build off of the objectives set in the SmartNest System Proposal
Requirements Analysis Document (RAD)	ITEC 4010 Assignment …	February 18th, 2024	This System Design Document heavily refers to and builds off of the SmartNest RAD. Boundary conditions and system architecture are decomposed based on key use cases outlined in the RAD. As well, activity and sequence diagrams from the RAD will be used as a reference point to develop mapping and flow. Please note that additional changes have been made to the RAD to better incorporate feedback and best meet the needs of this SDD.

Current Software Architecture

The current Smart Home Systems (SHS) in the market are designed as web-based applications, enabling seamless communication between user interfaces and central servers. The reason this web-based design is used is because it enables user-friendly interactions, real-time changes, and fast data transmission. In the current market smart home systems are structured using a 3-tier, Model-View-Controller (MVC), and client/server architecture.

The 3-tier architecture divides the system into 3 distinct players that consist of the presentation layer (interface), the application logic layer, and the storage layer. The presentation layer is used for user interactions with the system through the mobile app or web portal to capture the input and deliver it to the next layer in a way it will understand. The application layer is used to process user requests, control devices, automate tasks, handle events, process data, and ensure smooth operation of the overall system. The storage layer acts as a repository for various types of data that are critical to the system's functionality. This includes user profiles, device configurations, device statuses, automation rules/schedules, and security logs.

The client/server architecture involves smart devices (clients) that interact with a central server to manage system functions and user interactions. The clients in this architecture are various smart devices within the smart home that send and receive data to and from the central server. The central server acts as the heart of the Smart Home System, managing all system functions, data processing, and user

interactions. This architecture provides strong centralized control, scalability, reliability, and enhanced security for the entire smart home ecosystem.

The MVC architecture divides the application into three main components: the model, view, and controller. The model represents the data and business logic of the application, managing essential information such as user profiles, device configurations, automation rules, and security logs. On the other hand, the View acts as the presentation layer, offering interfaces like mobile apps or web portals for user interactions with the system. It displays the data from the model and captures users' inputs, forwarding them to the controller. The controller acts as the mediator handling user requests communicating with the model for data operations, executing business logic, and updating the view accordingly. This architecture structure provides a clear separation of concerns, and potential components for reuse, allowing for easier testing, scalability, and improved overall user experience.

The SmartNest home system will address common issues in current systems by overcoming significant restrictions found in existing systems. With a strong focus on real-time responsiveness, users can expect swift execution of commands and expect instant feedback on the status of the device. Another issue the new system will address is the limitations in scalability and flexibility for the integration of new users and devices without sacrificing the system's performance. Enhanced customization features allow users to personalize the automation rules and device configurations to suit their preferences. Robust data security measures provide maximum privacy protection through the use of authentication and encryption mechanisms.

Proposed Software Architecture

Overview

The Smart Nest system will use a 3-tier architecture, which facilitates an interface, application logic, and storage layer. This model was chosen as it best facilitates the Smart Nest functional requirements by allowing a boundary for user interactions. This allows for communication to interact with the application logic layer that contains all the necessary tools to facilitate user requests for several functional requirements; including controlling devices, creating new rooms, etc as outlined in the Requirement Analysis Document. Lastly, data (ie. user permissions, devices, etc) will all be stored in a secure database that uses an entity-relationship model.

Subsystem decomposition

Our SmartNest home automation system features an intuitive user interface, including a mobile app and web platform. These interfaces serve as the presentation layer within the 3-tier architecture, enabling users to interact with the system. The choice of this design is optimal because it provides a user-friendly experience for managing the smart home ecosystem. Users can issue requests to the application server through the interface, gaining control over their devices, customizing scenes, and managing preferences. This layer's design prioritizes ease of use and accessibility, ensuring that users can seamlessly navigate and control their smart home devices.

The application logic layer of the SmartNest system acts as the centralized control hub with advanced automation features. This layer is designed to be the centralized control hub because it is responsible for managing user commands, facilitating communication with connected devices, and handling the system's advanced automation features. Users can establish ‘rooms’ and allocate specific permissions within the ecosystem through the application server. The design of this layer supports multi-user access, creating a hierarchical control structure to ensure diverse users have suitable permissions. Additionally, personalized notifications and alerts can be configured through this layer, providing real-time updates on specific events. This layer's design optimally handles the complexity of user interactions and system notifications, ensuring efficient and effective management of the smart home system.

The data layer of the SmartNest system serves as the repository for storing user profiles, device configurations, automation rules, and security logs. This layer's design supports multi-user access and personalized notifications, integral components of the system's functionality. The system's extensive support for a wide range of smart devices and integration capabilities highlights the inherent adaptability of the 3-tier architecture. SmartNest seamlessly integrates new devices using protocols such as WiFi, Bluetooth, Serial, X10, and ZWave, showcasing the system's scalability and future-proofing. The design of this layer ensures efficient data storage and retrieval, supporting the system's adaptability to a broad spectrum of devices and user preferences.

In conclusion, the SmartNest 3-tier architecture's design choices ensure the system design goals are prioritized. A simple 3-tier architecture rather than a client/server architecture that is common for existing systems prioritizes system performance by reducing the possibilities of bottlenecks. This architecture facilitates evenly distributed processing without the added complexity of a more intricate model that would potentially further complicate the system, which would exacerbate performance issues. Additionally, the presentation layer provides a user-friendly interface, the application logic layer acts as a centralized control hub for advanced automation, and the data layer supports efficient data storage and integration with a wide range of devices. This design ultimately guarantees a seamless and user-centric experience for managing a smart home ecosystem.

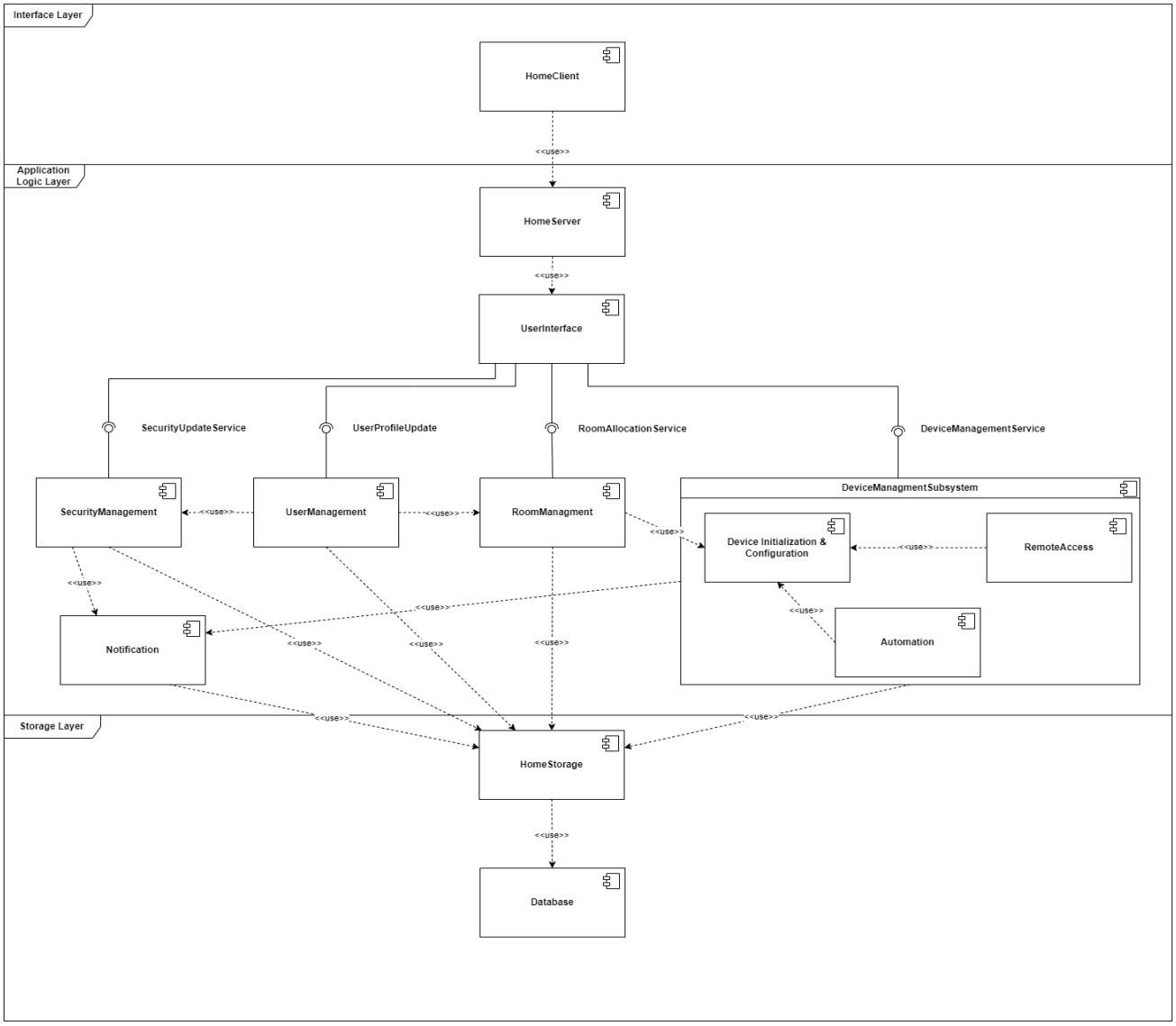

Figure 1.1: Subsystem Decomposition for SmartNest Smart Home System. The dashed arrow indicates dependencies between subsystems.

Indicated below are the key components and the functionality overview of each.

Component:	**Definition:**
Home Client	This acts as the central boundary the user sees and interacts with. This will handle user input, windows displayed, forms, etc.
Home Server	The central brain of SmartNest, stores data, manages connected devices, and hosts a user interface. Provides an intuitive platform for users to control their smart home, issue commands, customize settings, and manage preferences

	within the home network.
User Interface	The purpose of this component is to format and send what needs to be displayed to the user.
Security Management	Controls the security features of the smart home, such as arming and disarming the alarm, and monitoring, and detecting movement through security sensors.
User Management	Handles the creation of user accounts, user access permission, account creation, adding of additional users, and authentication.
Room Management	Manages the creation, setup, and ongoing organization of rooms within the home.
Notification	The notification component handles all notifications sent to the user, including those regarding security, devices, and more.
Device Management Subsystem	The device management component encapsulates several components related to the use and management of smart home devices linked to the Smart Nest home automation system.
Device Initialization and Configuration	Facilitates adding new devices to a smart home system, configuring their permissions, setting up access to the device, and startup/shutdown occurrences.
Remote Access	Allows users to directly view and control their smart home devices from their devices (phone, tablet, desktop, etc).
Automation	This component allows users to preconfigure automation events and conditions- such as scenes and schedules. This component is then responsible for automatically executing the occurrence when the preconfigured condition is met (ie. time reached or device state changed)
Home Storage	Acts as both a local database and a memory for SmartNest, reducing dependencies by securely storing data such as user profiles, device configurations, and security logs within the smart home system itself.
Database	The database contains all data essential to system functions, including information about user accounts, access permissions, rooms, devices, security, and more. In this system decomposition, the database is general, however it will be broken down as shown in the persistent data section of this document.

Hardware/software mapping

The SmartNest smart home system employs a hybrid approach, combining local network-based and internet-based cloud components to create a reliable, scalable, and adaptable platform. Its 3-tier architecture strategically assigns tasks, ensuring smooth user experiences both at home and when accessed remotely. However, this integration of multiple nodes in the system also presents a range of challenges.

When managing the communication between multiple nodes many challenges arise due to the increase in complexity which makes it harder to maintain smooth data flow and interactions. This makes systems more susceptible to network bottlenecks, more dominantly during larger traffic volumes when these nodes are being overloaded. The SmartNest system provides effective load balancing, routing, and fault tolerance mechanisms to handle these issues and provide a higher level of scalability.

In terms of software reuse, reusing parts across many nodes presents a unique set of challenges. This is largely due to the incompatible components within a system that may develop over time and disrupt the functionality of the system. This is where the task of managing updates and versions across nodes becomes increasingly important and can potentially result in vulnerabilities if not corrected accordingly.

Within the cloud environment, SmartNest's application logic takes charge of essential functions such as user authentication, scheduling, automation, and more, benefiting from the efficiency of cloud processing. The database securely stores user profiles, device configurations, automation rules, and security logs, enabling access to historical information from any location. Meanwhile, within the local network, the HomeServer acts as a central communication point for local devices, managing device communication and executing automation rules that do not rely on internet access. Every smart device comes equipped with a built-in controller, connecting directly to the HomeServer for swift and reliable logic execution.

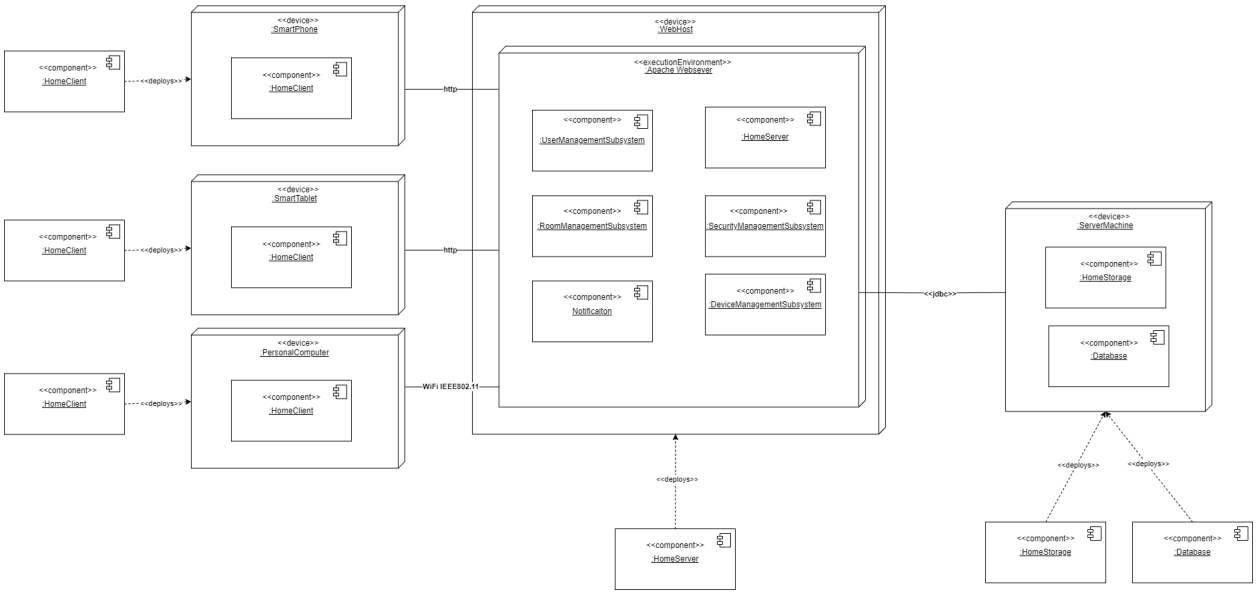

Figure 1.2: Deployment Diagram representing the hardware mapping of the allocation of components to different nodes. The HomeClient on personal devices can access WebServer which provides access to the smart home system for modification and storage of user preferences, device configurations, and more.

Persistent data management

In the 3-architecture of the SmartNest SHS, a Relational Database Management System (RDBMS) is a key component for efficient data management. Through the use of APIs like JDBC, the system ensures it can scale up and manage structured data effectively. This database storage is essential for managing the relationships between various parts of the SHS. It handles data related to rooms, user permissions, user management, device configuration, and security logs. This RDBMS offers a structured approach for storing and retrieving data, which is critical to the operation and growth of the system.

The RDBMS organizes the data into tables with predefined schemas, in our case user, device, security, room, notification, and automation are created to effectively store relevant data. The user table maintains user management information like user credentials, user permissions, and authentication which is required. On the other hand security table stores data related to alarm status, sensor information, and various security logs. One of the most important ones is the device table which is crucial for storing device-specific details like device ID, type, and location. The room table contains details about the various rooms spaced within the home ecosystem, facilitating the organization and management of devices within the home ecosystem. Any notification is stored in the notification table which includes information about various alerts sent to the user whether it's for adding a new device, security notification, user interactions, monitoring the status of devices, etc. Finally, the automation table would be responsible for managing automation events and conditions that are set and stored by the user. These tables' relationships represent the data that is stored through the persistence object to the database with their connections, supporting efficient querying and ensuring data integrity. In conclusion, these components are crucial as they provide the foundation for efficient data storage, management, and integration with the SmartNest System.

Access control and security

The handling of potential security issues, such as data breaches or unauthorized access is a major point of prioritization for the SmartNest system. Thus we have designed some tools to ensure user safety.

Authentication: Authentication will be handled by the user management component. As per the SmartNest RAD, our system will use 2-factor authentication (2FA). This will be done with the user's password, and then a push to their device. As the user has to log in to be able to access their home system, 2FA will provide an added layer of security for the entire home system.

Encryption: All data will be encrypted when transferred and stored in the SmartNest database. For the secure transfer of data, we plan to use HTTPS and WPA3 encryption protocols to ensure secure movement of data. This was chosen as they work well with HTTP and WIFI, which we indicated would be used in the Hardware/Software mapping section. Additionally, HTTPS works well for a client/server or

tiered architecture. As our system architecture brings about unique security challenges, it's important to address them directly.

Management of keys: The management of keys is essential for access control and security, to prevent unauthorized access. An electronic key management system simplifies this process by automatically verifying access permission for users. Each time a key is used to access the security system, home server, or specific smart devices, an audit trail is created. This level of monitoring makes sure that the smart home can only be operated by those who are authorized users, preventing unverified access and improving the system's general security as well as the security of the system's assets.

Access control on the SmartNest system is divided into two main user groups: General users and admin users. Users can switch between those groups by having the admin of a home system either appoint them as an admin or de-assign them from an admin position.

This Access Control Matrix best facilitates the SmartNest design goals outlined at the beginning of the document:

Actors Objects	Admin User	General User
Home	CreateHomeSystem() AddUserToHome() DeleteUserFromHome()	CreateHomeSystem()
Room	ConfigureNewRoom()	
User	AddAccount() EditAccount() DeleteAccount()	AddAccount() EditAccount() DeleteAccount()
Device	AddDevice() ControlDevice() AddDevicePermissions()	ControlDevice()
Schedule	CreateSchedule()	CreateSchedule()
Scene	CreateScene()	
Security	CheckSensorRecords() MonitorCameras() ArmSystem() DisarmSystem() ConfigureSecurityDevice() DeleteSecurityDevice()	CheckSensorRecords() MonitorCameras()
Notification	ReceiveDeviceNotification()	ReceiveDeviceNotification()

Global control flow

The following list categorizes the SmartNest subsystems identified in the subsystem decomposition into procedure-driven, event-driven, or threaded control flows for synchronization. Additionally, it identifies how messages flow between subsystems, and how requests are initiated:

1. The ‘User Interface’ subsystem uses an event-driven control flow. This is because the main loop waits for an event from an external user. A user interacts with the SmartNest system user interface (UI) to provide their input/actions, which are predefined events. As a result, the necessary objects are dispatched. However, they are not invoked in a specific sequence, which also differentiates it from a procedure-driven flow.

2. The SmartNest ‘Room Management’ subsystem uses a procedure-driven control flow. This is because the user interacts with the smart nest system to manage a room. This subsystem requires user input has a specific predefined sequence and specifically waits for a user to respond, which differentiates it from event and threaded control flows.

3. The ‘User Management’ subsystem uses a procedure-driven control flow. This is because user management-related activities are procedures, each done step by step. Each step waits for user input before proceeding to the next

4. The ‘Device Initialization subsystem uses a procedure-driven control flow as this subsystem has predefined procedures for setup/integration. When user interaction is needed, for example in the case of device configuration and permissions control, the system will wait for a response to ensure integration and proper management of concurrent access.

5. The Smart Nest ‘Remote Access Management’ uses an event-based architecture. The main loop of this subsystem waits for an external event- from the authorized user. There is no set sequence, and objects for specific devices are dispatched when the appropriate event is initiated.

6. The Smart Nest ‘Device Automation” subsystem utilizes a thread-based architecture. During device configuration, the user sets up a specific procedure for each device. This subsystem requires information inputted into the device configuration subsystem. The configured procedure is then followed, in the provided sequence and parallel for each automated device.

7. The ‘Security Management’ subsystem uses an event-driven control flow. When certain pre-configured external events occur, the subsystem dispatches specific objects as needed. External events in this subsystem include external devices, security breaches detected, end users, and so on. There is no set order for these events, they simply occur based on what security events occur. For security and user safety reasons, event-driven control is the best option for this subsystem.

8. The Notification subsystem utilizes an event-based control flow. This is because the Smart Nest system waits for an event within the system to occur with any of the connected devices. This event will be received from the device subsystem or externally, which will then trigger the necessary objects. Because different types of user notifications can occur, and they each have unique procedures, event-based is the most appropriate control flow. When an event occurs, such

as a temperature device detecting a temperature below 30°, the system responds by sending a notification to the client.

Concurrency and synchronization issues:

1. The most notable concurrency issue is the potential for a lost update, which occurs in this case if multiple threads attempt to update home or device permissions at the same time. To prevent this, When an Admin user attempts to edit information related to a device or home system, the SmartNest system blocks all of the users from editing the data at the same time.

2. However, the SmartNest solution for lost updates can lead to deadlocks, which means several threads are indefinitely waiting for each other to unlock so they can then read or edit device information for example. Our solution for this issue is timely, strong concurrency control. Our system software additionally involves an in-depth analysis of potential deadlocks to best avoid them.

3. Similarly, another concurrency and synchronization issue is inconsistent retrievals and the potential for race conditions, which is if users attempt to view a current home or device status while it is being updated by another user. This can result in users viewing the wrong information and in turn taking actions based on the incorrect information. The SmartNest system aims to address this by providing strict locking mechanisms when any user is writing to an object. Additionally when using a thread-based control flow the system will carefully coordinate between threads.

4. Another potential synchronization issue is related to the fact that we have two different user groups and various levels of permission for each device. If a change has been made to a user's access permissions for a home system, that change must probably synchronize across all devices. If this does not occur then there's a potential of a security breach as unauthorized users can easily access devices they should not be able to access. Additionally, it could also prevent a user from getting into their home system when they have already been permitted to access it.

Boundary conditions

Following are the key boundary conditions addressing system startup/shutdown, and system error behavior. Please note that this section is not inclusive of all boundary conditions, but prioritizes those that cover the largest scope of boundary conditions.

Start-up:

Start Home Server	The AdminUser starts the HomeServer. If the server was not cleanly shut down, this use case invokes the CheckDataIntegrity boundary use case described in the error behavior section. As soon as the initialization of the server is complete, AdminUsers and GeneralUsers can initiate any of their use cases.
Startup after network outage	When a user refreshes the page and network availability has been detected, the “Startup after network outage” boundary condition will occur. If the user was in the process of entering data meant to be saved

	to the system database that had not yet been saved, the Smart Nest system will automatically open and display the last open page alongside any recovered data upon startup.

Shutdown:

ShutDown Home Server	The AdminUser stops the HomeServer. The server terminates any ongoing Automation and stores any cached data. Once this case is completed the general users cannot access or modify the SmartNest.

Error Behavior:

Handle Network Outage	This boundary condition encapsulates the case that a network outage occurs during the time a client is using the system, or a network is not available. In the event of a network outage, the Smart Nest system will not be able to communicate with home devices. in this condition, the Smart Nest system will respond by displaying a “network unavailable” image on the screen.
Detected Device Failure	In the event of a device hardware failure or an external software failure connected to a linked device, the system will immediately send the user a notification indicating the occurrence, time, location, and relevant device details.
Failed to verify	This boundary use case encapsulates similar use cases that involved the failure to verify a user-inputted field in a Smart Nest input form. For example, in the ‘Create Account’ use case, this boundary condition can be invoked if the user does not enter a valid email address. An additional example is if an invalid username is entered. Both cases will occur in the ‘user’ component. If verification has failed, the system will provide an on-screen error message, indicating the source of the failure by outlining the failed input field in red. This boundary case aims to handle the verification issue immediately, within the same interface page and the same component. This is to facilitate the performance and supportability of design goals
CheckDataIntegrity	This boundary case occurs if the Smart Nest system detects the system was not shut down cleanly, meaning an unexpected error occurred during shutdown or a page failed to save correctly. This event will invoke the Check Data Integrity, which accesses the Database component and conducts standard tests to verify that the current data is accurate and complete.

Convert Persistent Storage	When HomeServer is shut down, the AdminUser can covert the persistent storage from flat file storage to a database storage or from a database storage to a flat file storage.

Please note that the SmartNest RAD has been updated to cover the additional boundary cases.

Glossary

Device Database: Stores details of the device (names, room, readings) to be accessed and managed by the user.

Schedule Subsystem: Internal scheduling system that monitors selected devices and performs specific actions when a time threshold is reached.

Security Subsystem: Responsible for arming or disarming any security-class devices around the house. The system then sends alerts to authorities or admin users if any of the security devices are activated.

Alert Subsystem: Automatically alerts select users to their phone or device when specific unwanted scenarios happen.

Connectivity Subsystem: Allows for all devices to connect over a unified network. Identifies devices and catalogs features presenting them through a unified interface.

User: Profile for customer stakeholders. Allows access to specified methods based on the profile's admin rights.

Button: An on-screen prompt the user can press to initiate the desired process.

Dashboard: Accesses the device database and displays the data to the user so that they can manage their device details easily.

Room: An add-on of the Home class, allows the user to customize their preferences and helps the system process diagnostic data collected by specific devices throughout the home.

Home: A grouping of users, rooms, and devices- made to represent an actual home. Each home allows for a unique setup and permissions.

Schedule: Linked to specific devices. Allows for specific actions to take place on a said device when the internal clock reaches a selected time.

Device: Broad smart tools placed throughout the home. Linked to each room, these devices can be diagnostic or recreational. Examples (are temperature scanners, and smart TVs).

Notification: Alerts and messages delivered to users' preferences with details of issues or confirming actions.

User notification: An alert sent to the user interface, indicating to them changes in the system or with devices. Permissions can be changed as a user desires.