# System and Software Architecture Description (SSAD)

**Smart Locks Control**

**Team 5**

| Team members | Roles |
|---|---|
| VaibhavVishal | Project Manager, Life Cycle Planner,Trainer |
| Diego Brandao | IIV&V, Software Architect, Developer |
| Zhe Wang | Feasibility Analyst, NDI/NCS Acquirer |
| Mohammadreza Barazesh | Software Architect, Developer |
| Alejandro Monroy | Prototyper, Evaluator, Tester, Developer |
| Hao-Yun Yang | Requirements Engineer |
| Katarzyna Ruszowska | Quality Focal Point, UML Modeler, Developer |

**11/26/2017**

# Version History

| Date | Author | Version | Changes made | Rationale |
|---|---|---|---|---|
| 10/10/17 | Diego | 1.0 | • Create initial SSAD document for Fundamental Commitment Package<br>• Completed sections: Introduction, System Analysis Overview, System Analysis Rationale and Architectural Styles, Patterns and Frameworks.<br>• Added diagrams: Artifacts and Information, Domain Model, Sequence, Deployment, Hardware Component, and Software Component. | • Used in Fundamental Commitment Package |
| 10/14/17 | Katarzyna | 1.1 | • Added Use Case Diagram and Process Description | • Used in Fundamental Commitment Package |
| 10/16/17 | Diego | 1.2 | • Added Class Diagram<br>• Added REST Service Class Diagram<br>• Removed multiple maintainers on System Context Diagram based ARB feedback.<br>• Renamed actors to be consistent with other documents based on ARB feedback. | • Final DC package version |
| 12/01/17 | Diego | 1.3 | • Updated status of the document<br>• Updated software component diagram<br>• Updated deployment diagram<br>• Added Frontend class diagrams<br>• Added Backend class diagrams<br>• Updated Domain Model diagram<br>• Added more Sequence diagrams<br>• Updated Architectural Styles, Patterns, and Frameworks | • Final Delivery |

# Table of Contents

# Table of Tables

# Table of Figures

# 1. Introduction

## 1.1 Purpose of the SSAD

The purpose of the SSAD document is to provide an architectural view of the project. It includes several diagrams, each showing the system from a different viewpoint. It illustrates the operational concept, the designed architecture, the proposed implementation and the reasoning behind the architectural choices.

## 1.2 Status of the SSAD

This is the version 1.3 of this document, which will be submitted along with the as-built package. The document has been completed. However, it's important to point out that some diagrams have been left outside on purpose. For example, the independent technology section, because we already knew the technology that we were going to use and the time constraints for a one-semester project. Also, at this point, we did not find it useful to go back and design Robustness diagrams given that we already had designed Sequence diagrams.
We included sequence diagrams for non-trivial and most important functions of the system. Therefore, we did not include trivial functions like add/update/delete users and login.

# 2. System Analysis

## 2.1 System Analysis Overview

The principal purpose of the Smart Lock Control System is to allow small-scale Airbnb houses and Vacation Rentals the ability to control their smart locks remotely. The current process requires someone to change the codes physically every time that a new guest leaves. The Smart Lock Control system can perform multiple operations on smart locks such as lock, unlock, change codes, assign multiple codes to one lock, generate random codes, group locks, see entry logs, and disable lock access either temporarily or permanently. In addition, the system provides user management via roles and permissions allowing the house manager or administrator to control which user has access to a given lock.

### 2.1.1 System Context

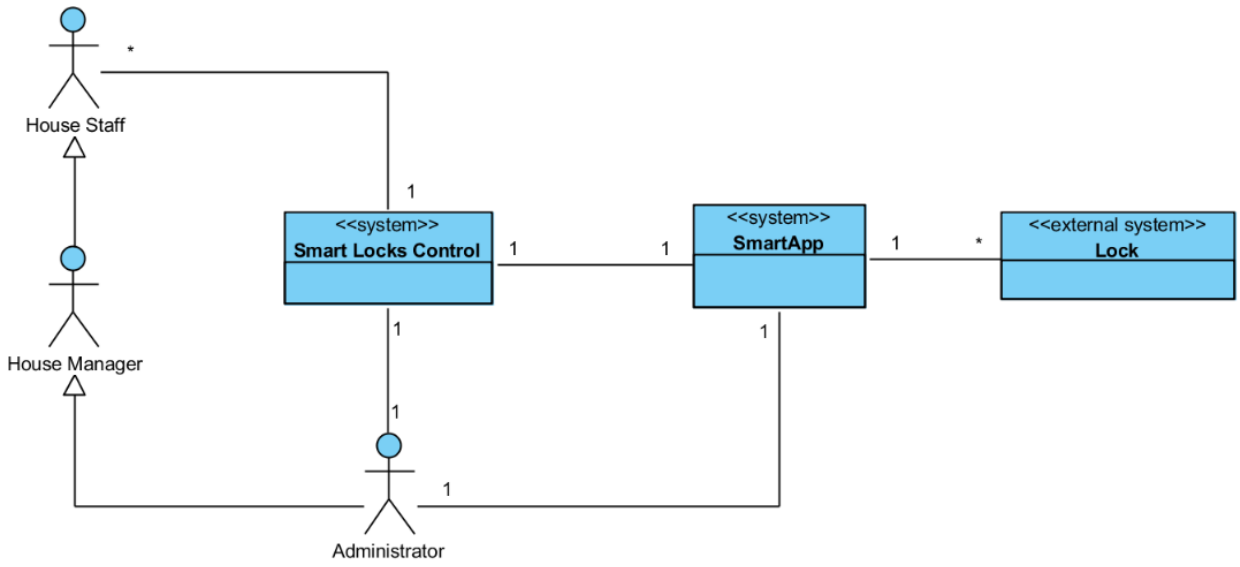


**Figure 1: System Context Diagram**

**Table 1: Actors Summary**

| Actor | Description | Responsibilities |
|---|---|---|
| Administrator | The administrator of the system is the person who authenticates into the SmartThings account to enable the communication between the Smart App and the Smart Locks Control application. Also, the administrator holds the master code for the all the locks in the house. It can be the owner or someone assigned by the owner. | • Configures the SmartApp<br>• Gets the client id and client secret from the SmartApp.<br>• Adds the SmartApp web service authentication information.<br>• Manages any type of user account, including resetting passwords.<br>• Creates groups of locks<br>• Access to all features available to the manager and other users. |
| House Manager | The house manager is either the Airbnb house owner, or the someone assigned by the owner. | • Manages lock codes.<br>• Disable/Enable lock codes.<br>• Manages the accounts of the users below the management level (Other).<br>• View the devices (locks and hubs) status.<br>• View lock usage log.<br>• Views lock number, name, and their own code. |
| House Staff | represent house keepers, maintainers, and any other staff member. | • Views lock number, name, and their own code. |
| SmartApp | The Smart App was built in the SmartThings platform, and its intent is to allow the Smart Locks Control application to perform operations on the locks via REST APIs. | • Provides the Smart Locks Control application with an interface for sending commands to the locks. |
| Lock | A lock is an electromechanical smart lock which is designed to perform locking and unlocking operations on a door using codes entered in a keypad or commands send by the Smart Lock Control application. | • Sends events to the SmartApp.<br>• Receives commands from the SmartApp. |

*Manage is used to represent all the basic operations (add/update/delete).

## 2.1.2 Artifacts & Information

This section of the document describes the artifacts and information created by the system. These artifacts can either be used by the end user or reused by the system to produce further artifacts. The artifacts and relationships between/among them are described below.

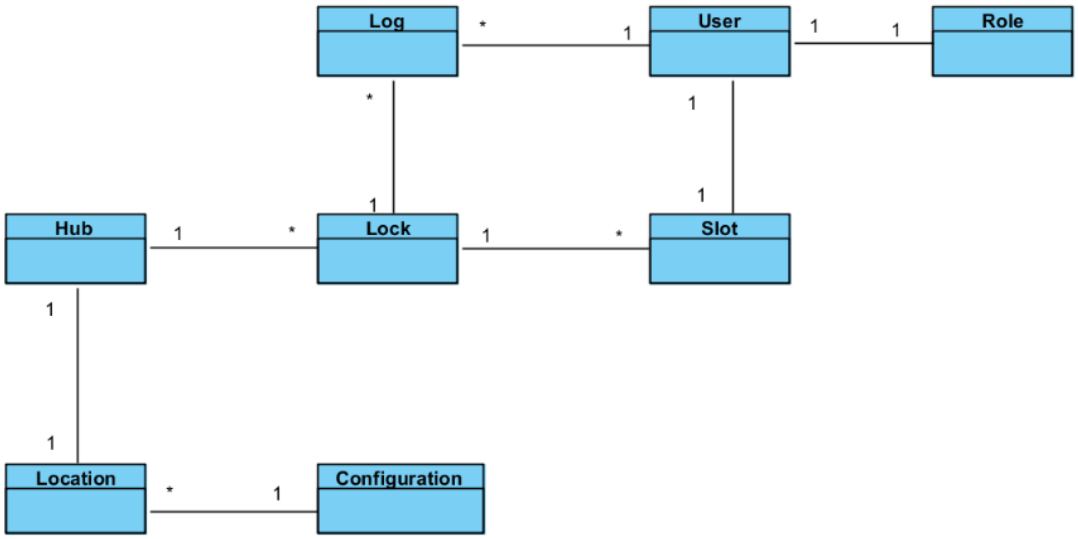


**Figure 2: Artifacts and Information Diagram**

**Table 2: Artifacts and Information Summary**

| Artifact | Purpose |
|---|---|
| Configuration | Contains the client identification and client secret used to start the authentication process into the SmartThings account. |
| Location | Contains the identifier, name and the location unique URI, which is used to access the locks via REST services. |
| Hub | Contains information about the hub used to connect the locks to the SmartThings account. It includes name, identifier, and status. |
| Lock | Contains information about the smart lock, including its name, location, status, battery, the slots containing the access codes, and its universal identifier. |
| Slot | Contains the user assigned to the slot, the code to open the smart door, the number of the slot and a flag indicating if the slot is enabled or not. |
| User | Contains the name of the user, password, role, and a flag indicating if the user is enabled or not. |
| Role | It defines the type of user and their permissions. The first release includes three roles (admin, manager, other). |
| Log | Contains the log of the a given lock, including the action performed (locked/unlocked) |

### 2.1.3 Behavior

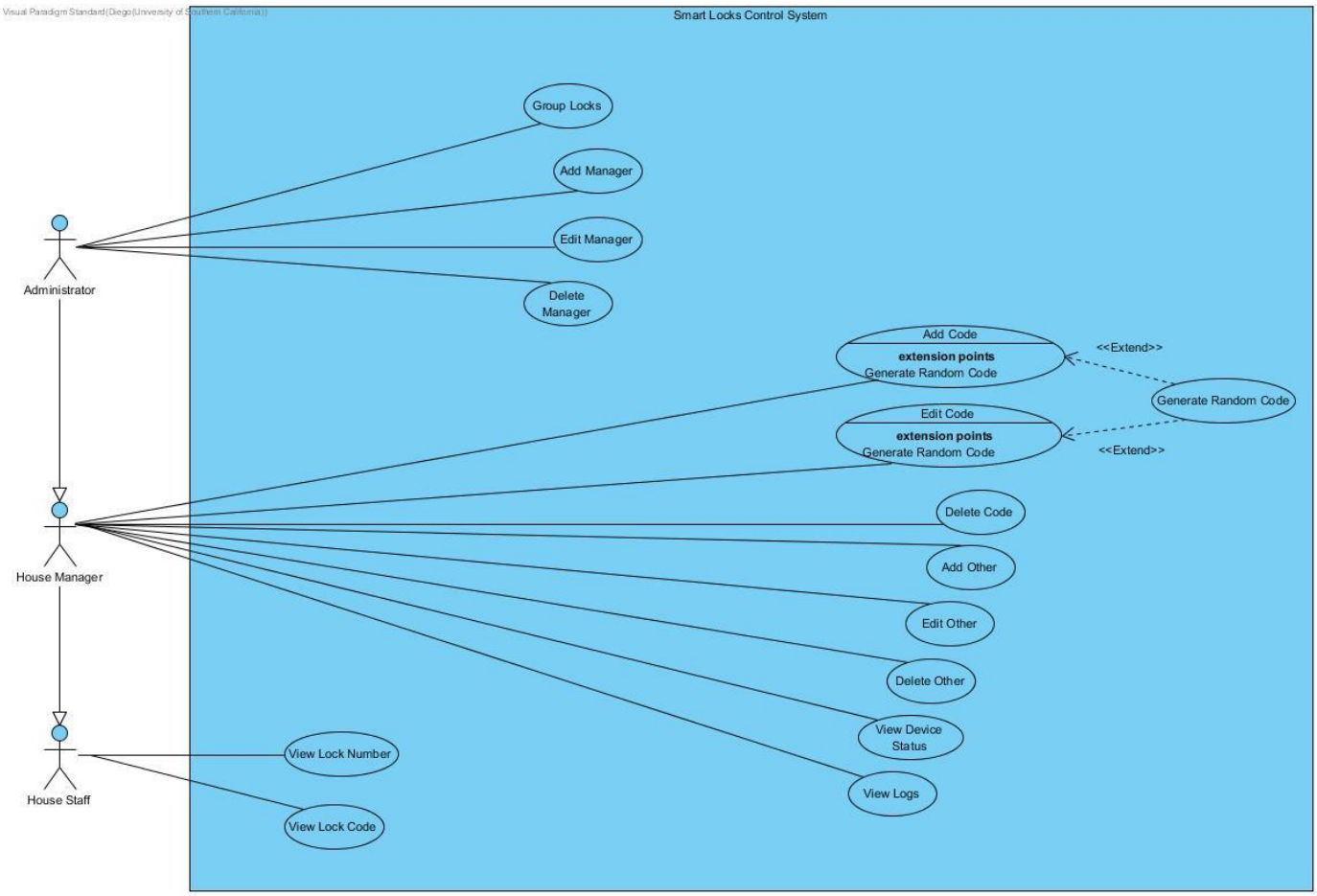


**Figure 3: Process Diagram**

**Table 3: Use Case List**

| Use Case Sequence | System's Response |
|---|---|
| UC-1 | Add User |
| UC-2 | Edit User |
| UC-3 | Delete User |
| UC-4 | Add Slot |
| UC-5 | Edit Slot |
| UC-6 | Delete Slot |
| UC-7 | Group Locks |
| UC-8 | View Logs |
| UC-9 | View Device Status |
| UC-10 | View Lock Number and Code |

# 2.1.3.1 User Account Control

## 2.1.3.1.1 Add User

**Table 4: Add User - Process Description**

| | |
|---|---|
| **Identifier** | UC-1: Add User |
| **Purpose** | Create a new user account. |
| **Requirements** | WC_4577: As an administrator, I can add/delete 'house managers' and 'house staff' and help them reset their password.<br>WC_4517: As a house manager, I should be able to add/delete others. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level to create account of access level requested for the new account. |
| **Post-conditions** | New account with the input details exists. |

**Table 5: Typical Course of Action - Add User: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "User Info" tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on "Add User" | |
| **4** | | Redirect to a page with input fields for information about the new user. |
| **5** | Fill in all required fields with unique and correctly formatted information. | |
| **6** | Click "Create". | |
| **7** | | Check for uniqueness, correct format, and completion. |
| **8** | | Update database. |
| **9** | | Display "user successfully created" message. |

**Table 6: Exceptional Course of Action - Add User: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "User Info" tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on "Add User" | |

| | | |
|---|---|---|
| **4** | | Redirect to a page with input fields for information about the new user. |
| **5** | Fill in fields with incorrectly formatted, incomplete, duplicate, or no information. | |
| **6** | Click “Create”. | |
| **7** | | Check for uniqueness, correct format, and completion. |
| **8** | | Display error message. |

#### 2.1.3.1.2 Edit User

**Table 7: Edit User - Process Description**

| | |
|---|---|
| **Identifier** | UC-2: Edit User |
| **Purpose** | Edit an existing user account. Includes changing the name, password, or access level. |
| **Requirements** | WC_4577: As an administrator, I can add/delete ‘house managers’ and ‘house staff and help them reset their password.<br>WC_4517: As a house manager, I should be able to add/delete house staff. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level to edit account of access level of selected account. |
| **Post-conditions** | Requested changes applied to the selected account. |

**Table 8: Typical Course of Action - Edit User: Success**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “User Info” tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on “Update” next to the name of the account the user wishes to edit. | |
| **4** | | Redirect to a page with options for how to update the account. |
| **5** | Make the desired changes to the account. | |
| **6** | Click “Save”. | |
| **7** | | Update the database. |
| **8** | | Display “user successfully updated” message. |

**Table 9: Exceptional Course of Action - Edit User: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "User Info" tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on "Update" next to the name of the account the user wishes to edit. | |
| **4** | | Redirect to a page with options for how to update the account. |
| **5** | Make the desired changes to the account. | |
| **6** | Click "Save". | |
| **7** | | Display error message. |

#### 2.1.3.1.3 Delete User

**Table 10: Delete User - Process Description**

| | |
|---|---|
| **Identifier** | UC-3: Delete User |
| **Purpose** | Delete an existing user account. |
| **Requirements** | WC_4577: As an administrator, I can add/delete 'house managers' and 'house staff' and help them reset their password.<br>WC_4517: As a house manager, I should be able to add/delete house staff. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level to delete account of the access level of the selected account. |
| **Post-conditions** | Deleted account no longer exists. |

**Table 11: Typical Course of Action - Delete User: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "User Info" tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on "Delete" next to the name of the account the user wishes to delete. | |
| **4** | | Update the database. |

| | | |
|---|---|---|
| **5** | | Display “user successfully deleted” message. |

**Table 12: Exceptional Course of Action - Delete User: Failure**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “User Info” tab. | |
| **2** | | Redirect to a page displaying user information. |
| **3** | Click on “Delete” next to the name of the account the user wishes to delete. | |
| **4** | | Display error message. |

## 2.1.3.2 Lock Management

### 2.1.3.2.1 Add Slot

**Table 13: Add Slot - Process Description**

| | |
|---|---|
| **Identifier** | UC-4: Add Slot |
| **Purpose** | Add a new slot to a lock and assign a code to it. |
| **Requirements** | WC_4573: As a house manager, I can select one or multiple locks to add/delete/edit codes.<br>WC_4524: As an administrator, I should be able to allow more than one working code per lock.<br>WC_4522: As an administrator or house manager, I should be able to assign random codes through a random number generator. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level. |
| **Post-conditions** | The new slot will work for the selected lock. |

**Table 14: Typical Course of Action - Add Slot: Success**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to add a new slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |

| | | |
|---|---|---|
| **5** | Click on “Add Slot”. | |
| **6** | | Redirect to a page with input fields for information about the new slot. |
| **7** | Fill in all of the required input fields with correctly formatted, unique information. | |
| **8** | Click “Add”. | |
| **9** | | Verify the format, completeness, and uniqueness of the input information. |
| **10** | | Generate a random code if the “New Code” input field was left blank. |
| **11** | | Update the database. |
| **12** | | Display “slot successfully added” message. |

**Table 15: Exceptional Course of Action - Add Slot: Failure**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to add a new slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |
| **5** | Click on “Add Slot”. | |
| **6** | | Redirect to a page with input fields for information about the new slot. |
| **7** | Fill in the fields with incorrectly formatted, incomplete, or no information. | |
| **8** | Click “Add”. | |
| **9** | | Display error message. |

#### 2.1.3.2.2 Edit Slot

**Table 16: Add Slot - Process Description**

| | |
|---|---|
| **Identifier** | UC-5: Edit Slot |
| **Purpose** | Edit an existing slot. |
| **Requirements** | WC_4573: As a house manager, I can select one or multiple locks to add/delete/edit codes.<br>WC_4522: As an administrator or house manager, I should be able to assign random codes through a random number generator. |

| | |
|---|---|
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level. |
| **Post-conditions** | The requested changes will be applied to the selected slot. |

**Table 17: Typical Course of Action - Edit Slot: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Locks" tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to edit a slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |
| **5** | Click on "Update" next to the slot the user wishes to edit. | |
| **6** | | Redirect to a page displaying options for how to edit the lock slot. |
| **7** | Make the desired changes to the slot. | |
| **8** | Click "Save". | |
| **9** | | Generate a new code if the user has left the code field blank. |
| **10** | | Update the database. |
| **11** | | Display "slot successfully updated" message. |

**Table 18: Exceptional Course of Action - Edit Slot: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Locks" tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to add a new slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |
| **5** | Click on "Update" next to the slot the user wishes to edit. | |
| **6** | | Redirect to a page displaying options for how to edit the lock slot. |
| **7** | Make the desired changes to the slot. | |

| | | |
|---|---|---|
| **8** | Click “Save”. | |
| **9** | | Display error message. |

#### 2.1.3.2.3 Delete Slot

**Table 19: Delete Slot - Process Description**

| | |
|---|---|
| **Identifier** | UC-6: Delete Slot |
| **Purpose** | Delete a slot from a lock. |
| **Requirements** | WC_4573: As a house manager, I can select one or multiple locks to add/delete/edit codes. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level. |
| **Post-conditions** | Slot will no longer work for selected lock. |

**Table 20: Typical Course of Action - Delete Slot: Success**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to add a new slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |
| **5** | Click on “Delete” next to the code the user wishes to delete. | |
| **6** | | Update the database. |
| **7** | | Display “slot successfully deleted” message. |

**Table 21: Exceptional Course of Action - Delete Slot: Failure**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the lock for which the user wishes to add a new slot. | |
| **4** | | Redirect to a page displaying the information for the selected lock. |

| | | |
|---|---|---|
| **5** | Click on “Delete” next to the code the user wishes to delete. | |
| **6** | | Display error message. |

#### 2.1.3.2.4 Group Locks

**Table 22: Group Locks- Process Description**

| | |
|---|---|
| **Identifier** | UC-7: Group Locks |
| **Purpose** | Create a lock group for better organization and in order to facilitate lock management. |
| **Requirements** | WC_4525: As an administrator, I can create a group of locks and add/delete locks in the group. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to a Smart Locks Control account with administrator access level. |
| **Post-conditions** | Group of the selected locks exists in the system. |

**Table 23: Typical Course of Action - Group Locks: Success**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the locks to add to the lock group. | |
| **4** | Click “Create Lock Group”. | |
| **5** | | Update the database. |
| **6** | | Display “lock group successfully created” message. |

**Table 24: Exceptional Course of Action - Group Locks: Failure**

| Seq# | Actor’s Action | System’s Response |
|---|---|---|
| **1** | Click on “Locks” tab. | |
| **2** | | Redirect to a page displaying lock information. |
| **3** | Select the locks to add to the lock group. | |
| **4** | Click “Create Lock Group”. | |
| **5** | | Display error message. |

#### 2.1.3.2.5 View Logs

**Table 25: View Logs- Process Description**

| | |
|---|---|
| **Identifier** | UC-8: View Logs |
| **Purpose** | View the logs of lock access for a specific lock. |
| **Requirements** | WC_4456: As an administrator or house manager, I should be able to view and filter logs of lock access. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level. |
| **Post-conditions** | System will display log data for the selected lock. |

**Table 26: Typical Course of Action - View Logs: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Logs" tab. | |
| **2** | | Display log selection screen. |
| **3** | Select the lock for which the user wishes to view the log. | |
| **4** | Click "Get Log". | |
| **5** | | Display the log for the selected lock. |

**Table 27: Exceptional Course of Action - View Locks: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Logs" tab. | |
| **2** | | Display log selection screen. |
| **3** | Select the lock for which the user wishes to view the log. | |
| **4** | Click "Get Log". | |
| **5** | | Display error message. |

## 2.1.3.3 Device Status

### 2.1.3.3.1 View Device Status

**Table 28: View Device Status- Process Description**

| | |
|---|---|
| **Identifier** | UC-9: View Device Status |
| **Purpose** | View the status of the hardware: whether the hub is online or offline and whether the lock is locked or unlocked. |
| **Requirements** | WC_4519: As an administrator or house manager, I should be able to see the hub (offline/online) and lock (lock/unlocked) status. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account of sufficiently high access level. |
| **Post-conditions** | System will display status of the selected hubs or locks. |

**Table 29: Typical Course of Action – View Device Status: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Device Status" tab. | |
| **2** | | Redirect to device selection page. |
| **3** | Select the devices for which the user wishes to view the status. | |
| **4** | Click "View Status". | |
| **5** | | Display the status(es) of the selected device(s). |

**Table 30: Exceptional Course of Action – View Device Status: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Device Status" tab. | |
| **2** | | Redirect to device selection page. |
| **3** | Select the devices for which the user wishes to view the status. | |
| **4** | Click "View Status". | |
| **5** | | Display error message. |

#### 2.1.3.3.2 View Lock Number and Code

**Table 31: View Lock Number and Code - Process Description**

| | |
|---|---|
| **Identifier** | UC-10: View Lock Number and Code |
| **Purpose** | View assigned lock number and code. |
| **Requirements** | WC_4593: As a 'house staff', I should be able to see given lock number and lock code. |
| **Development Risks** | None |
| **Pre-conditions** | User logged in to Smart Locks Control account. |
| **Post-conditions** | System will display lock number and code assigned to the user. |

**Table 32: Typical Course of Action – View Lock Number and Code: Success**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Lock Info" tab. | |
| **2** | | Redirect to a page showing the lock number and code assigned to the user. |

**Table 33: Exceptional Course of Action – View Lock Number and Code: Failure**

| Seq# | Actor's Action | System's Response |
|---|---|---|
| **1** | Click on "Lock Info" tab. | |
| **2** | | Display error message. |

### 2.1.4 Modes of Operation

The system will not have multiple modes. Therefore, no description could be stated in this section.

## 2.2 System Analysis Rationale

The first phase of this system, which is part of our deliverables is targeting only small-scale Airbnb homes and vacation rental houses. The goal is to target other types of business such as study rooms, office spaces, motels and hotels in next phases of the project.

An important point is that the Smart Lock Control system relies on a Smart App, also developed as part of this delivery and deployed to the SmartThings platform which is responsible for communicating with the locks. Therefore, creating a SmartThings account, registering the hub(s), lock(s) and deploying the provided Smart App is pre-requisite for the system's operation.

Regarding the Smart Locks Control system, one of its main requirements of the client in addition to the lock management feature was a hierarchical user management system, where the access could be controlled based on the type of the user. Therefore, the major operational stakeholders of the system are categorized as administrator, managers, and others. The functions performed by each user are described in the System Context Diagram description.

The categories are determined in the system as roles and each role has certain privileges in the application. This role-based permission control allows the system to grow in the future to support more use cases.

# 3. Technology-Independent Model

This section was left out on purpose, due to the time constraints of the project and the fact that we already know the technology that will be used. Therefore, we moved the content of this section to the next section and updated the design overview and design rationale. This includes the conceptual domain model which is depicted as “Domain Model Class Diagram.”

# 4. Technology-Specific System Design

## 4.1 Design Overview

### 4.1.1 System Structure

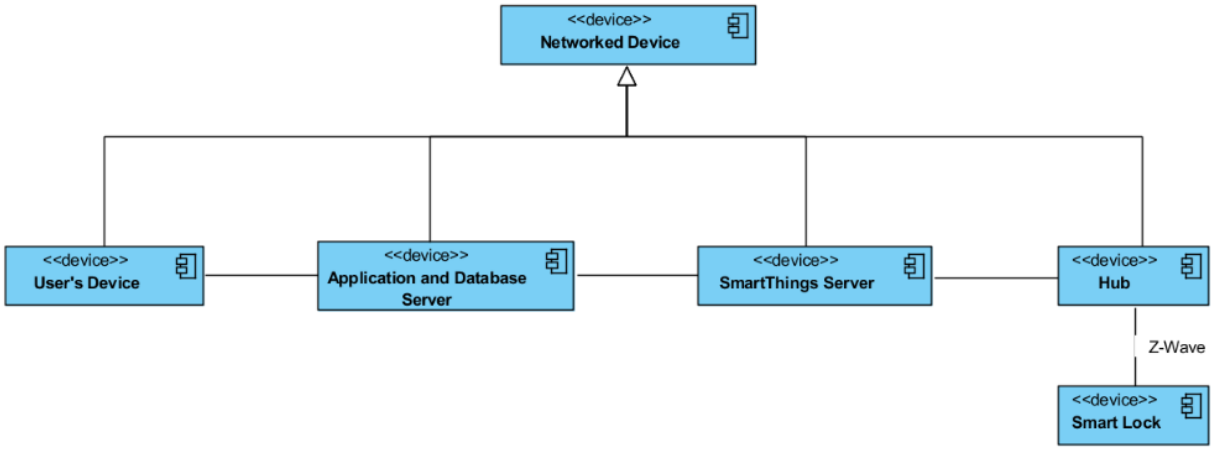


**Figure 4: Hardware Component Class Diagram**

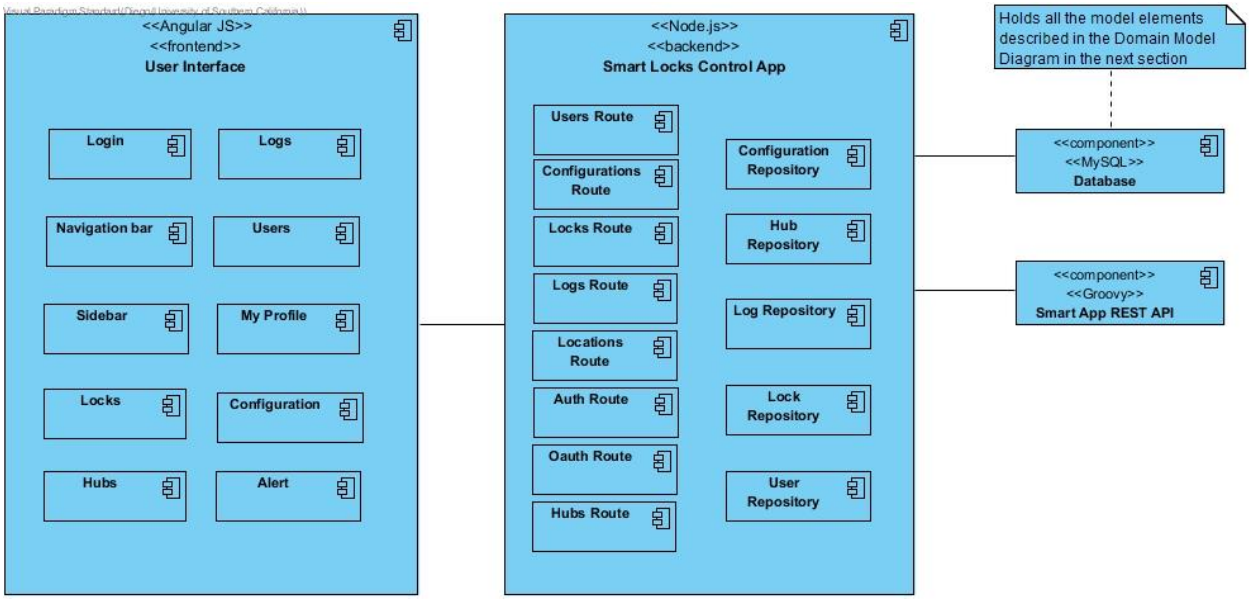


**Figure 5: Software Component Class Diagram**

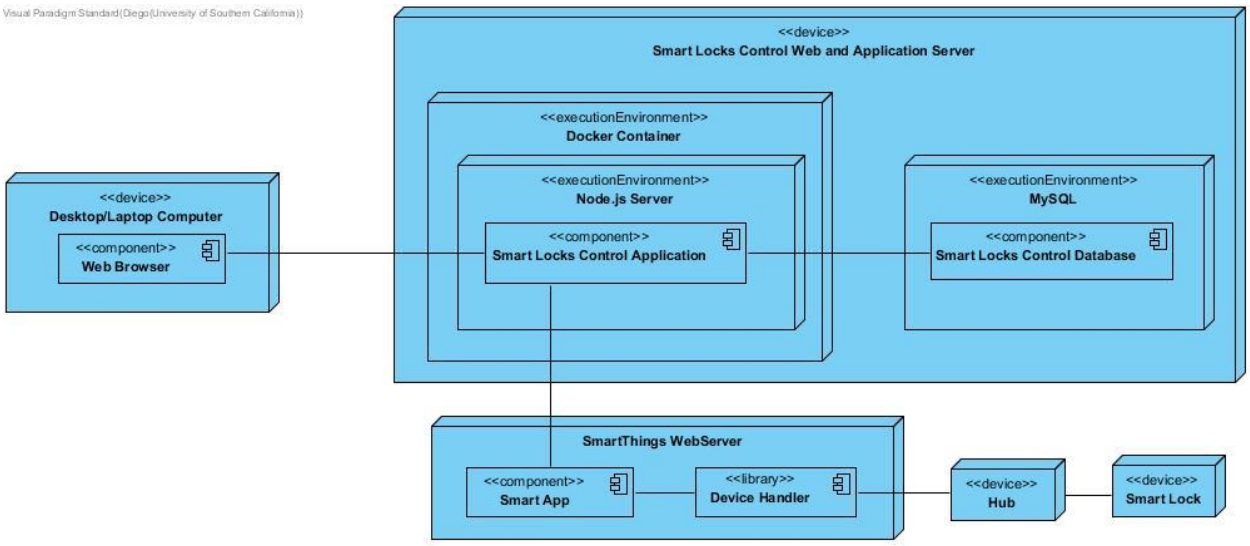


**Figure 6: Deployment Diagram**

**Table 34: Hardware Component Description**

| Hardware Component | Description |
|---|---|
| Desktop/Laptop Computer | User's computer with a web browser |
| Application and Database Server | Ubuntu Linux server containing the docker container running a container with the Node.js server and the smart locks control web application. This server also contains the MySQL database server hosting the database of the application. |
| SmartThings WebServer | SmartThings web server containing the Smart App REST API and the Device Handler used to perform operations on the Smart Lock |
| Hub | SmartThings Hub used to connect the smart lock to the SmartThings web server. The hub communicates to the smart lock via Z-Wave network and a uses the internet to connect to SmartThings Web server. |
| Smart Lock | The smart lock device which is what the Smart Locks Control system is controlling, it connects to the hub via a Z-Wave network. |

**Table 35: Software Component Description**

| Software Component | Description |
|---|---|
| Login | It manages all functionality related to user login screen, validation of username and password. |
| Navigation bar | It is shown at the top of screen. Shows name of logged in user and logout button. |
| Sidebar | It is shown at the left-hand side of the screen. It shows all the components the user can interact with. |
| Locks | It manages all the functionality related to remote control of locks. Actions such as add, delete and update slot codes are supported. |
| Hubs | It shows information related to hub status (online/offline) and information related to all locks in the hub (status, battery). |

| | |
|---|---|
| Logs | It shows a LOG of all the actions that have been performed involving the locks. |
| Users | Manages functionality related to users. Can handle actions such as add, delete and update users. |
| My Profile | Allows users to modify their passwords. |
| Configuration | Allows user to give permissions to the app to hubs and locks on Samsung SmartThings account. |
| Alerts | Manages success and error alerts. |
| Configurations Route | This component handles all the requests targeting a 'configuration' resource. It provides all the basic operations via http GET/POST/PUT methods. |
| Locks Route | This component handles all the requests targeting a 'lock' resource. It provides all the basic operations via http GET/POST/PUT methods. |
| Logs Route | This component handles all the requests targeting a 'log' resource. It provides all the basic operations via http GET/POST methods. |
| Auth Route | This component handles the user authentication via http POST. It checks if the user exists returning either a token or an error. |
| Oauth Route | This component is responsible for authenticating to SmartThings and also for retrieve the locations, hubs and locks. |
| Hubs Route | This component handles all the requests targeting a 'hub' resource. It is responsible for updating the Hub's status and locks and returning them to the frontend. |
| Users Route | This component handles all the requests targeting a 'user' resource. It provides all the basic operations via http GET/POST/PUT/DELETE methods. |
| Location Repository | Repository used to hide the logic used to persist/retrieve a location to/from the database. |
| Hub Repository | Repository used to hide the logic used to persist/retrieve a hub to/from the database. |
| Lock Repository | Repository used to hide the logic used to persist/retrieve a lock and its slots to/from the database. |
| User Repository | Repository used to hide the logic used to persist/retrieve a user to/from the database. |
| Configuration Repository | Repository used to hide the logic used to persist/retrieve a configuration to/from the database. |
| Log Repository | Repository used to hide the logic used to persist/retrieve a log event to/from the database. |
| Database | This component represents the database, which holds all the model elements of the system. A detailed description of the model elements can be seen in the diagram "Domain Model Class Diagram". |
| Smart App REST API | This component represents all the services deployed to Samsung SmartThings and used to interact with the lock. A description of the services available is provided in the diagram "REST Services Class Diagram" |

## 4.1.2 Design Classes

### 4.1.2.1 Domain Model Class Diagram

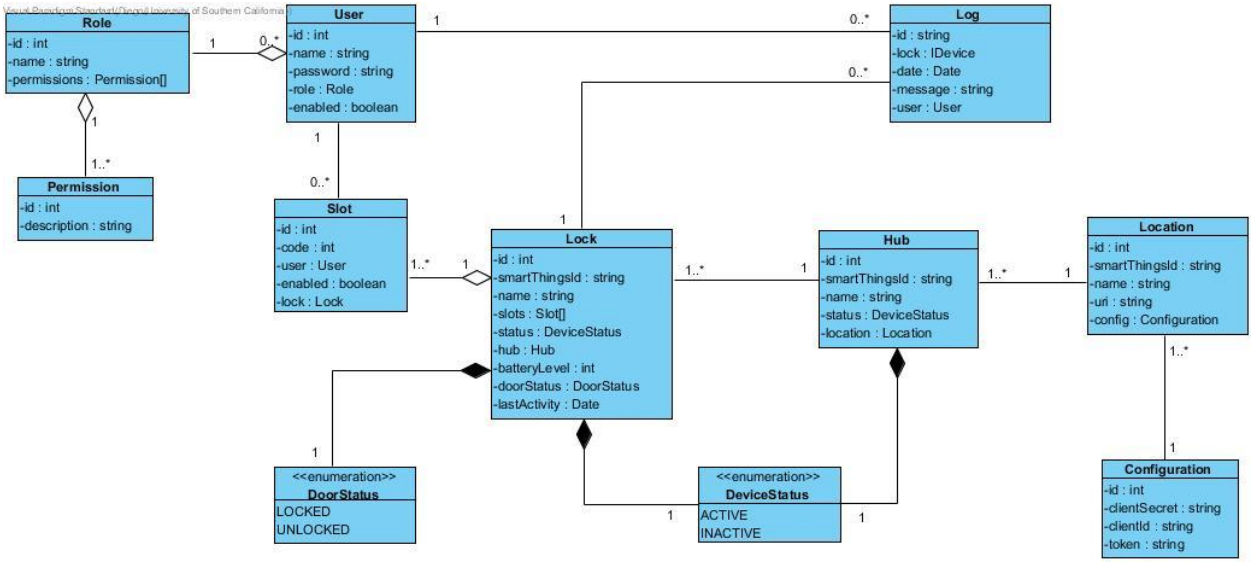


**Figure 7: Domain Model Class Diagram**

**Table 36: Design Class Description**

| Class | Type | Description |
|---|---|---|
| Role | Model | This model entity represents a function assumed by a given user. It used to define the users can access in the system. |
| Permission | Model | This model entity represents an action that a user can perform. This class has been added for future permission support. |
| User | Model | This model entity represents the users that can access the system, what the user access is determined based on his/her attached role. It's also possible to create a user without any role e.g., (Guest). |
| Slot | Model | This model entity represents a slot in a given lock. The main function of the slot is to hold the code to access a lock. |
| Lock | Model | This model entity represents a smart lock. It holds information about the device that the system is controlling. |
| Log | Model | This model entity holds the logs/events of a given log/hub. E.g. (Log the entries of a given user). |
| Configuration | Model | This model entity is used to store the configuration used to access the SmartThings account. |
| Hub | Model | This model entity holds information about the hub used to access a given lock. |

| | | |
|---|---|---|
| Location | Model | This model entity represents a logical location defined in the SmartThings site. Each location has a unique URI to access its devices. |
| DeviceStatus | Enumeration | Enum used to hold the status (Active/Inactive) of the devices. |
| DoorStatus | Enumeration | Enum used to hold all status of the door (Locked/Unlocked). |

### 4.1.2.2 Frontend Services Class Diagram

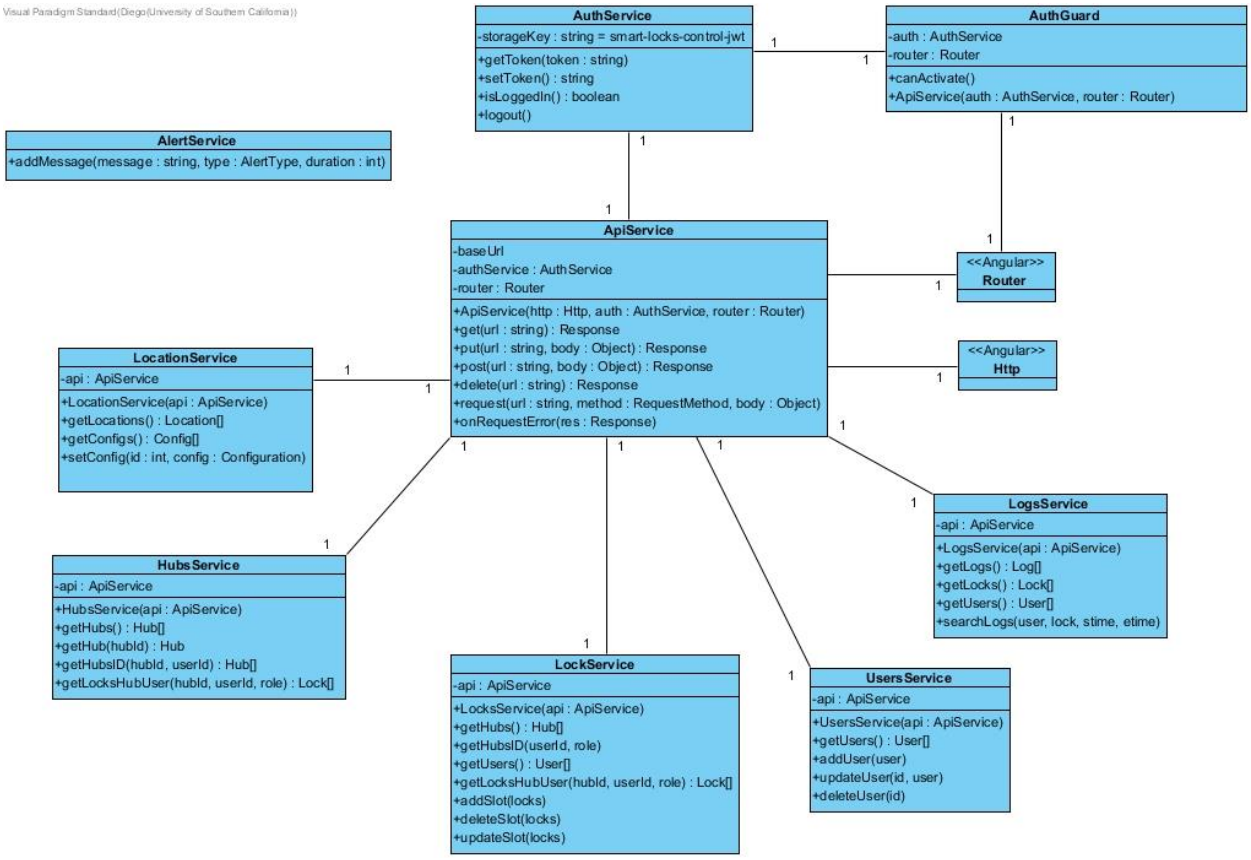


**Figure 8: Frontend Services Class Diagram**

**Table 37: Frontend Services Class Diagram Description**

| Class | Type | Description |
|---|---|---|
| AuthService | Service | Service used to save the token in the client's local storage. It also checks if the user has a token and if it is not expired. |
| AuthGuard | Service | Guard service used to block the user accessing sections of the application based on the user's role. |
| ApiService | Service | Service used to communicate with the backend. It provides basic HTTP methods that are used by other services to retrieve/store information. |
| AlertService | Service | Service used to manage alert message displayed to the user. |
| LocationService | Service | Service used by the components to retrieve the locations, configurations and for updating the configuration. |

| | | |
|---|---|---|
| HubsService | Service | Service used by the components to retrieve the list of hubs. It also provides methods to retrieve the updated status of the hubs and locks. |
| LocksService | Service | Service used to retrieve and update the slots that a given user can add/delete/update. |
| UsersService | Service | Service used to update/delete/add a user. |
| LogsService | Service | Service used to search logs. |

* Not including here classes provided by Angular (Http, Router)

### 4.1.2.3 Frontend Components Class Diagram

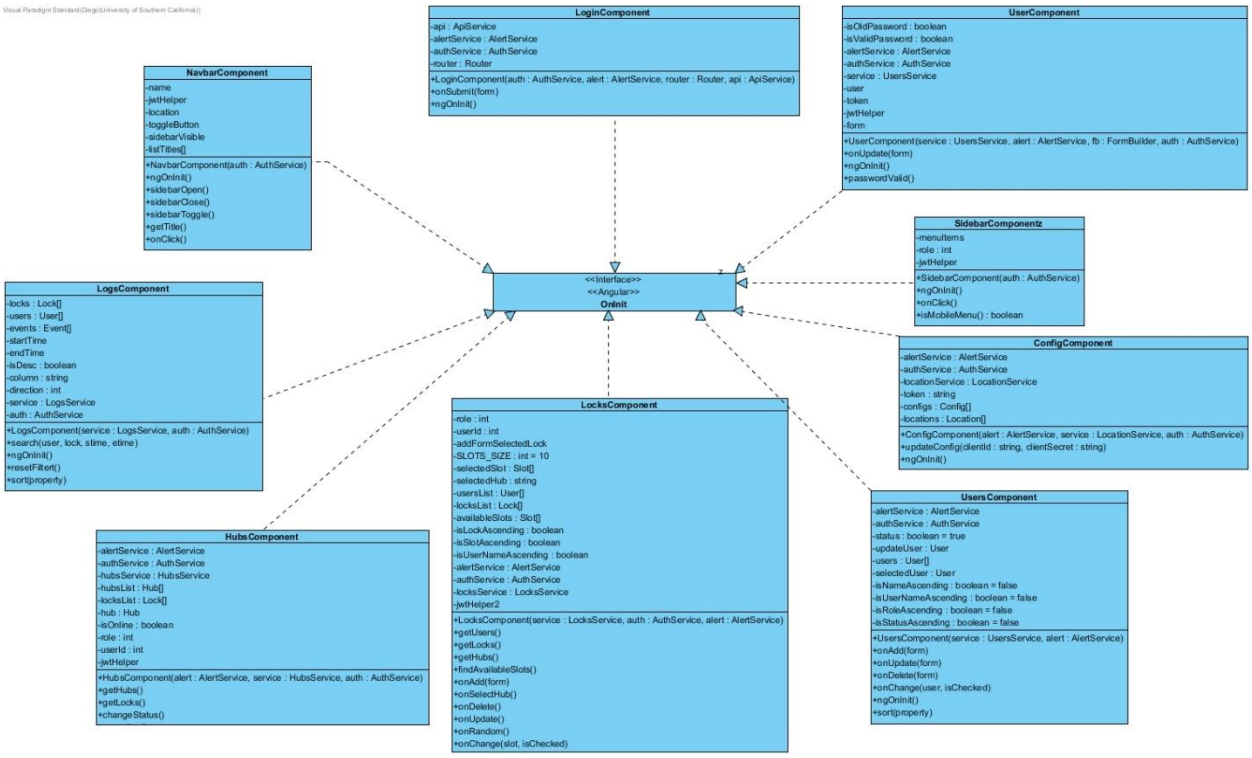


**Figure 9: Frontend Components Class Diagram**

**Table 38: Frontend Components Class Diagram Description**

| Class | Type | Description |
|---|---|---|
| LogsComponent | Controller | It manages the actions performed by the user, such as search and sort. It is a bridge between the HTML page and the logs service. |
| HubsComponent | Controller | It manages the actions performed by the user, such as listing status of the locks and the status of the hub. It is a bridge between the HTML page and the hubs service. |
| LocksComponent | Controller | It manages the actions performed by the user, such as retrieving and managing the slots. It is a bridge between the HTML page and the locks service. |
| UsersComponent | Controller | It manages the actions performed by the user, such as add/delete/update a user. It is a bridge between the HTML page and the users service. |
| ConfigComponent | Controller | It manages the actions performed by the user in the configuration page, such as updating the config id and secret and logging to SmartThings. It is a bridge between the HTML page and the location service. |

| | | |
|---|---|---|
| UserComponent | Controller | It manages the actions performed by the user in the profile. It is a bridge between the HTML page and the users service. |
| SidebarComponent | Controller | Shows name of logged in user and logout button. |
| LoginComponent | Controller | It manages the actions performed by the user on the login page. |
| NavbarComponent | Controller | It shows all the components the user can interact with. |
| OnInit | Interface | Angular lifecycle hook that is called after data-bound properties of a directive are initialized. |

#### 4.1.2.4 Repository Class Diagram

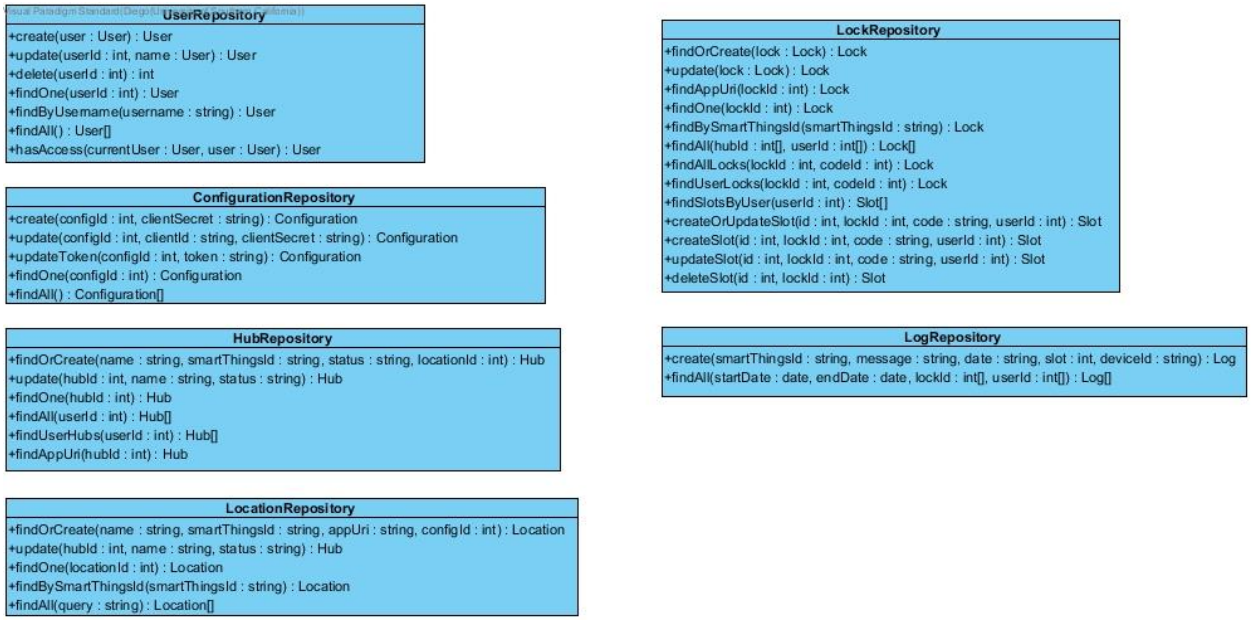


**Figure 10: Repository Class Diagram**

**Table 39: Repository Class Diagram Description**

| Class | Type | Description |
|---|---|---|
| ConfigurationRepository | Repository | Repository used to hide the logic used to persist/retrieve a configuration to/from the database. |
| LockRepository | Repository | Repository used to hide the logic used to persist/retrieve a lock and its slots to/from the database. |
| HubRepository | Repository | Repository used to hide the logic used to persist/retrieve a location to/from the database. |
| LogRepository | Repository | Repository used to hide the logic used to persist/retrieve a log event to/from the database. |
| UserRepository | Repository | Repository used to hide the logic used to persist/retrieve a user to/from the database. |
| LocationRepository | Repository | Repository used to hide the logic used to persist/retrieve a location to/from the database. |

### 4.1.2.5 Backend RESTful Services Class Diagram

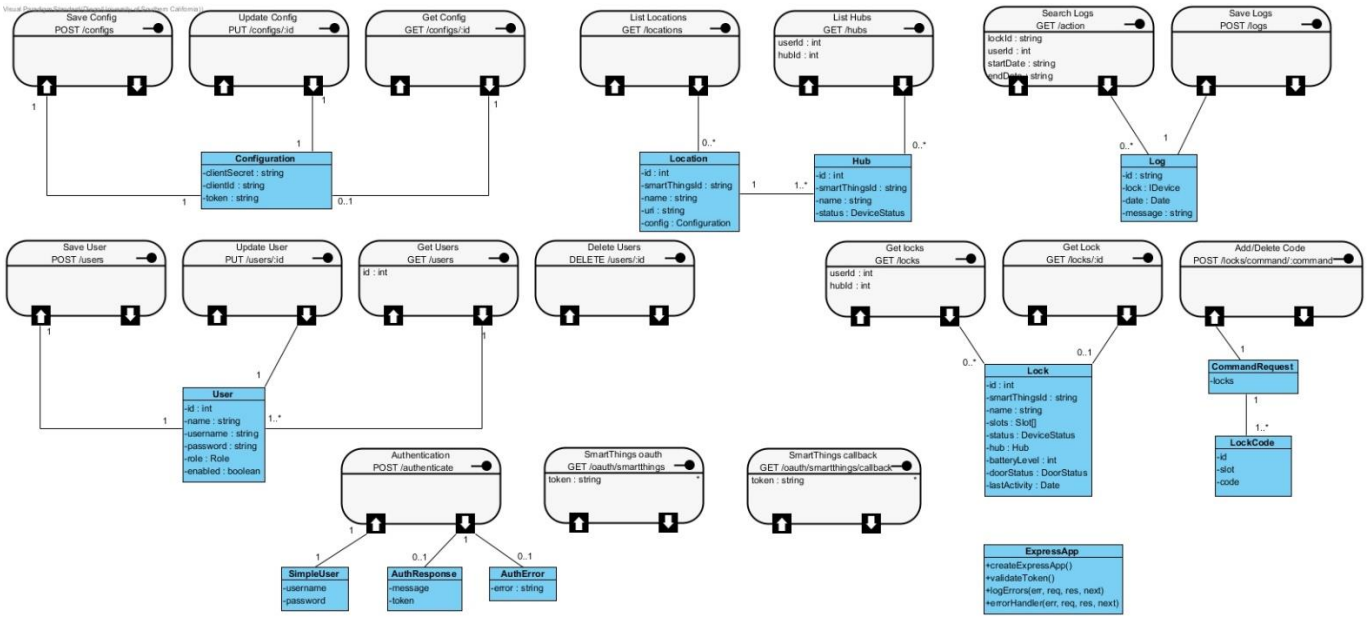


**Figure 11: Backend RESTful Services Class Diagram**

**Table 40: Backend RESTful Services Class Diagram Description**

| Class/Resource | Type | Description |
|---|---|---|
| POST /configs | REST Resource | It represents a service used to save a config to the database. |
| PUT /configs | REST Resource | It represents a service used to update a config to the database. |
| GET /configs/:id | REST Resource | It represents a service used to retrieve one configuration from the database. |
| Configuration | JSON/Model | It represents the JSON object expected/returned by the "/configs" service. |
| GET /locations | REST Resource | It represents a service used to retrieve a list of locations from the database. |
| Location | JSON/Model | It represents the JSON object returned by the "/locations" service. |
| GET /hubs | REST Resource | It represents a service used to retrieve a hub or a list of hubs from the database, based on the given parameters, i.e., (userId and hubId). |
| Hub | JSON/Model | It represents the JSON object returned by the "/hubs" service. |
| POST /users | REST Resource | It represents a service used to save a user to the database. |
| PUT / users | REST Resource | It represents a service used to update a user to the database. |
| GET / users /:id | REST Resource | It represents a service used to retrieve one user from the database. |
| GET / users | REST Resource | It represents a service used to retrieve a list of users from the database. |
| DELETE / users /:id | REST Resource | It represents a service used to delete one user from the database. |
| User | JSON/Model | It represents the JSON object expected/returned by the "/users" service. |
| POST /locks/command/:command | REST Resource | It represents a service used to add/delete/update the slots of a lock to the database, and also for sending the command to SmartThings to update the lock. |
| GET / locks /:id | REST Resource | It represents a service used to retrieve one lock from the database. |
| GET / locks | REST Resource | It represents a service used to retrieve a list of locks from the database, based on the userId and hubId if provided. |
| User | JSON/Model | It represents the JSON object returned by the "/locks" service. |

| | | |
|---|---|---|
| CommandRequest | JSON/ DTO | It represents the JSON file expected by POST “locks/command” service. |
| LockCode | JSON/ DTO | Simple JSON object used to send only the fields required by the service. |
| ExpressApp | Helper/Utils | Class used to start the express framework responsible for managing the routes and validate the requests |

### 4.1.2.6 SmartApp RESTful Services Class Diagram

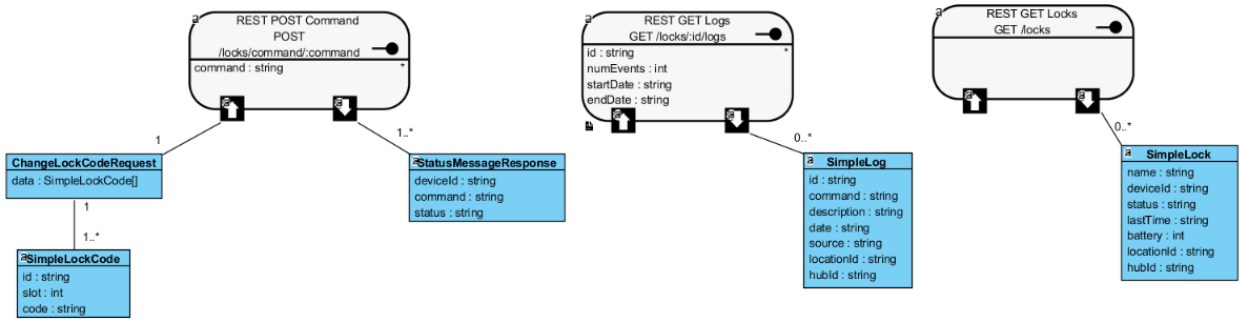


**Figure 12: SmartApp RESTful Services Class Diagram**

**Table 41: SmartApp RESTful Services Class Diagram Description**

| Class | Type | Description |
|---|---|---|
| REST POST Command | REST Resource | It represents a REST service used to post commands to the Locks. E.g. (setCode, deleteCode) |
| ChangeLockCodeRequest | JSON/Model | It represents the JSON format expected by the POST Command service. |
| SimpleLockCode | JSON/DTO | A SimpleLockCode represents the JSON attributes needed by the Command POST service. The id is a universal identifier of the Lock, and the slot is the slot that we want to update the code. |
| StatusMessageResponse | JSON/ DTO | It represents the JSON format returned by the POST Command service. |
| REST GET Logs | REST Resource | It represents a REST service used to get the events logged by the lock. |
| SimpleLog | JSON/ DTO | It represents the JSON file returned by the GET logs service. |
| REST GET Locks | REST Resource | It represents a REST service used to get locks available in the location. |
| SimpleLock | JSON/ DTO | It represents the JSON file returned by the GET Locks service. |

*DTO – Data Transfer Object

### 4.1.3 Process Realization

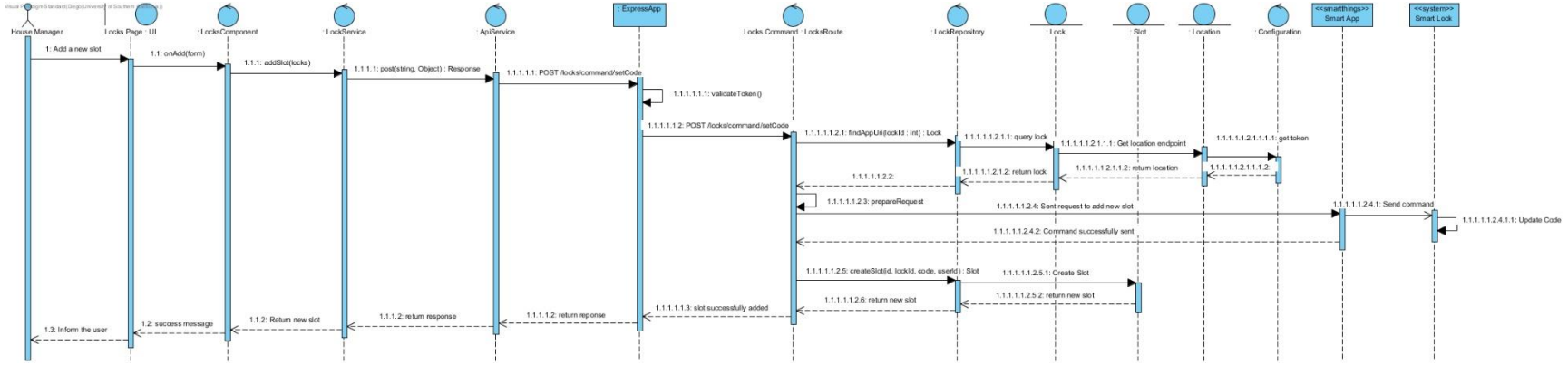


**Figure 13: Add Slot Code Sequence Diagram**

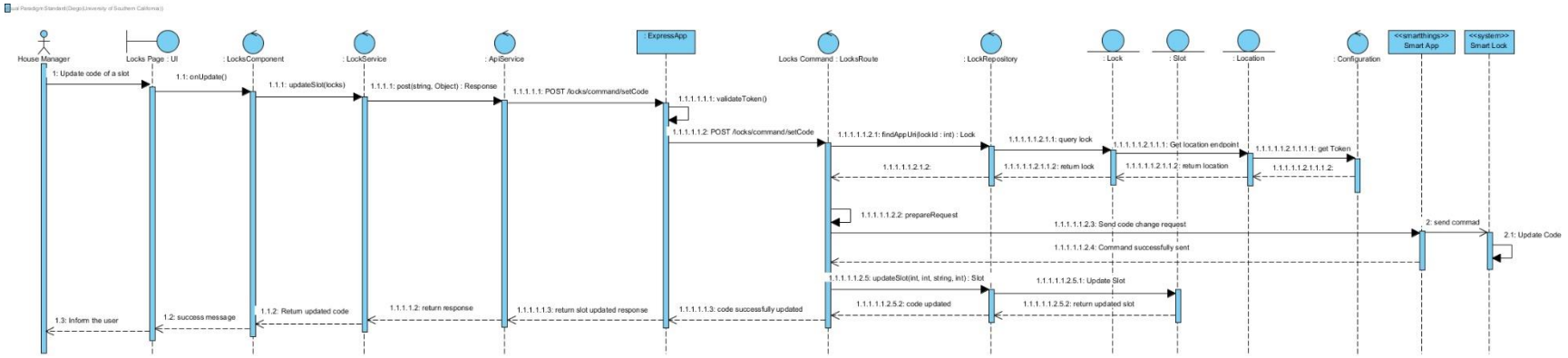


**Figure 14: Change Slot Code Sequence Diagram**

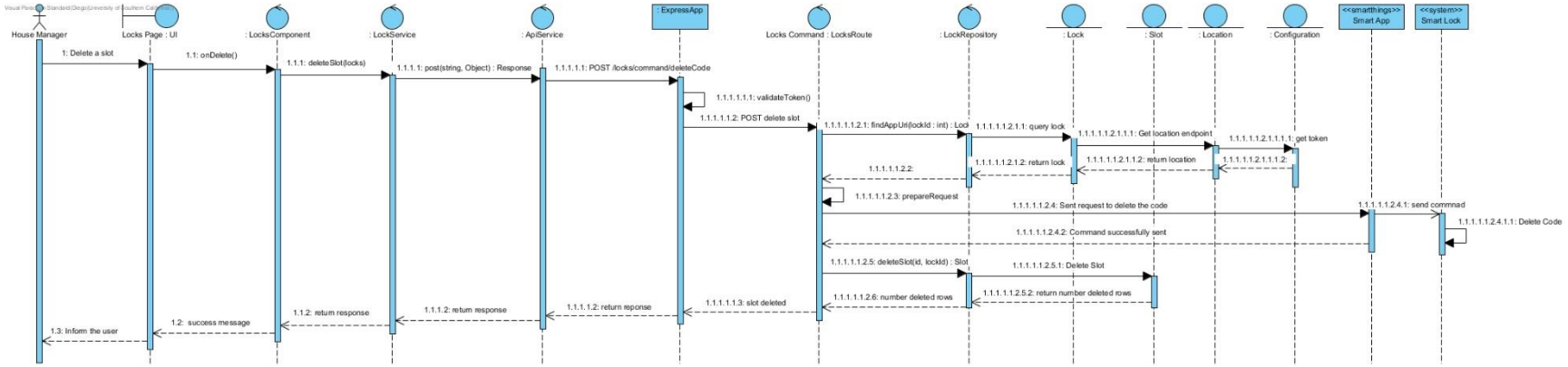


**Figure 15: Delete Slot Code Sequence Diagram**

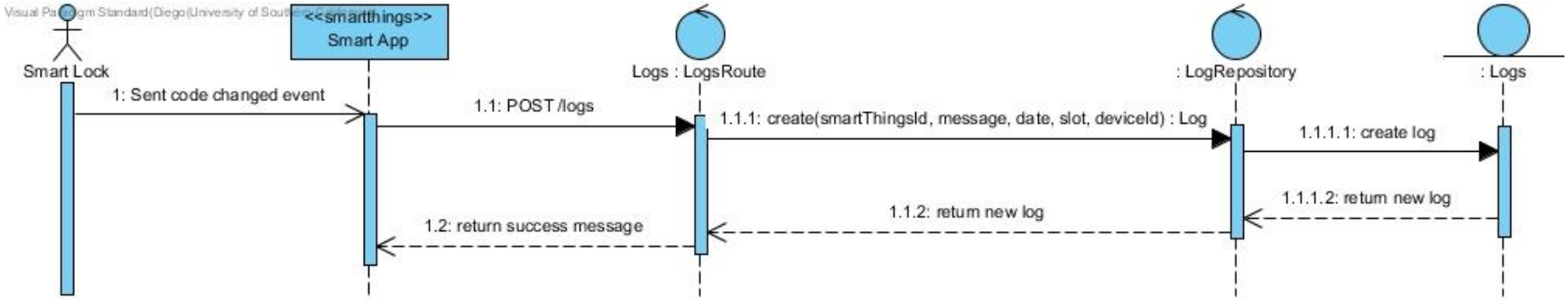


**Figure 16: Post Log Webhook Sequence Diagram**

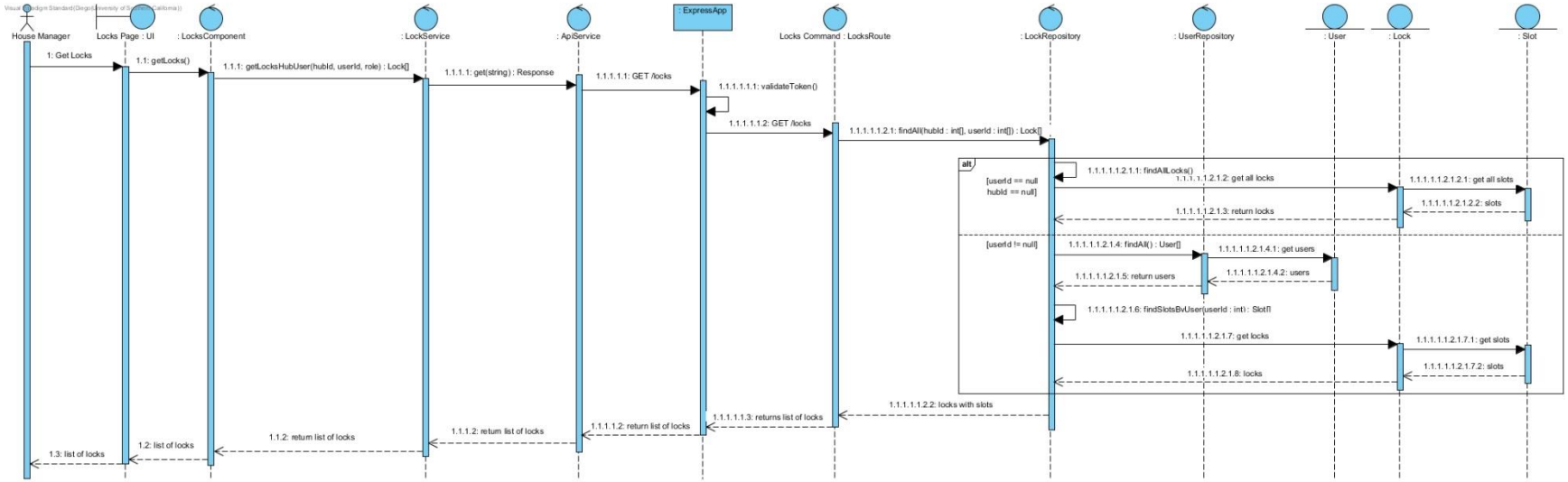


**Figure 17: Get Locks Sequence Diagram**

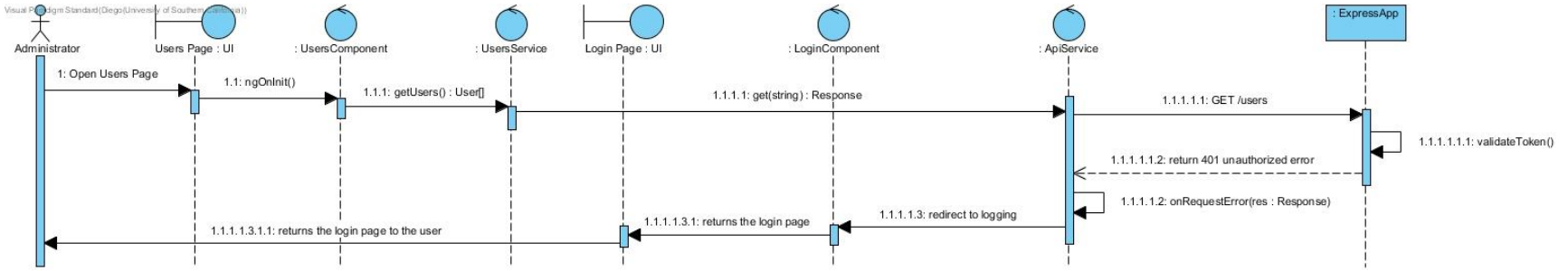


**Figure 18: Invalid Token Sequence Diagram**

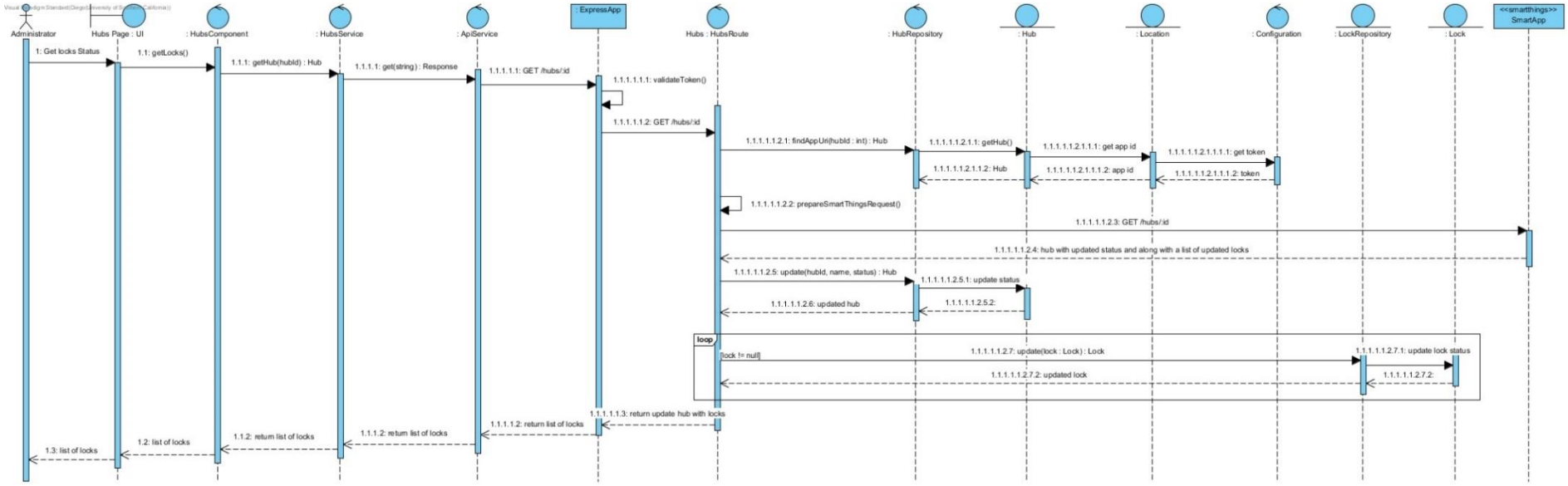


**Figure 19: Get updated hub status**

## 4.2 Design Rationale

We re-designed the whole application after the mid-term, due to scalability, security, and adaptability issues to name a few that our prototype had at that point in time. Therefore, several COTS have been replaced, e.g., Node.js instead of PHP, and MySQL instead of SQLite.

The main objective of the new design was to carefully isolate each layer of the application making it less dependent on each other, using as guidance basic software engineering principles, such as separation of concerns, modularity, abstraction, among others.

The result is a modular architecture, highly flexible and prepared to grow.

# 5. Architectural Styles, Patterns, and Frameworks

**Table 42: Architectural Styles, Patterns, and Frameworks**

| Name | Description | Benefits, Costs, and Limitations |
|---|---|---|
| 3-tier architecture | Separate the application into 3 tiers: the presentation tier using a web browser, a logic tier with Node.js server, and a data tier with MySQL. | • Using web browser allows the user to use any devices that they want, resulting in no need to maintain users' devices.<br>• Any change made in the application in the server will be automatically presented to the users.<br>• There is no cost for adopting the architecture.<br>• Need to acquire servers for application and database. Also, the database might be deployed on the same server if needed. |
| Repository | This design pattern is used to encapsulate the logic to access the database. | • "A Repository mediates between the domain and data mapping layers, acting like an in-memory domain object collection. Client objects construct query specifications declaratively and submit them to Repository for satisfaction." (Martin Fowler)<br>• "A Repository encapsulates the set of objects persisted in a data store and the operations performed over them, providing a more object-oriented view of the persistence layer." (Martin Fowler) |
| REST | The Smart App exposes RESTful services to access and perform an operation on the locks. | • "Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web." (Oracle Website) |
| Node.js | The backend of the application uses Node.js. | • Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side. It operates on a single thread, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching.<br>• built-in support for package management using NPM.<br>• Several free and open-source packages available. |
| AngularJS | AngularJS is a web application framework based on JavaScript. | • AngularJS provides an easy way to render user interface.<br>• Writing unit tests in AngularJS is easy<br>• It makes the code organized and easy to maintain. |