Smart Home System

Software Engineering Project Report 3

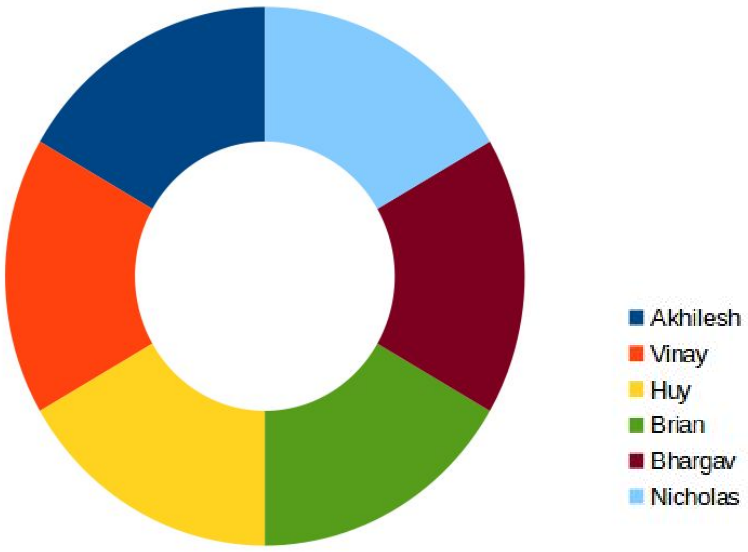
Group 12

05/01/2017

https://github.com/RUSEGroup12

Akhilesh Bondlela, Brian Ellsworth, Nicholas Grieco, Huy Phan, Vinay Shah, Bhargav Tarpara

Contribution Breakdown



Everyone contributed equally.

Table of Contents

Summary of Changes **5**

1. Customer Statement of Requirements (CSR) **6**

1.1 Problem Statement 6

2. Glossary of Terms **9**

3. System Requirements **10**

3.1 Enumerated Functional Requirements 10

3.2 Enumerated Nonfunctional Requirements 12

3.3 On-Screen Appearance Requirements 12

Project Management for Part 1 **13**

4. Functional Requirements **13**

4.1 Stakeholders 13

4.2 Actors and Goals 13

4.3 Use Cases 14

4.3.1 Causal Description 14

4.3.2 Use Case Diagrams 17

4.3.3 Traceability Matrix 19

4.3.4 Fully Dressed Descriptions 20

4.4 System Sequence Diagrams 25

5. User Effort Estimation using Use Case Points **30**

Project Management For Part 2 **31**

6. Domain Analysis **32**

6.1 Domain Models 32

6.1.1 Concept Definitions 35

6.1.2 Association Definitions 36

6.1.3 Attribute Definitions 38

6.1.4 Traceability Matrices 40

6.2 System Operation Contracts [5] 43

6.3 Mathematical Model 45

7. Interaction Diagrams **46**

8. Class Diagram and Interface Specification **53**

A. Class Diagrams 53

SES Class Diagram 53

SAS Class Diagram 54

SSS Class Diagram 55

B. Data Types and Operation Signatures 56

C. Traceability Matrices 59

SAS Traceability Matrix 59
SSS Traceability Matrix 60
SES Traceability Matrix 60
9. System Architecture and System Design 61
A. Architecture Style 61
B. Identifying Subsystems 61
C. Mapping Subsystems to Hardware 62
D. Persistent Data Storage 62
E. Network Protocol 62
F. Global Control Flow 63
G. Hardware Requirements 63
Project Management for Part 2 64
10. Algorithms and Data Structures 64
Algorithms 64
Smart Audio System: Algorithms 64
Smart Scheduling System: Algorithms 64
Smart Energy System: Algorithms 65
Data Structures 65
Smart Audio System: Data Structures 65
Smart Scheduling System: Data Structures 65
Smart Energy System: Data Structures 65
11. User Interface Design and Implementation 66
11.1 Preliminary Design 66
11.2 Secondary Design 69
12. Design of Tests 71
Casual Test Cases 71
Descriptive Test Cases 73
Integration Testing 78
13. History of Work, Current Status, and Future work 78
A. Merging the Contributions from Individual Team Members 78
B. Project Coordination and Progress Report 79
C. Plan of Work 80
D. Breakdown of Responsibilities 81
14. References 82

Summary of Changes

- Word formatting was made more consistent.
- Customer statement of requirements was clarified and categorized.
- In the actors and goals part, more detail description may be needed for different roles
- Formatted the use case diagrams for easy readability
- Traceability matrix may be drawn for different roles

- Added outlines of which network protocols were used and how they were implemented in each of the various subsystems. Also an outline of the global flow of the system as the data passes through the system.
- Sensor information will be simulated through the unity simulation. Realistically to implement this we would use an arduino to interface with a relay and a light.

- Added use case descriptions
- Clarified diagrams and improved readability.
- Defined actor to the reference of each individual use case, where it is necessary

- Parameter values should not be shown in the class diagrams as professor stated in his requirements. Below the diagrams are fully dressed functions with parameters and proper return types
-
- The description of the class should be more precise, especially the description for some important methods, it would be better.

- Network Protocol; hope the description of it can include more detail and how you use them.
- DB table design is not complete, you may show an example of your data storage.
- The hardware requirement were made more specific
- Improved Android interface design
- Detailed pass/fail criteria were provided
- References are numbered in the report, making it easier to find where they were cited.
- Improved time approximation for Smart Schedule System.
- A user login screen was incorporated.
- Schedule, Audio, and Energy Systems were integrated

1. Customer Statement of Requirements (CSR)

1.1 Problem Statement

Our Pitch

Are you tired of having a fragmented home environment, fuddling through apps to control various devices? Well, you have come to the right place; we are developing a new unified software environment that will make your daily routines a little less cumbersome. Our Smart Home System integrates your alarm, lights, utilities, and speakers into one neat and easy to use app.

You wake up early in the morning to commute to work or school but always seem to get stuck in traffic or delays and end up being late. Your time is too valuable to be spent sitting in traffic. Now we present you, with the option to beat unexpected delays with our Smart Schedule System. Before you wake up, the Smart Alarm will look up traffic and delay conditions on your commute and wake you up earlier to make sure you make it on time for that meeting with your bosses or the final exam you studied all week for. After you come from all long days work in the cold, you want your home to be *extra* warm. Instead, you are hit by a cold blast of air from your archaic heating system. If you had our Smart Energy System, your home would think for itself. Your home can be an igloo when you are not there but make it extra toasty just before you arrive at your kingdom. But wait it gets a better, we also provide an innovative way to listen to your music library. You are listening to your favorite tunes and want go to the kitchen to get the snack; you would have to pause the song, get the snack, and resume. With our intelligent tracking, the music follows you as you move from one room to another. These are some of the many ways our Smart Home System can make your life a little more easier.

Current home automation utilities are available from different vendors. This means that you have to pay a lot of money for each individual system and waste time configuring your specifications. After that, you have to use a set of apps on your smartphone to change any settings or trigger events that you would like to occur in the future. If you want your lights to be on at certain times, you need to buy special connected light bulbs, which cost hundreds of dollars each. This is only to have lighting in your home. Moving on to having your home at a comfortable temperature; you need to buy another system which connects to your HVAC system or thermostat to regulate the temperature of the home according to your habits and liking. Now that the home is lit and heated, you want to enjoy some music on your surround sound speaker system; however, you have to play it in a single room and is confined to that room. If you leave, you have to either pause it or miss a potion. To have a speaker system that is synchronized throughout your home, would require several speakers and all of them would have to be from the same manufacturer to make sure they can be interconnected. The price of automating your lights, HVAC, and sound system is now easily several thousand dollars. To earn this money, you have to wake up early and go to work. However, when your on your way to work, you may run into unexpected traffic, which eventually causes you to be late to work. In your rush to get ready and get to work on time, you get preoccupied in other things and may forget some of the important tasks that you had to accomplish during the day. Now you'd definitely wish you had our system.

Our Smart Home System is a viable and cost effective way to automate your home at a fraction of the cost as the individual corporations currently in the market. We blend all of the features into a single environment, which can be accessed from your smartphone. The problems many people have with their disjointed smart ecosystems are the clunky interfaces that come along with each and every smart device. You have one app for your smart lights, another one for your climate control; the list is neverending. And worse, you may have to create a new user account for each and every device with a password. With our unified ecosystem those days are over. We will have a simple GUI app that will include a dashboard where you can control all your smart systems. It contains a section for the Smart Schedule System, Smart Audio System, and the Smart Energy System. Each section will have its own settings with an intuitive navigation control. Functionality and ease of use is our number one priority, not flashy cheap looking interface.

Our system will use a wide range of sensors and radios. The Smart Schedule System, Smart Audio System, and Smart Energy System will all connect to your Local Area Network(LAN) along with the Wide Area Network(WAN). The physical connections between the systems and your local switch(or wireless router) can occur with preferably a gigabit ethernet connection or at minimum 802.11n [1] access point. This allows for all of the components to be connected wirelessly and avoid the hassle of tangled wires.

The Smart Schedule System will connect to the Wide Area Network(Internet or Intranet) to get traffic updates. The Smart Audio System will connect to Amazon's Voice Assistant, Alexa[2] , and other Amazon [2] services. And all three systems will communicate with the app on the user's smartphone to get updates or changes directly from the user.

Waste is not good for the environment, the economy, and one's bank account. At home we waste a lot for our convenience -- sometimes knowing or unknowingly. For example, leaving lights on in empty rooms and cooling/heating on when we are not home. The current lighting system in the average household is energy inefficient. We must walk over to turn on and off lights at a physical switch. When we are ready for bed we must turn the lights off and go unnaturally from a illuminated room to darkness creating risk of injury and inconvenience. Additionally, leaving heating and lights on wastes energy meaning it costs you more, and if the environment is one's concern, more coal/oil is wasted in generating that electricity to power heating/cooling/lighting. With the advent of sensors, algorithms, and the internet this waste can be eliminated. Why can't the system know when and where the resident needs lighting, cooling/heating? This is because the current systems are all manual and the new smart systems do not share information with other hardware and services.

At the end of your day, this system will inform you when you should go to bed based on how much sleep you would like to get. This will prepare you optimally for waking up the next day. Now that you have seen what our smart system can do for you; we hope you consider it the next time you are stuck in traffic, listening to music, or wishing your utility bill was lower.

Smart Audio System Goals and Description

To make sure you don't miss a beat, the Smart Audio System will automate your music system throughout your home. When you play music, the system will use the position of your smartphone to locate where you are in the house and play your music in your vicinity. That means, if you're playing music in your room and suddenly walk to the kitchen, you don't have to pause the music. It will follow you and provide the surround sound experience wherever you are in the home.

It's easy to adjust any of the playback settings, such as volume or song, because of the Amazon's Voice Assistant, Alexa. With Alexa integrated in the system, you can speak to the system and it will adjust accordingly. This is efficient because, you don't have to preoccupied with managing the settings and can work on being more productive in their daily lives.

To make sure that your are the most productive, the Smart Audio System will read your schedule aloud to you when you wake up. This way, you know what needs to get done and can prepare for the day ahead in a more efficient manner.

Smart Scheduling System Goals and Description

The Smart Scheduling System will ensure that you are always punctual to your events. The system will initiate a wake-up notification before your first event in the morning. This notification is processed by the Smart Audio System so that you can be readily awoken and informed about your day. The wake-up notification is dynamically scheduled based on potential delays so that you always have plenty of time to prepare for your day and get to your destination on time. This system takes numerous factors into account, including travel distance, real-time traffic patterns, and weather forecasts.

The Smart Scheduling System allows you to add events to your daily calendar and will automatically handle timing conflicts for you. It will optimally prioritize the order of your events and will even inform you when you should prepare to leave for your next event. The scheduler is always keeping track of how long it will take to reach your next event so that you never have to worry about being late.

Smart Energy System Goals and Description

This system's purpose is to control and optimize one's hvac and lighting system. In particular from the application one should be able to control one's home energy systems: hvac and lighting. The home temperature should be connected to and controlled by the application.Using machine learning the system should learn the preferences and time it takes to change the temperature in the house. One should also be able to control the home lights with application.

2. Glossary of Terms

HVAC — Heating, Ventilation, and Air Conditioning Unit.

GUI — Graphical User Interface

Smart Audio System (SAS) — A system which will allow the user to control music playback through a mobile application or voice assistant.

Smart Energy System (SES) — A system in which temperature and lighting are controlled based on user preferences and behaviors so that there is minimal to no energy waste.

Smart Home System (SHS) — The combination and intercommunication of the SAS, SES, and SSS.

Smart Schedule System (SSS) — An application that will inform the user when to go to bed and when to wake up.

Time-to-temperature — Time required to reach a specific home temperature based on current temperature

Time-to-home — Users estimate of how much time it will take them to arrive home

User — The individual using the SHS who is living in the home.

3. System Requirements

3.1 Enumerated Functional Requirements

Req-x	Priority Weight	Description
REQ-1	5	Scheduler should provide the user with a wake-up notification that allows the user enough time to travel to their destination.
REQ-2	4	Adjust wake-up notification time based on weather forecast and historic weather conditions.
REQ-3	4	Adjust wake-up notification time based on traffic delays along commute route.
REQ-4	1	**[FROZEN]** Adjust wake-up notification time based on the user's sleep phase
REQ-5	4	Set a bedtime notification for the user based on desired sleep duration
REQ-6	3	Create a schedule based on time ranges, durations, and event priorities
REQ-7	3	Notify the user of if an attempt is made to create an event that causes a timing conflict
REQ-8	4	Adjust heating, cooling, and lighting system when the user is at home and not at home for maximum cost efficiency
REQ-9	4	Heating and cooling system are adjusted for maximum comfort just before the user arrives
REQ-10	5	SAS should read aloud notifications from the SSS
REQ-11	4	Amazon Alexa integration into SAS to read todo list, play music, etc.
REQ-12	1	**[FROZEN]** Allow user to intercom from the phone to any room in the house.
REQ-13	2	Handle music conflicts on SAS by defaulting to master user when multiple users try to play music in the same room
REQ-14	5	Playing audio on multiple speakers in different locations at once

REQ-15	3	Follow the user around the home with audio playback
REQ-16	5	System predicts time required to reach a specific temperature by recording previous data and learning the temperate habits of user
REQ-17	3	System shall detect when residents are only on one floor of a dual zone heating/cooling system and will adjust the zone which is not being used
REQ-18	3	User should be able to notify the system that they are moving to another zone in their house by using the application so the system can begin adjusting the temperature before their arrival
REQ-19	3	Machine learning linear regression shall predict when the user has awoken from sleep so the temperature can be adjusted
REQ-20	3	PIR sensors will detect when the user has awoken from sleep so the lighting can be adjusted accordingly based on user preferences
REQ-21	4	The system shall detect which areas are not being occupied by using passive infrared sensors so the lighting shut off when motion is not detected
REQ-22	3	The system shall adjust temperature around sleeping hours based on users feedback for energy efficiency and comfort
REQ-23	5	System adjusts temperature before arrival to the home based on user's input and approximate time until they are home
REQ-24	5	System adjusts temperature and lighting after departure from the home either after a specific time period of inactivity or input from the user application
REQ-25	3	Systems displays energy savings
REQ-26	5	Admin User Access to System Shell

3.2 Enumerated Nonfunctional Requirements

Req-x	Priority Weight	Description
REQ-27	5	Installation Documentation
REQ-28	4	Simple GUI supported on wide range of platforms
REQ-29	5	Properly Grounded Circuit(for safety during power loss)
REQ-30	4	Loss of power and/or connection, results in manual control of utilities(lights & HVAC)
REQ-31	3	Fast Ethernet and 802.11n [1] (Gigabit and ac preferred respectively)
REQ-32	3	Plug-and-play and retrofitted
REQ-33	3	Snooze feature for wake-up notification
REQ-34	2	Ability to turn on/off Alexa
REQ-35	1	Legal
REQ-36	1	Advertisment
REQ-37	2	The user is able to view the factors that lead the SSS to produce a given schedule: weather, traffic, and travel duration.

3.3 On-Screen Appearance Requirements

Req-x	Priority Weight	Description
OSR-1	5	Dashboard with tabbed section for each smart system
OSR-2	5	Settings menu inside of each section for each systems
OSR-3	4	Simple toggle switches to manage settings
OSR-4	3	Easy to use interface with reduced number of clicks
OSR-5	1	Splash screen to introduce the app upon open
OSR-6	2	Tutorial to help the user become oriented with the app

Project Management for Part 1

The first two meetings we had an agenda that we needed to decide the project cluster and then actual ideas. The other thing was that we set expectations of what each person in the group is responsible for in general. Next we had to narrow a project choice down. In the end we switched our initial idea traffic monitoring to home automation because we had a generally good idea with it and we all liked the idea. The last attribute that the group did was break into subgroups based on subproject.

4. Functional Requirements

4.1 Stakeholders

Investors: Selling smart home system bundled with homes and/or providing smart home services with a markup can generate a profit.

Super Commuters: People that have long commutes to work would want to utilize our system for its practicality of maximizing efficiency of sleep, dealing with delays, and keeping tracks of daily requirements.

Residents (General): People who like listening to music at home, having their home temperature and lighting save energy while maximizing comfort, and never missing in their scheduled day will greatly benefit from our SHS.

4.2 Actors and Goals

End-User — Initiating: The user will control various smart home functions and customize their settings. The user can customize the alarm to act appropriately to delays, adjust settings on his climate control system, add/remove items from his todo lists, control music being played, and more.

Communication Medium — Participating: Medium to relay information from one system to another(e.g. Wires, WiFi, Bluetooth, etc.)

Indoor Positioning System Client — Initiating: Pulls Wi-Fi signal strength from the user's network interface card, opens socket, and sends signal strength value over it.

Indoor Positioning System Server — Listening: Listens for connections from client, matches signal strength to a location, and changes output to respective speakers.

Data Collection Sensor Controllers — Initiating: As the smart home systems learns user preferences it will adjust home settings by itself. As controllers for various smart subsystems detect changes of interest from its sensors; it will take action to start a defined procedure. Such examples include, the SAS detecting a change in user location and will change where music is being played, HVAC detects changes in time and appropriately changes temperature, and etc.

System Database — Participating: SHS keeps tracks of different logs from each subsystem, so the user can track down possible issues that may or may not occur.

System Shell — Participating: Monitoring what is happening in real-time on SHS and subsystems for troubleshooting purposes.

Amazon Alexa — Initiating/Participating: Voice commands to Alexa can be used to interact with various Amazon services. Alexa is also equipped with automated services which can remind you of tasks when the time is right, communicate with other devices, and play music.

Unity Simulation — Participating: Demos most functionality of the SHS. This will present a virtual home and users making use of our smart features.

4.3 Use Cases

4.3.1 Causal Description

Use Case - 1: Todo List [REQ-11]
At a specified time the SAS will play an automated todo list indicating the user's planned schedule throughout the day. It should play wherever the user is located.

Use Case - 2: Audio Room Selection [REQ-14]
User should be able to specify where his/her audio will be played.

Use Case - 3: Audio Selection Conflict Control [REQ-13]
Multiple user's should not conflict in audio room selection. When a user requests audio being played in one room where music is already playing, the system should default to the primary user's audio.

Use Case - 4: User Audio Tracking [REQ-15]
When a user is playing music in his/her room, the music should automatically detect and track users as they walk to other rooms(given no conflicts).

Use Case - 5: Amazon Alexa Integration [REQ-11]
The user can access all of Amazon Voice Assistant Services from the SAS. The functionality of Amazon's voice assistant is baked into the SAS.

Use Case - 6: Speakers for SSS [REQ-10]
The user can hear notifications from the SSS on the SAS.

Use Case - 7: Monitoring [REQ-26]
Administrative users has access to the SHS shell for troubleshooting purposes in the case something goes wrong with his/her system. Higher privileged users can look at backup logs of the whole system when they are troubleshooting.

Use Case - 8: Temperature Ready [REQ-16, 23]
User is returning home from work/errands and checks application for the time-to-temperature prediction and initiates heating/cooling when time-to-home is approximately the same.

Use Case - 9: Dual Zone Inactivity [REQ-17, 21]
No movement has been detected for a certain duration of time in one or both of the zones in a dual heating/cooling home and the temperature adjusts to minimize energy consumption and lights are shut off in the respective zone

Use Case - 10: Activate Dual Zone [REQ-18]
Users notifies system to adjust lighting and temperature in a zone they are moving to preferred settings

Use Case - 11: Sensor Activation [REQ-21]
User's enter a room and lights are turned on if it is dark (low luminosity)

Use Case - 12: Wake Up Adjustments [REQ-19, 20]
Temperature adjusts to user's temperature preference slightly before the time they normally wake up. Lights turn on gradually when motion is detected.

Use Case - 13: Going to Bed Adjustments [REQ-22]
Temperature adjusts to energy efficient temperature around normal sleep hours. Lights turn off when user notifies the system through the application

Use Case - 14: Empty Home [REQ-24]
User notifies the system they are leaving their home. Lights are shut off and temperature adjusts to minimize energy consumption

Use Case - 15: Energy savings [REQ-25]
User checks their energy savings for a given time duration

Use Case - 16: Schedule Initialization [REQ-6, REQ-7]
The user has just downloaded the app and must enter his schedule, work location, and home location when the app is opened for the first time.

Use Case - 17: Traffic Delay Factors [REQ-37]
User can view weather condition, traffic on his route, time taken to his next destination when he opens the app.

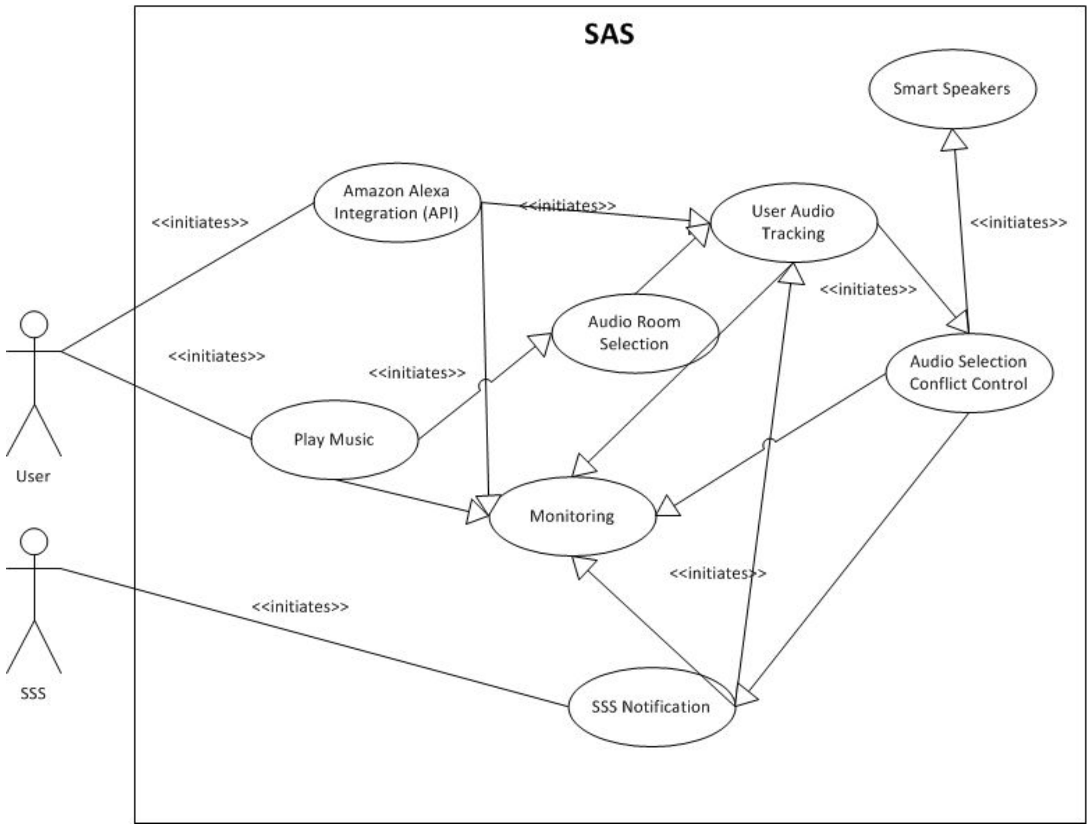
Use Case - 18: Schedule Modification [REQ-11]
User can modify his schedule, destination, add holiday, change alarm sound.

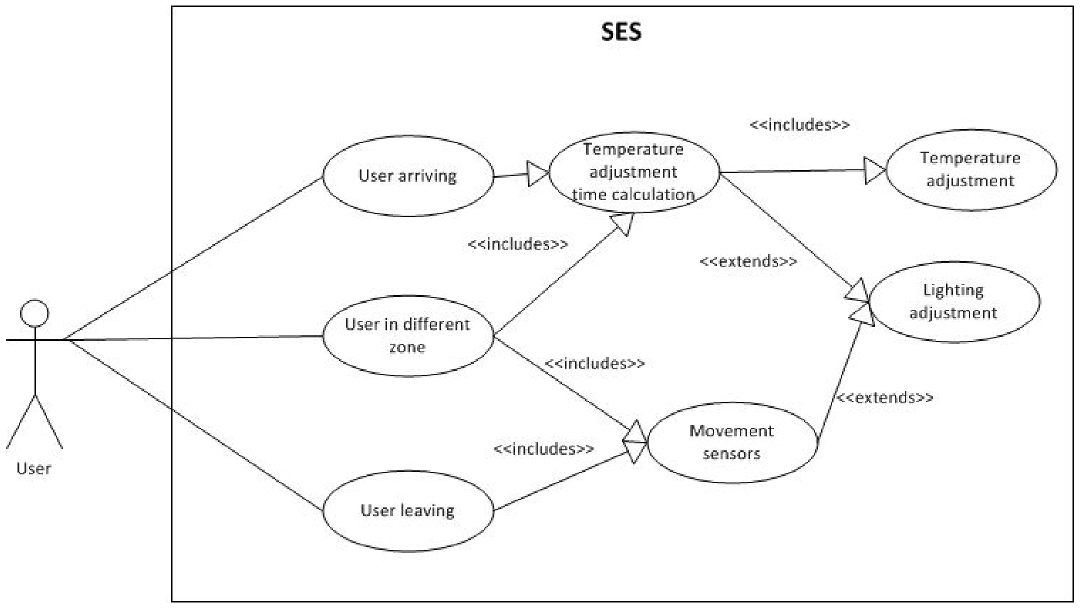
Use Case - 19: Dynamic Wake-up Notification [REQ-1, 2, 3]
The user would like to go to bed without having to worry about being late for work. He knows that he must wake up when the alarm goes off and that if he does not, then he will be late. A wake-up notification will be set for the user, accounting for traffic conditions.

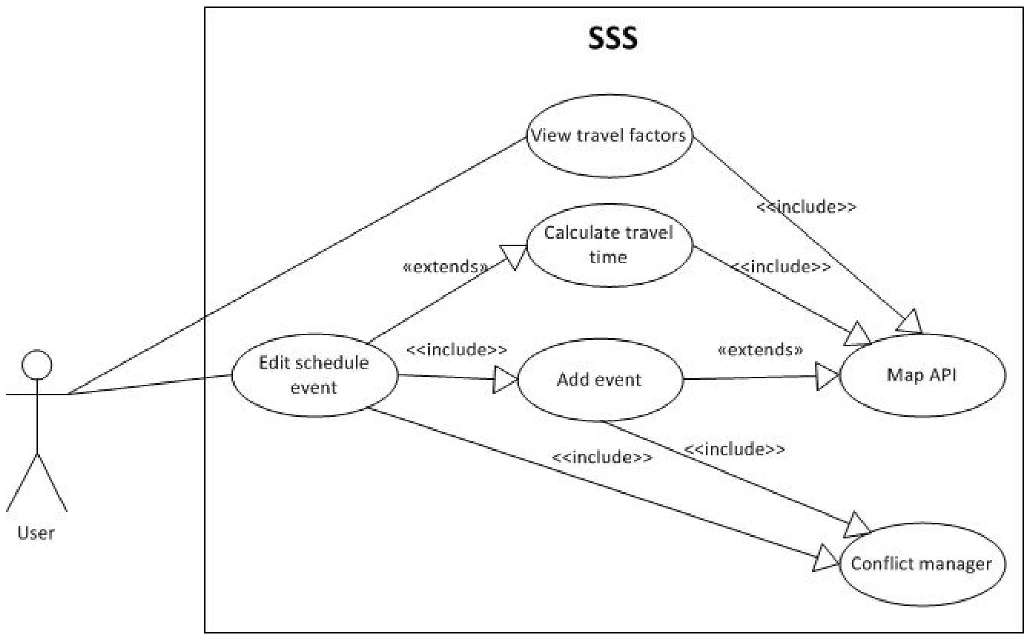
Use Case-20: Estimated Time of Arrival [REQ-6, 7]
The user would like to put events in his schedule. If the event has a destination, then the SSS will tell the user the optimal time to leave for the event.

Use Case-21: Bedtime Notification [REQ-5]
The user has entered his daily activities, but would like to know when to go to bed in order to wake up the next morning after a specified duration, so the SSS will tell the user when he should go to sleep.

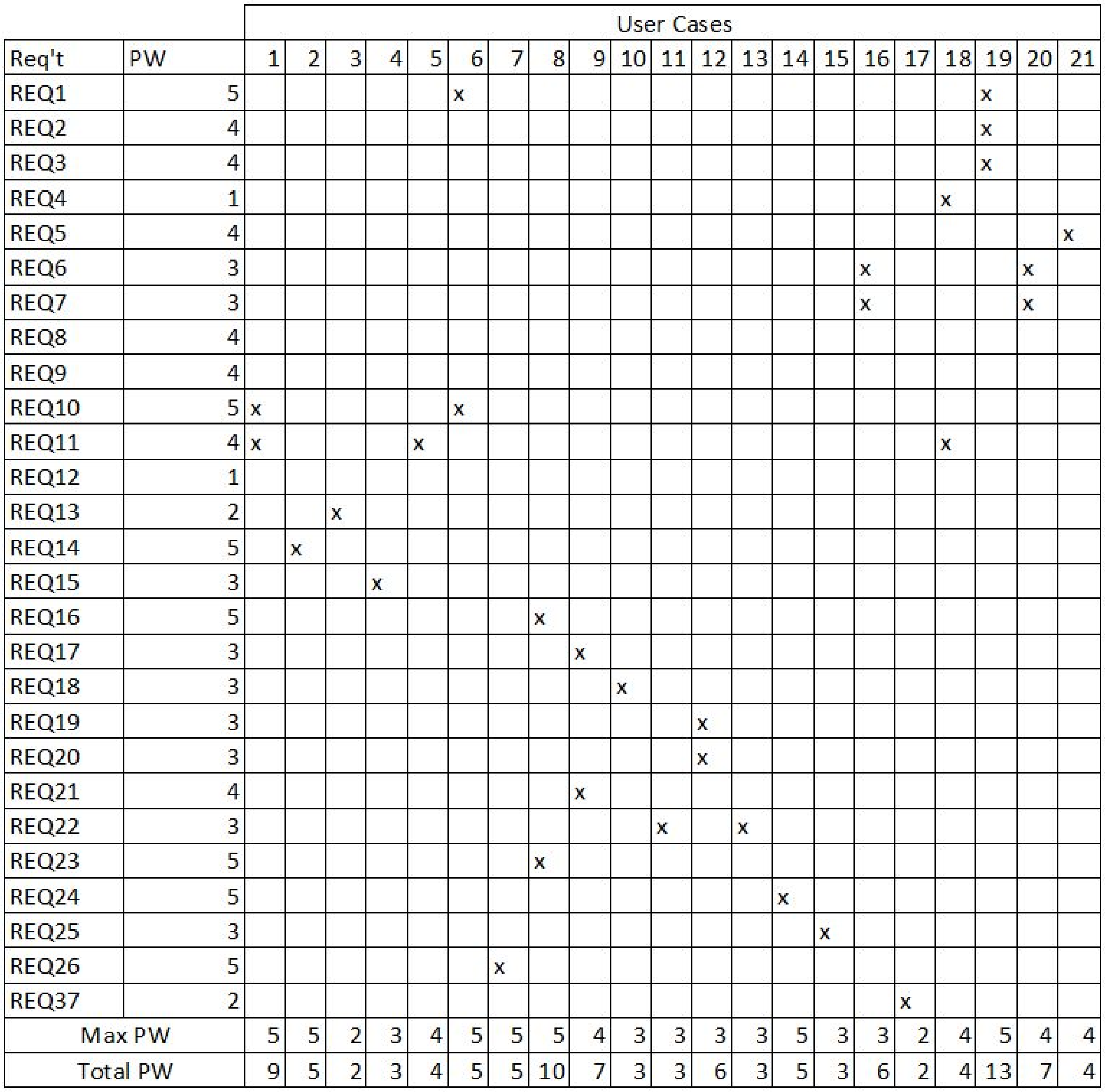
4.3.2 Use Case Diagrams






SSS
View travel factors
<<include>>
Calculate travel time
«extends»
<<include>>
Edit schedule event
<<include>>
Add event
«extends»
Map API
<<include>>
<<include>>
Conflict manager
User

4.3.3 Traceability Matrix



		User Cases																				
Req't	PW	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
REQ1	5						x													x		
REQ2	4																			x		
REQ3	4																			x		
REQ4	1																		x			
REQ5	4																					x
REQ6	3																x				x	
REQ7	3																x				x	
REQ8	4																					
REQ9	4																					
REQ10	5	x					x															
REQ11	4	x				x													x			
REQ12	1																					
REQ13	2			x																		
REQ14	5		x																			
REQ15	3				x																	
REQ16	5								x													
REQ17	3									x												
REQ18	3										x											
REQ19	3												x									
REQ20	3												x									
REQ21	4									x												
REQ22	3											x		x								
REQ23	5								x													
REQ24	5														x							
REQ25	3															x						
REQ26	5							x														
REQ37	2																	x				
Max PW		5	5	2	3	4	5	5	5	4	3	3	3	3	5	3	3	2	4	5	4	4
Total PW		9	5	2	3	4	5	5	10	7	3	3	6	3	5	3	6	2	4	13	7	4

4.3.4 Fully Dressed Descriptions

The smart audio system's integration with Amazon Alexa is a core design function. It allows expandability of the audio system's function as new services and features are added to Alexa. The user will be able to use Alexa to handle various commands to control the SSS. When the user wakes up, it can play his/her todo list, play music wherever he/she is and more. Alexa API will be used to access the service and will be outputted to the speakers nearest to the user.

Use Case 5	Amazon Alexa Integration
Related Requirements	REQ 10 & REQ 11
Initiating Actor	SAS
Actor's Goal	To notify the user of the planned schedule for the day ahead.
Participating Actors	Amazon Assistant Alexa
Preconditions	Internet Access Amazon Alexa Integration to the SAS Network Access
Post Conditions	SAS gives up use of speakers

Flow of Events for Main Success Scenario
1. User Gives Alexa a Voice Command on App 2. Voice Command Uses API to Access Amazon Services 3. SAS returns Response from Amazon 4. SAS pulls location information of user 5. SAS plays response on nearest speakers 6. Alexa Proceeds to Commit Action(some commands)

Notifications sent from the SSS will be handled by the SAS. When the user sets an alert or an alarm on the scheduler all the information from the scheduler will have to be relayed to the SAS. All this information is transmitted across the local network so it will be important to have a good backbone. We recommend using 802.11n or AC but Gigabit Ethernet is preferred. Same as with the other functions as well, notifications will be played at the location of the user. So only specific user notification will delivered to the user.

Use Case 6	Speakers for SSS
Related Requirements	REQ 10 & REQ 1
Initiating Actor	SSS
Actor’s Goal	To notify user with audio message relayed through the SAS
Participating Actors	SSS, SAS
Preconditions	SAS can communicate with SSS SSS can communicate with SAS
Post Conditions	SAS gives up use of speakers

Flow of Events for Main Success Scenario
1. A Reminder or Alarm Goes off on SSS 2. SSS signals SAS of the notification 3. SAS signals SSS that the notification was received 4. SAS pulls location information of user 5. SAS plays notification on nearest speakers

Use Case 8	Temperature Ready
Related Requirements	REQ 16,23
Initiating Actor	User of the application, resident of the house
Actor’s Goal	To notify the system that they are arriving home so that the desired temperature could be met by the time they are home
Participating Actors	Thermostat, thermometer, server, application, timer
Preconditions	SES is live and working, no one is at home
Post Conditions	SES watches for movement

Flow of Events for Main Success Scenario
1. User will open the mobile or web application 2. User will input eta or give an eta based on Google Maps [3] 3. System will calculate the time to start adjusting the temperature 4. System will start a clock and make the change at the appropriate time

Use Case 9	Dual Zone Inactivity
Related Requirements	REQ 12, 21
Initiating Actor	Sensors
Actor's Goal	To sense the lack of movement in the a zone of the house
Participating Actors	Residents, pets, sensors, thermostat
Preconditions	A dual or tri zone hvac system, lack of movement
Post Conditions	Temperature adjustment

Flow of Events for Main Success Scenario
1. User moves from one zone to another 2. A certain amount of time, x has passed since movement has been seen in the other zone 3. System adjusts heating/cooling in that zone 4. User can reactivate zone by telling app

Use Case 10	Schedule Event Addition
Related Requirements	REQ-6, REQ-7
Initiating Actor	User
Actor’s Goal	To add an event to his schedule
Participating Actors	SAS, Google Maps [3]
Preconditions	App installed properly SSS able to send signals to the SAS
Post Conditions	Updated schedule. Updated wake-up notification time. Updated departure times.

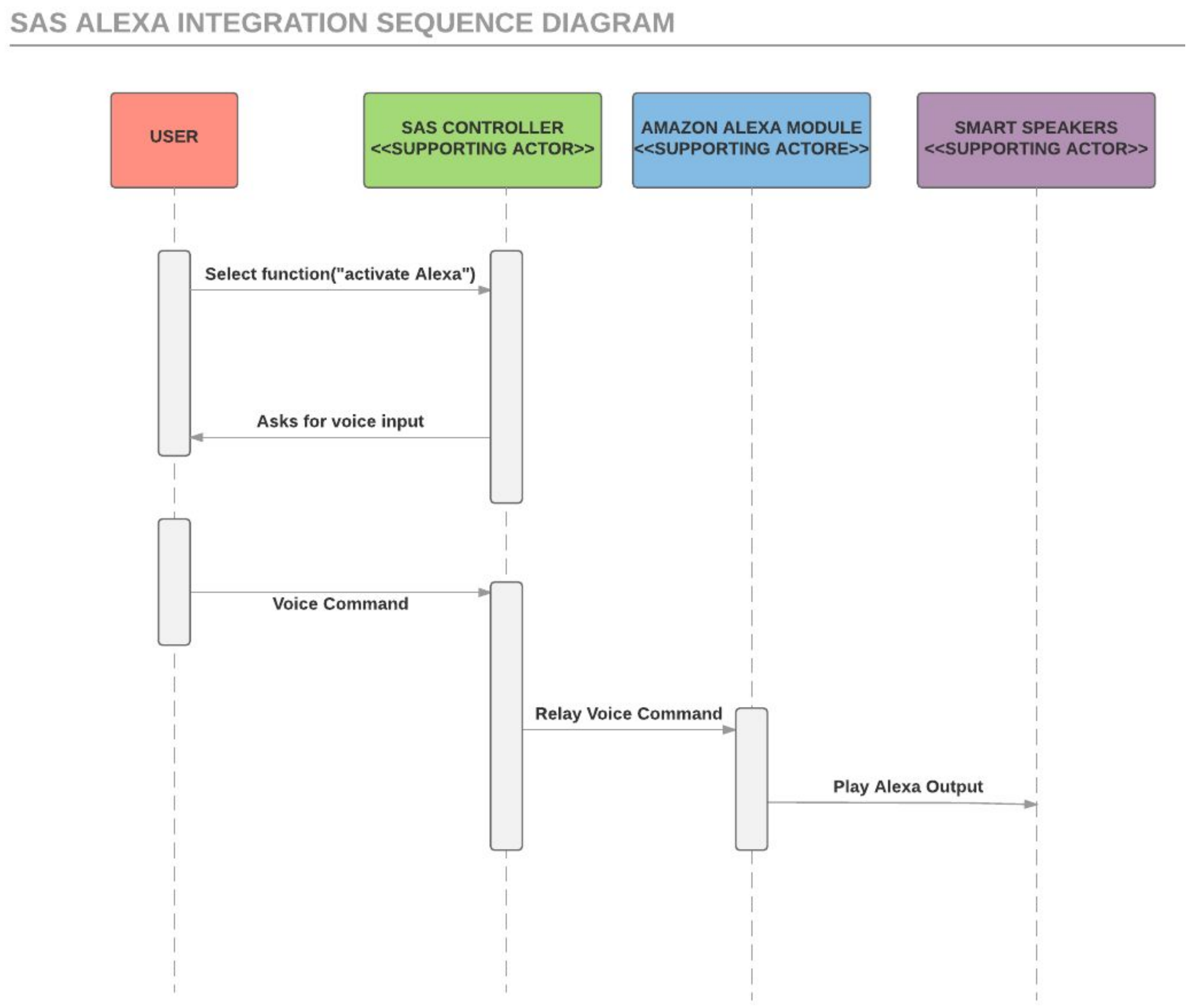
Flow of Events for Main Success Scenario
The user opens the SHS application. The user selects the SSS button from the sidebar The user inputs relevant event data: name, location, time, duration, deadline The SSS checks for a timing conflict with a preexisting event. The SSS requests travel route and duration to the event destination from Google Maps. The SSS checks for a timing conflict with a pre existing event based on the travel duration. The SSS sends a notification to the user indicating success or failure of event creation.

Use Case 17	Checking relevant information about traffic
Related Requirements	REQ-37
Initiating Actor	User
Actor’s Goal	To check time taken to next destination, weather condition, alternative routes
Participating Actors	Google Maps [3], Weather API[18]
Preconditions	Internet Access, GPS
Post Conditions	Traffic and weather conditions updated successfully

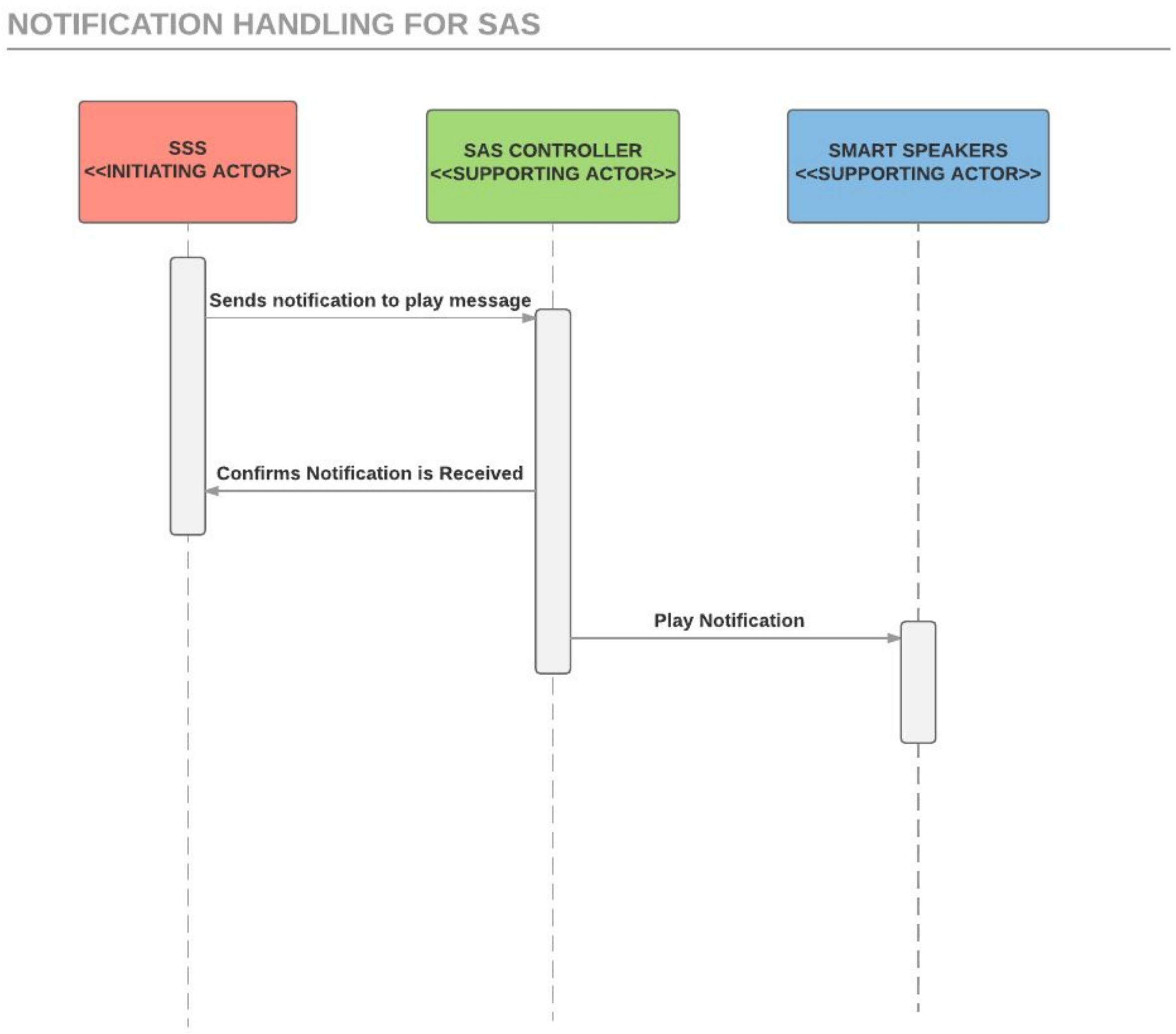
Flow of Events for Main Success Scenario
1. The user opens the SHS application. 2. The user selects the SSS button from the sidebar 3. The user chose the Next Destination Tab 4. The SSS display relevant information about traffic and weather to the next event

4.4 System Sequence Diagrams

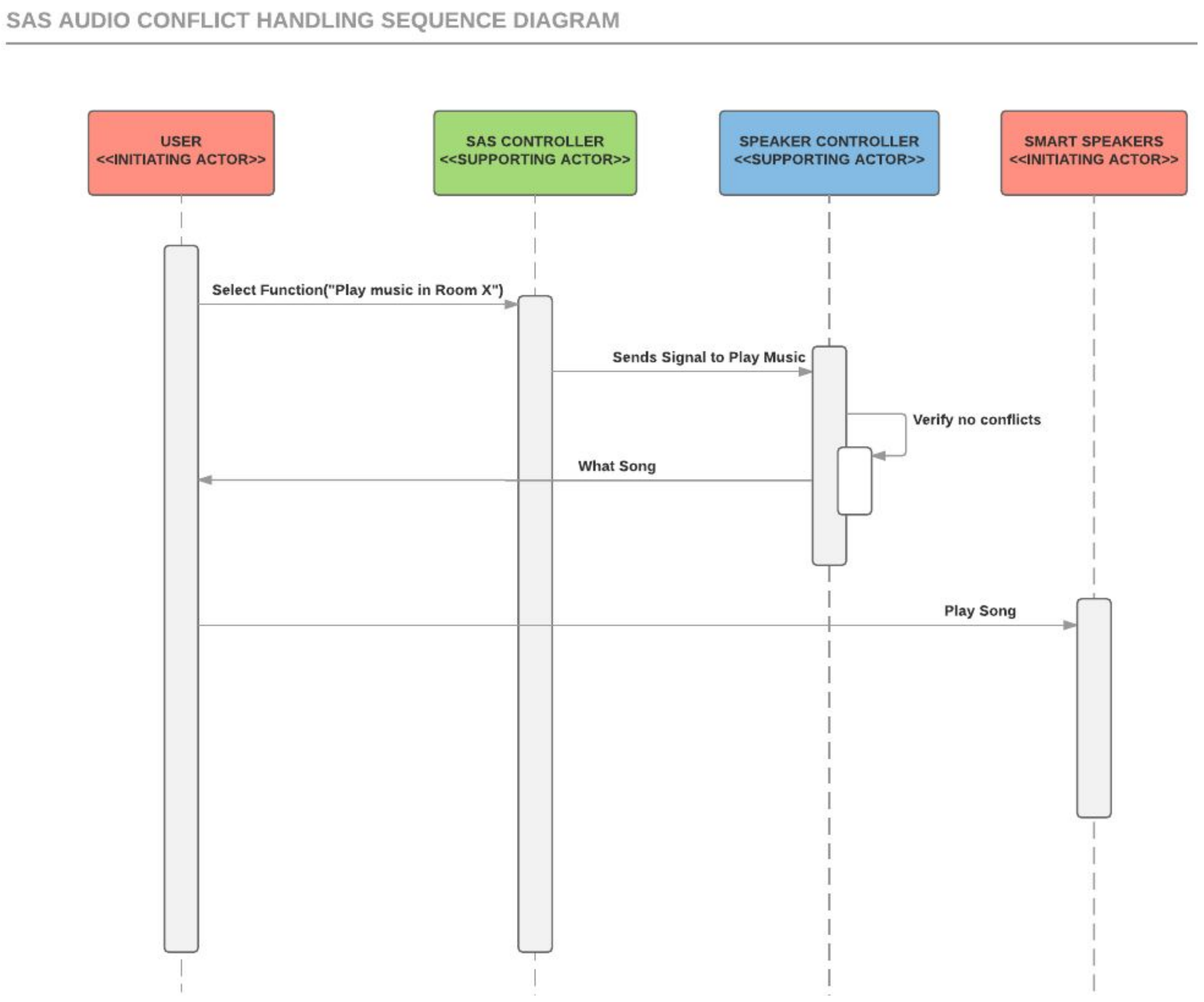
SAS: Alexa Integration Sequence Diagram


SAS ALEXA INTEGRATION SEQUENCE DIAGRAM
USER
SAS CONTROLLER <<SUPPORTING ACTOR>>
AMAZON ALEXA MODULE <<SUPPORTING ACTORE>>
SMART SPEAKERS <<SUPPORTING ACTOR>>
Select function("activate Alexa")
Asks for voice input
Voice Command
Relay Voice Command
Play Alexa Output

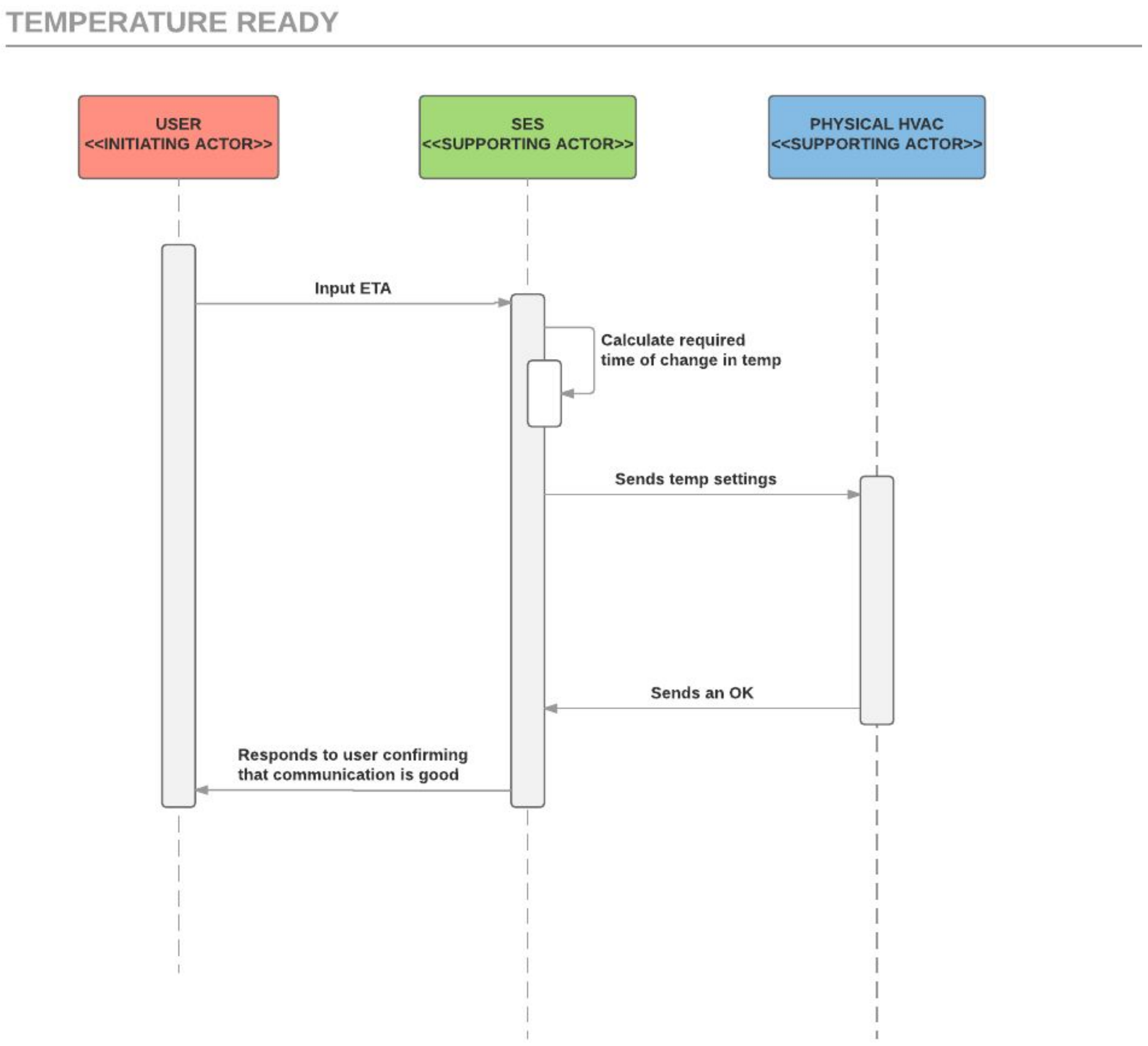
SAS: Notification Handling Sequence Diagram



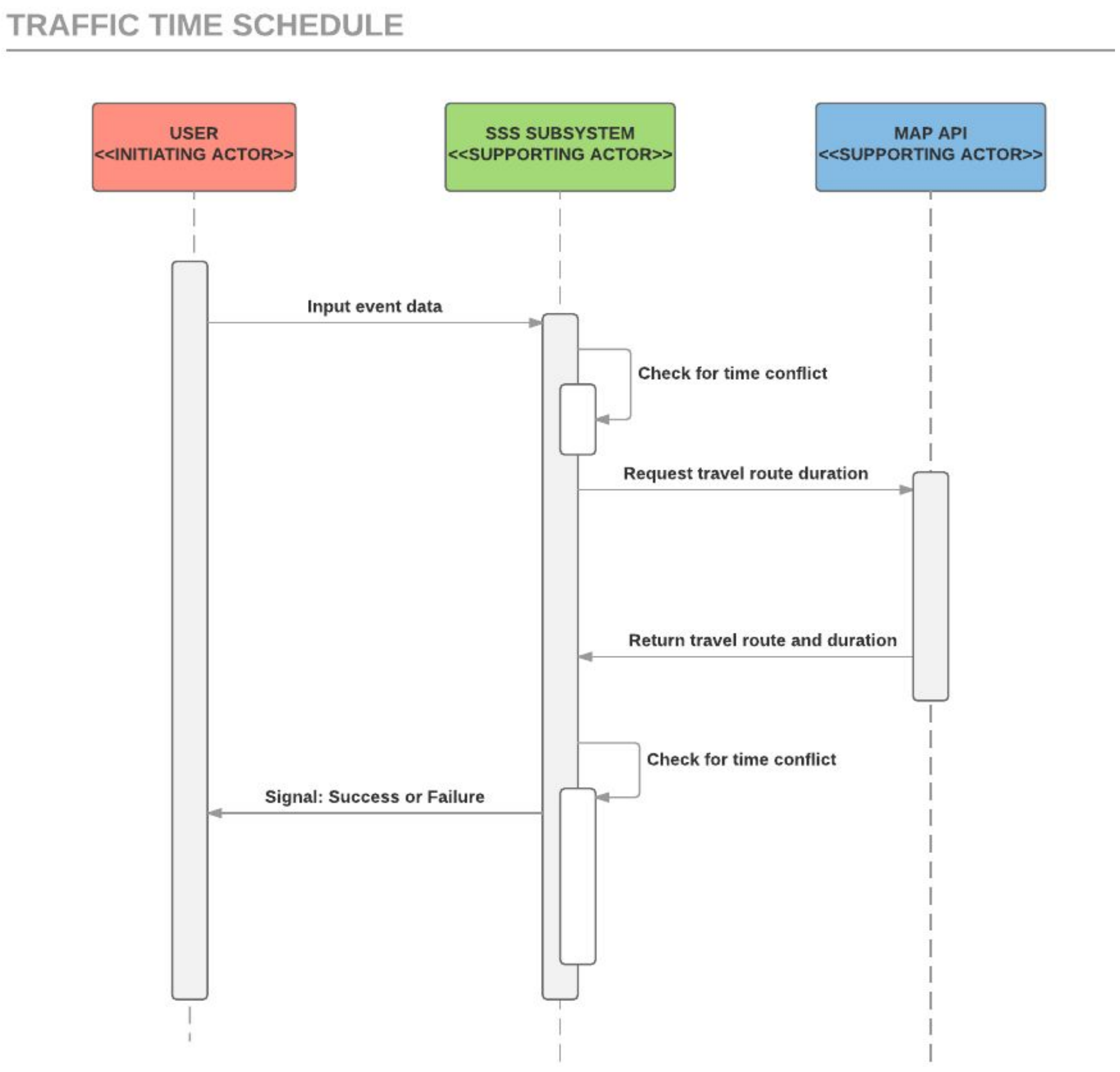
SAS: Audio Conflict Handling Sequence Diagram



SES: Temperature Ready Sequence Diagram


TEMPERATURE READY
USER <<INITIATING ACTOR>>
SES <<SUPPORTING ACTOR>>
PHYSICAL HVAC <<SUPPORTING ACTOR>>
Input ETA
Calculate required time of change in temp
Sends temp settings
Sends an OK
Responds to user confirming that communication is good

SSS: Traffic Time Schedule Sequence Diagram



5. User Effort Estimation using Use Case Points

The navigation of our system is simple. Once you open the application you are given a dashboard with four navigation buttons: Alarm, Sound, Light and HVAC

SAS

- Dashboard
  - Select the SAS button in the sidebar.
  - Click on Various Function you would like to Perform
- Settings
  - Select the SAS button in the sidebar.
  - Click the Settings Gear Icon
  - Toggle Switches of your Preferences
  - Enter User Information in Text Boxes
  - Hit Back Button to Get Back to Functions
- Number of Clicks
  - One Click to Function Dashboard
  - Two Clicks to Access Amazon Alexa
  - Two Clicks to Play Music
  - Two Clicks to get to Settings

SES

- Use case 8 requires 3 clicks [2 navigational, 1 data entry]
  - Click HVAC
  - Cick Time-to-temperature
  - Click Start
- Use case 15 requires 2 clicks [2 navigational]
  - Click HVAC
  - Click Energy Savings
- 5 clicks needed to change heating/cooling preferences[4 navigational, 1 data entry]
  - Click HVAC
  - Click Heating/Cooling Settings
  - Click Edit Temperature Preference
  - Slide to Preferred Temperature
  - Click Save

SSS

This case evaluates the effort required in order to add an event to the SSS assuming that the SHS application is already open.

Navigation:

Select the SSS button in the sidebar.

Select the “add entry” button in order to add an event to the schedule.

Data Entry

Select the “name” field and enter a name.

Select the “location” field and enter a location.

Select the “time field” and enter the time range of the event.

Alternatively, enter a duration and a deadline.

Select the “ok” button.

Project Management For Part 2

We have developed a variant of Agile. There are two phases of the project. The first is from the beginning of the semester till mid February in which we have mainly been communicating, writing up the reports, and discussing ideas. The second phase is when we have reports plus beginning of development so that we have minimum viable product by the first demo. The last phase will be similar to the second phase only it's the time between the first demo and last demo.

While in the first phase of the project, we have had weekly meeting. Initially, there was a few preset things to go over for the agenda like which project idea to pick and how to divide. Later we developed a variant of the standing meeting in which we had a round table on answering the following questions:

1. What did I do in this past week that helped the team with the goal?
2. What will I do this week to help the team reach the goal?
3. Do I see any impediments that prevent me or the development team from meeting its goal?

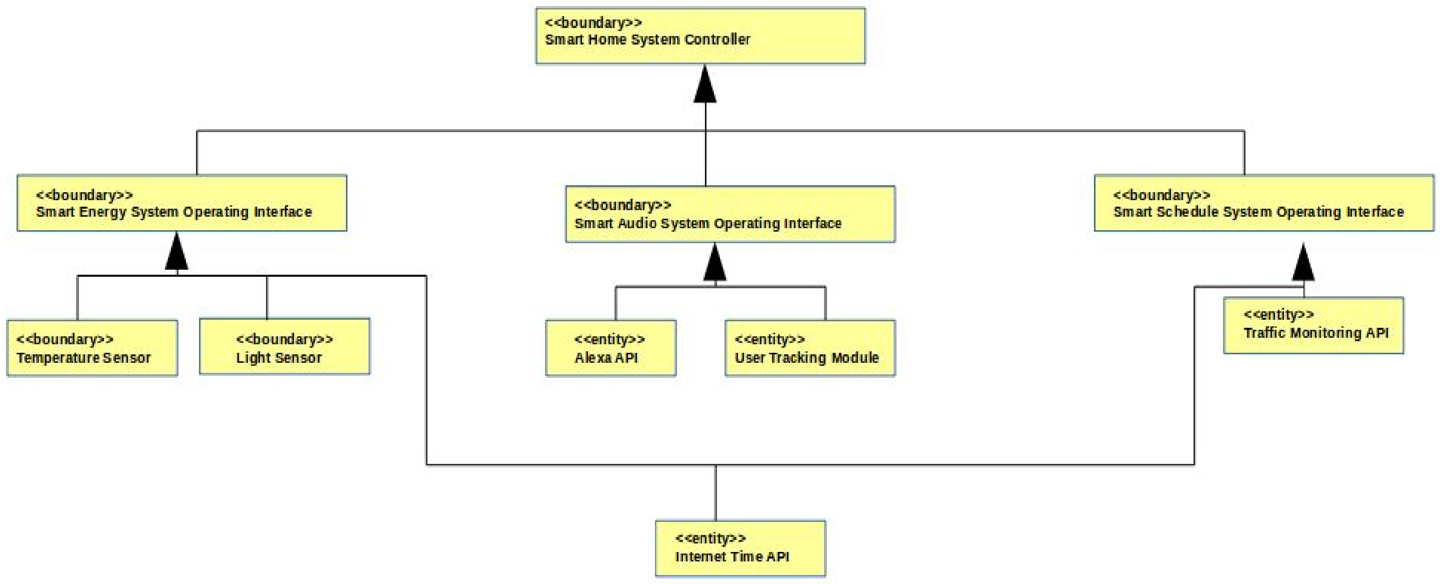
In addition some weekly agenda was talked about based on what was due that week.

As we move into development phase our coordination will be more so on integrating the code and making sure the overall system works. For those weekly meeting it will probably just consist of the stand up meeting questions and then concerns for the group. The rest should be coordinated subgroup to subgroup, person to person.

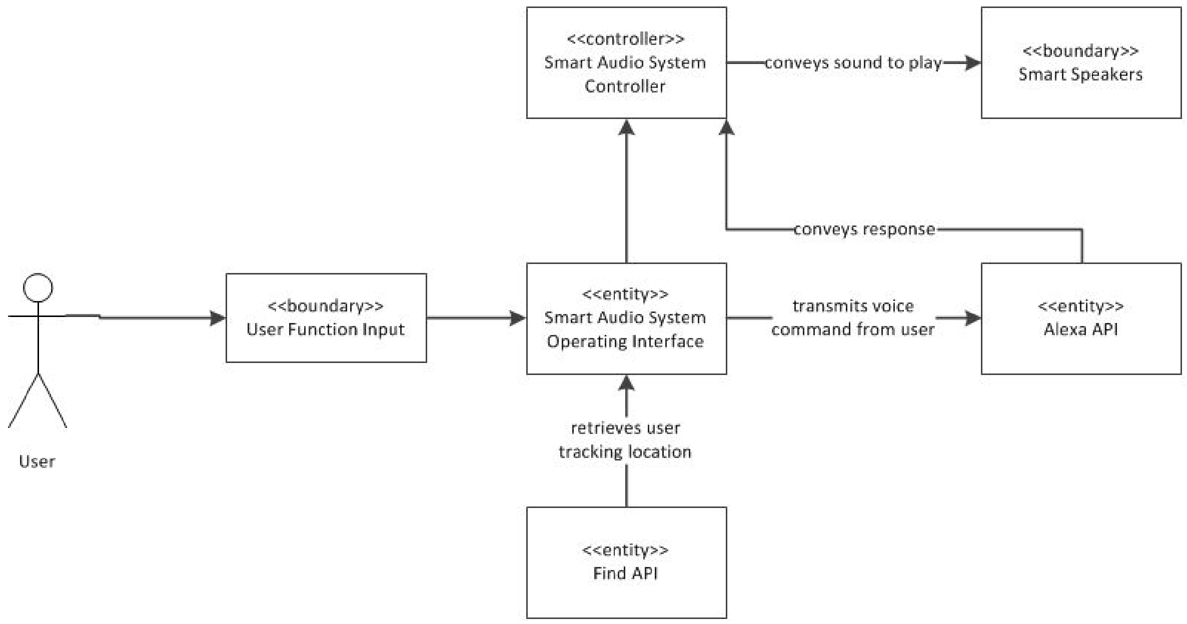
6. Domain Analysis

6.1 Domain Models

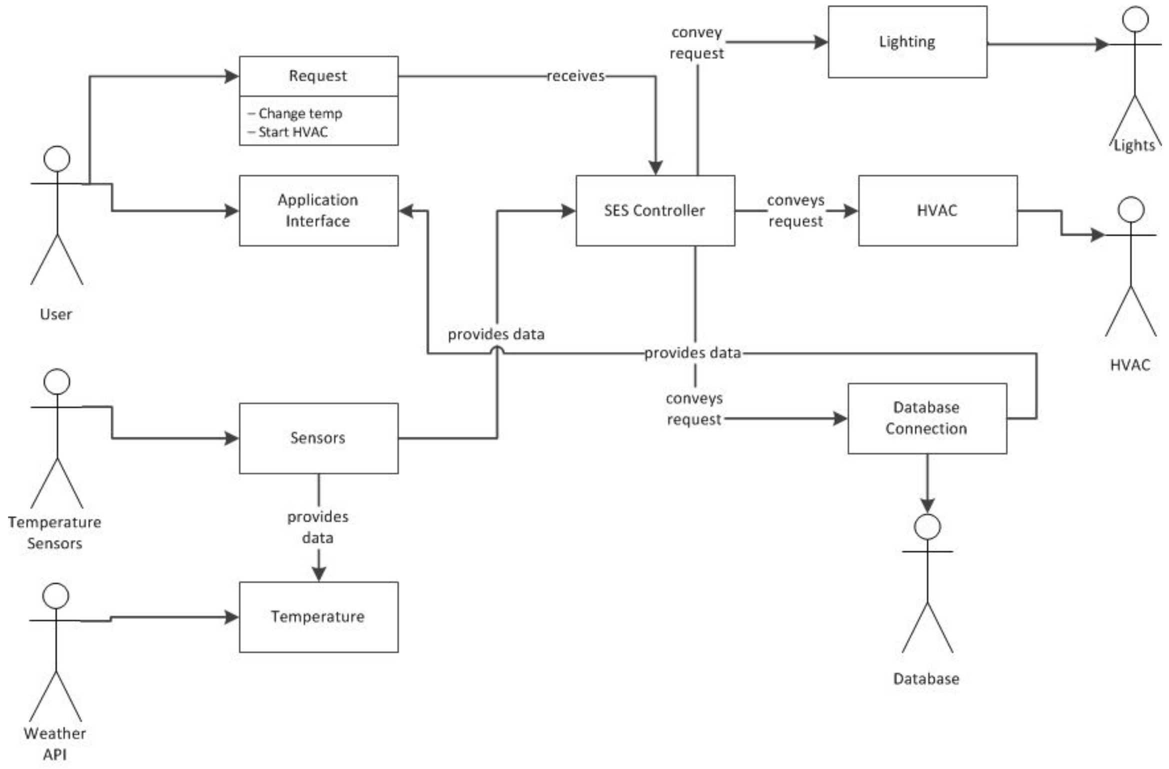
Smart Home System Domain Diagram



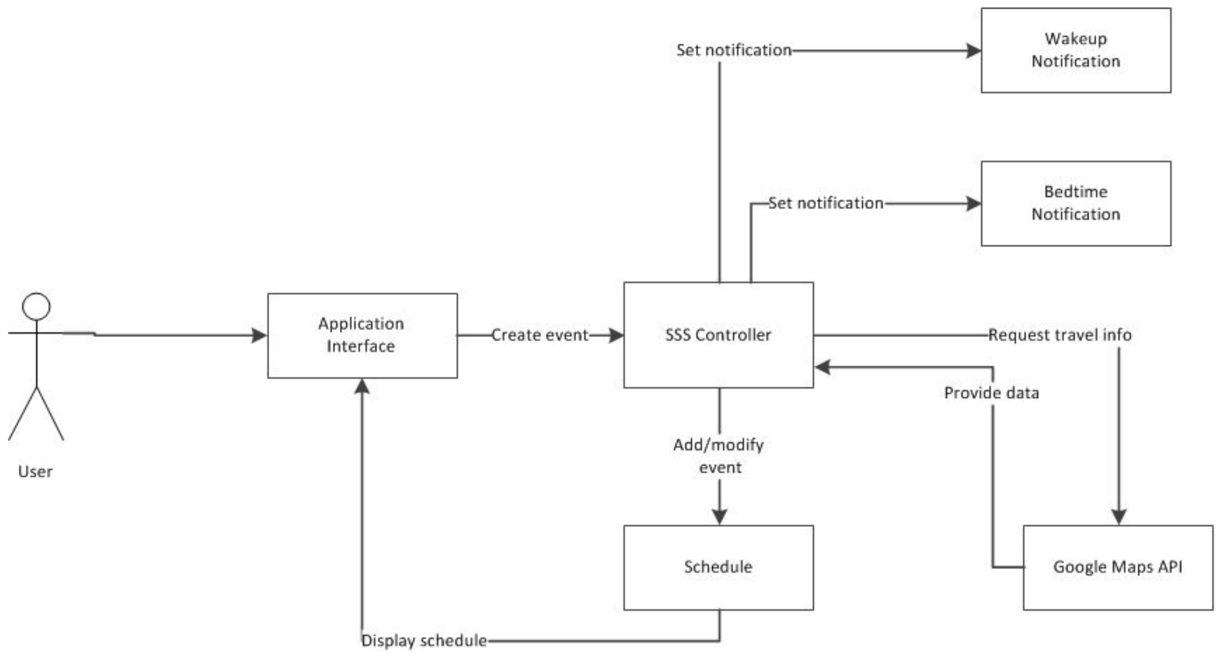
SAS Domain Diagram



SES Domain Diagram



SSS Domain Diagram



6.1.1 Concept Definitions

Description	Type	Name
A controller which controls the features of the Smart Home System	D	**Smart Home System Controller**
System that controls the interactions between the entities of the Smart Audio System	D	**Smart Audio System Operating Interface**
Speakers(Speaker System) which are connected to the internet, Amazon's Alexa, and the internet	K	**Smart Speakers**
Touch input from user app for requesting certain actions and features from the system	D	**User Function Input**
A client/server application that tracks and monitors user's location throughout the home	K	**Indoor Positioning System**
An API created by Amazon for access and use of their voice assistant, Alexa	D	**Alexa API**
Connects the data stream from the application controllers to the database	D	**Database Connection**
App access to information from the internet	K	**Internet Connection**
Controls the HVAC	D	**HVAC**
Controls the physical lights	D	**Lighting**
Handles routing for the system	D	**SES Controller**
Is responsible for the taking data from the sensors and calling an API to provide data for the controller	D	**Temperature**
Temperature sensors that collect live data from the inside of the smart house	K	**Sensor**
Serves users pages and information via application so the user can get the current data about the	K	**Application Interface**
Takes the the input from the user	D	**Request**
A controller which controls the features of the Smart	D	**SSS Controller**

Scheduling System		
A 3rd party software for determining travel time,	**D**	**Google API**
Indicates when the user should wake up	**D**	**Wakeup Notification**
Indicates when the user should go to sleep	**D**	**Bedtime Notification**
Stores all of the events that the user plans to accomplish	**K**	**Schedule**

6.1.2 Association Definitions

Concept Pair	Association Description	Association Name
Sensor ↔ Temperature	Sensor receives data from sensors and sends data to temperature	Provides data
Temperature ↔ Controller	Temperature sents home temp. and outside temp. to controller	Provides data
Request ↔ Controller	Controller receives a request to process	Receives
Controller ↔ Light	Controller conveys users request to control the lights	Conveys request
Controller ↔ HVAC	Controller conveys user's request or request based on sensors to HVAC	Conveys request
Controller ↔ DatabaseConnection	Controller requests info and has database send info to the Application Interface	Conveys request
DatabaseConnection ↔ Application Interface	Provides information for the UI	Provides data
SAS Operating Interface ↔ Indoor Positioning System	SAS Operating Interface retrieves data about location from the Indoor Positioning System	Provides data
SAS Operating System ↔ User Function Input	SAS Operating Interface gets user input from app through User Function Input	Provides data
Alexa API ↔ Smart Audio Operating System	Smart Audio System provides queries for information for the Alexa API. The Alexa API also provides response data from	Conveys request

	queries back to the SAS Operating Interface	
Smart Audio System Controller ↔ Smart Speakers	SAS Audio System Controller conveys what needs to play to the speakers	Conveys data
User Function Input ↔ SAS Operating Interface	SAS Operating System interprets the User Function Input from the user app.	Conveys data
SAS Controller ↔ Alexa API	SAS Controller recieves the information(data) from the Alexa API	Conveys data
SAS Controller ↔ SAS Operating Interface	SAS Operating Interface transfers data to the SAS controller to perform the desired actions.	Conveys data
Application Interface ↔ SSS Controller	Application Interface recieve the data from the user, then send it to to the controller	Create Event
SSS Controller ↔ Schedule	The controller send new data to the database	Add/Modify Event
Schedule ↔ Application Interface	Dispaly schedule data from the local database	Display schedule
SSS Controller ↔ Wake up notification	Set wake up notification base on data	Set notification
SSS Controller ↔ Bed time notification	Set go to bed notification base on data	Set notification
SSS Controller ↔ GG Map APIs	Set HTTP Get request to Google to get traffic data	Request Travel Info
SSS Controller ↔ GG Map APIs	Parsing JSON data from Google and adjust schedule	Provide Data

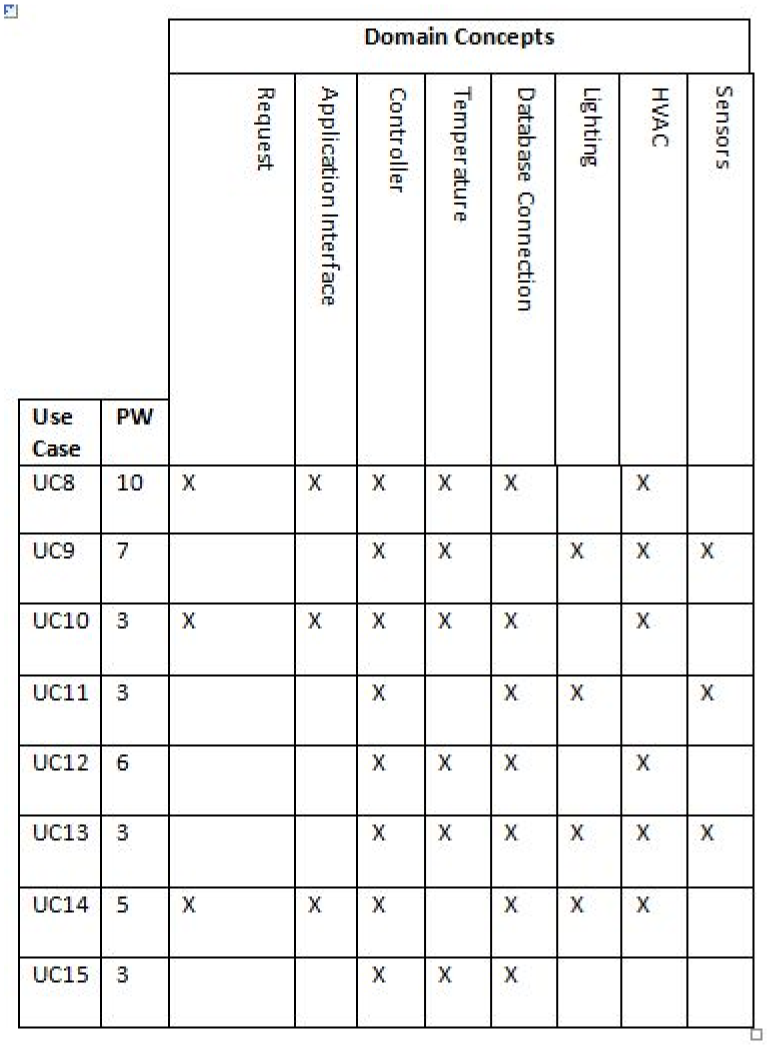
6.1.3 Attribute Definitions

Responsibility	Attribute	Concept
R1: Sends an HTTP POST request to the server to update the prefered temperature	**changePrefTemp**	**Request**
R2: Lets the system know that the user is away so that the temperature can be adjusted	**isAway**	**Request**
R3: Requests for the data related to the calculated saving to generate a graph/chart	**showSavings**	**Request**
R4: Sends payload to frontend with all the required data: current temperature, last preferred temp,	**serveData**	**Application Interface**
R5: Sends the page/UI that the user will interact with	**servePage**	**Application Interface**
R6: Calls all physical sensors and obtains their values	**senseData**	**Sensors**
R7: Sends payload of data to temperature object	**sendData**	**Sensors**
R8: Calls a weather API and returns the outdoor temperature at that particular zip	**getTempAtZip**	**Temperature**
R9: Gets the temperature data from the sensor and brings it to the temperature object	**getSensorData**	**Temperature**
R10: Sends the data to the controller	**sendPayload**	**Temperature**
Takes the temperatures update request and	**handleTempRequest**	**Controller**
Gets location of user by cross checking signal strength from user to mapped location	**locationLookup(Indoor Positioning System)**	**Request**
Amazon Service Request, Response	**connectAmazon(Alexa API)**	**Application Interface**

Play Service Request handling of smart speakers	**relayToSpeaker(Smart Speakers)**	**Request**
Relays Audio to Smart Speakers	**shsAudioHandler**	**Handling**
Play Music Playlist	**playPlaylist**	**Requests**
Area Ownership for Conflicting Audio	**isAudioConflicting**	**Handling**
Status and troubleshooting between connection between SAS and SSS	**getSASSSSInternetworkStatus**	**Monitoring**
Status and troubleshooting between LAN and WAN	**getNetworkStatus**	**Monitoring**
Reset SAS to Default Settings	**factoryResetSAS**	**Troubleshooting**
Reset SES to Default Settings	**factoryResetSES**	**Troubleshooting**
Reset SSS to Default Settings	**factoryResetSSS**	**Troubleshooting**
Reset whole SHS to Default Settings	**factoryResetAll**	**Troubleshooting**

6.1.4 Traceability Matrices

SES: Traceability Matrix

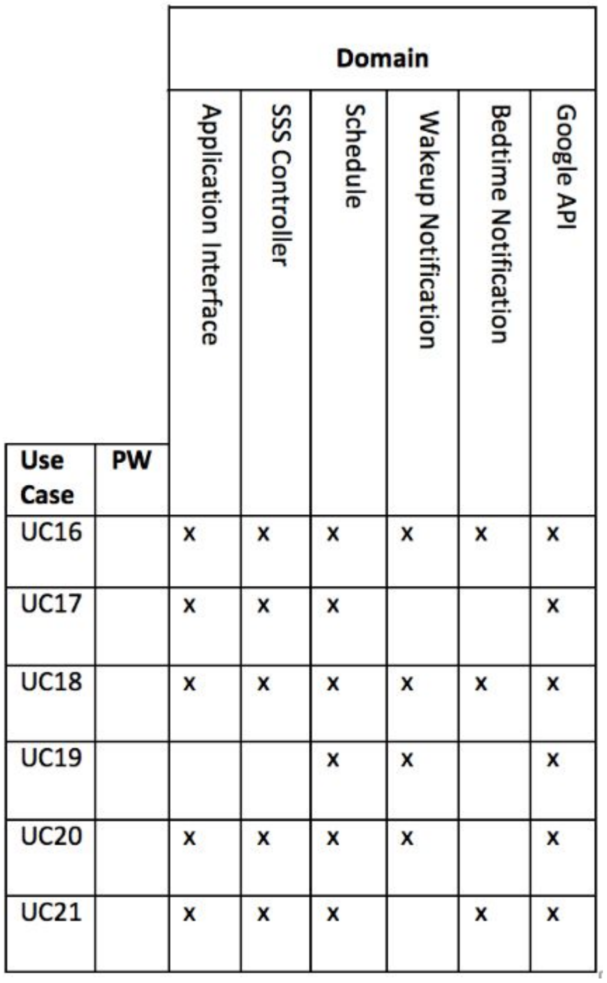


		Domain Concepts							
Use Case	PW	Request	Application Interface	Controller	Temperature	Database Connection	Lighting	HVAC	Sensors
UC8	10	X	X	X	X	X		X	
UC9	7			X	X		X	X	X
UC10	3	X	X	X	X	X		X	
UC11	3			X		X	X		X
UC12	6			X	X	X		X	
UC13	3			X	X	X	X	X	X
UC14	5	X	X	X		X	X	X	
UC15	3			X	X	X			

SAS: Traceability Matrix

		Domain Concepts						
Use Case	**PW**	Smart Audio System Controller	Smart Audio System Operating Interface	Smart Speakers	User Function Input	FIND API	Alexa API	Database Connection
UC1	9			X			X	X
UC2	5	X	X	X		X		
UC3	2	X				X		
UC4	3	X				X		
UC5	4	X		X			X	X
UC6	5	X		X				

SSS: Traceability Matrix



		Domain					
Use Case	PW	Application Interface	SSS Controller	Schedule	Wakeup Notification	Bedtime Notification	Google API
UC16		x	x	x	x	x	x
UC17		x	x	x			x
UC18		x	x	x	x	x	x
UC19				x	x		x
UC20		x	x	x	x		x
UC21		x	x	x		x	x

6.2 System Operation Contracts [5]

Operation	**Schedule Event Addition**
Responsibilities:	Add new events
Use case:	UC-16
Exception:	Throw an error to user.
Preconditions:	App install successful. User give app access to device location, calendar. Device has internet access. Device has internet access. Device has GPS location.
Postconditions:	Updated schedule. Updated wake-up notification time. Updated departure times.

Operation	**Checking relevant information about traffic**
Responsibilities:	Fetch new data to update notification frequently
Use case:	UC-17
Exception:	None
Preconditions:	Device has internet access when fetching new data Device has GPS location when fetching new data
Postconditions:	Traffic and weather conditions updated successfully Alarms adjusted correspondingly to the new data.

Name:	**Temperature Ready**
Responsibilities:	Predict time needed for the home to adjust to a certain temperature and allow user to initiate heating/cooling when necessary
Use Case:	UC-8
Exceptions:	None
Preconditions:	User has defined set point temperature
Postconditions:	Time-to-temperature is displayed on the application along with a start button for heating/cooling

Name:	**Dual-Zone Inactivity**
Responsibilities:	When movement hasn't been detected for a fixed amount of time, heating/cooling systems and lights are adjusted
Use Case:	UC-9
Exceptions:	None
Preconditions:	No movement has been detected in x minutes
Postconditions:	Heating/cooling are adjusted to energy efficient settings; lights are shut off

Name:	**Amazon Alexa Integration**
Responsibilities:	When a request is sent to Amazon's Alexa through the app, the voice assistant will perform the desired task or retrieve the requested data.
Use Case:	UC-5
Exceptions:	If the voice is not interpreted properly, the app will ask for the same input once again from the user.
Preconditions:	User must have Alexa services enabled in the app settings and must have the activation word set up.
Postconditions:	Amazon's Alexa interprets the user's voice request and retrieve the desired data and/or perform the desired action.

Name:	**Speakers for SSS**
Responsibilities:	After the user is woken by the SSS, the SAS will read the user's schedule through Amazon's Alexa.
Use Case:	UC-6
Exceptions:	None
Preconditions:	The alarm goes off to wake the user, and the user wakes up and dismissed the alarm.
Postconditions:	The app will use Amazon's Alexa to notify the user of his/her schedule for the day ahead.

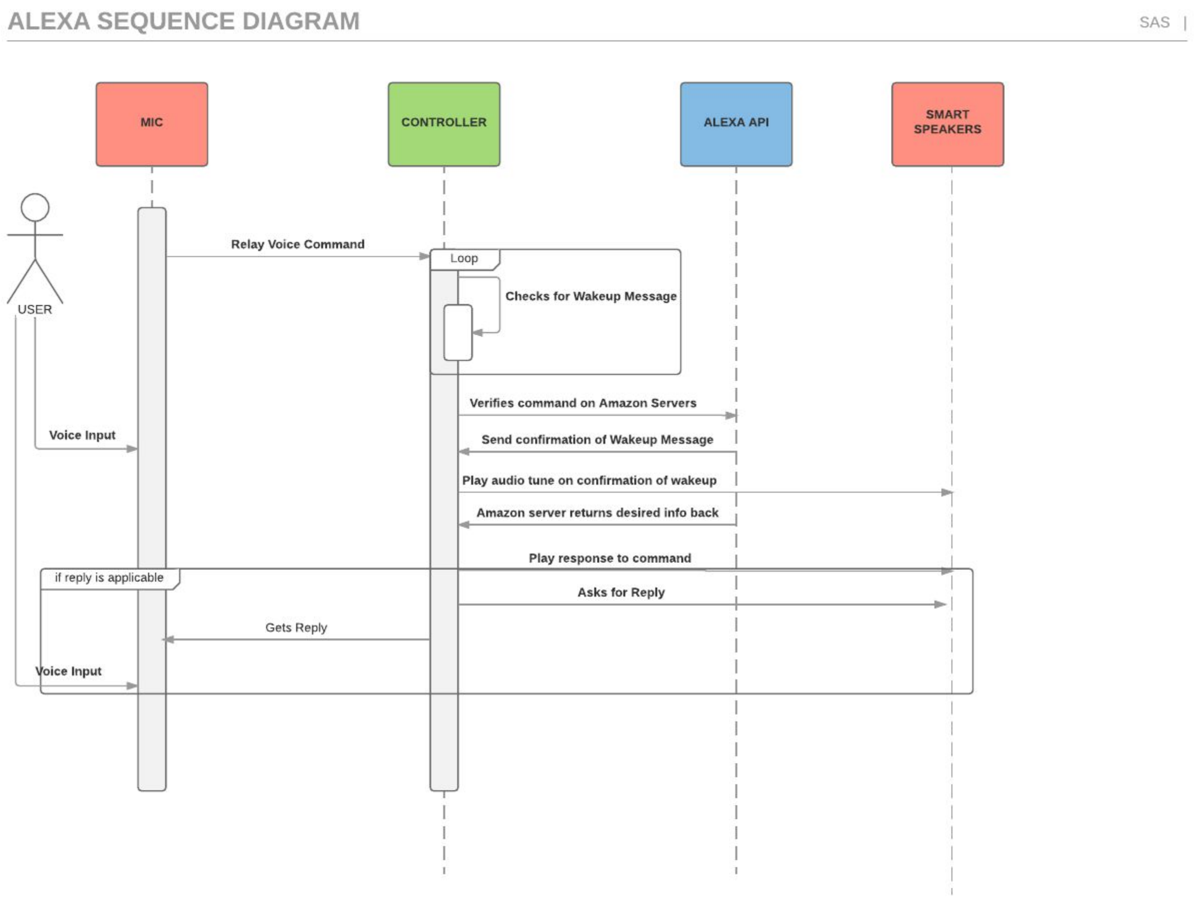
6.3 Mathematical Model

For SES, we decided to implement a model similar to Nest's thermal model [6], which is used to predict the time-to-temperature feature of our system. For our model, we plan on using the current state of the home and the HVAC state as the inputs. Sensors will collect data throughout the day and train the model so the system can make better predictions.We can also use Newton's Law of Cooling [7] for modeling the heating and cooling of a house. If we consider T(t) represent the temperature inside the house at time t. H(t) is the rate of increase in temperature is. U(t) is the increase or decrease of temperature. M(t) is the outside temperature of the building.

7. Interaction Diagrams

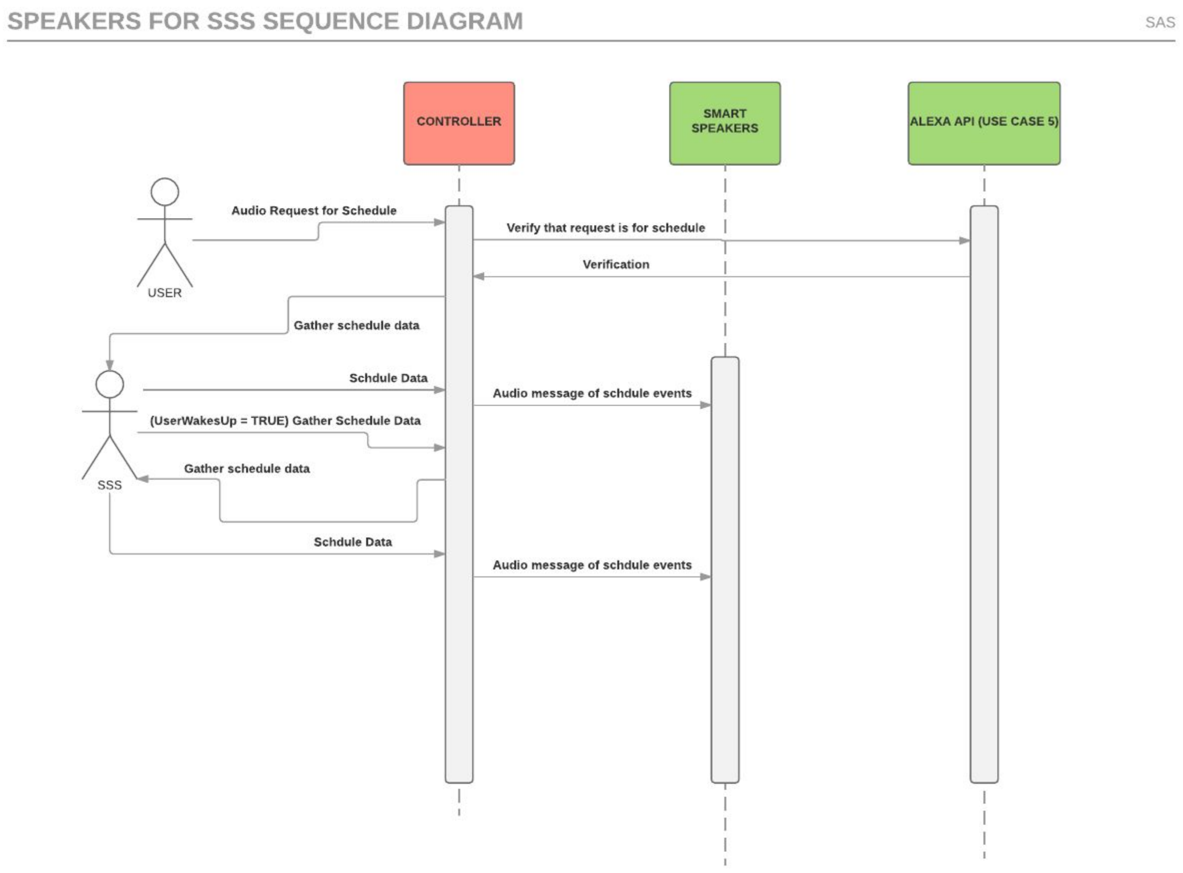
Use Case 5 - Amazon Alexa Interaction [6]

The user can access all of Amazon Voice Assistant Services from the SAS. The functionality of Amazon's voice assistant is baked into the SAS.



Use Case 6 - Speakers for SSS

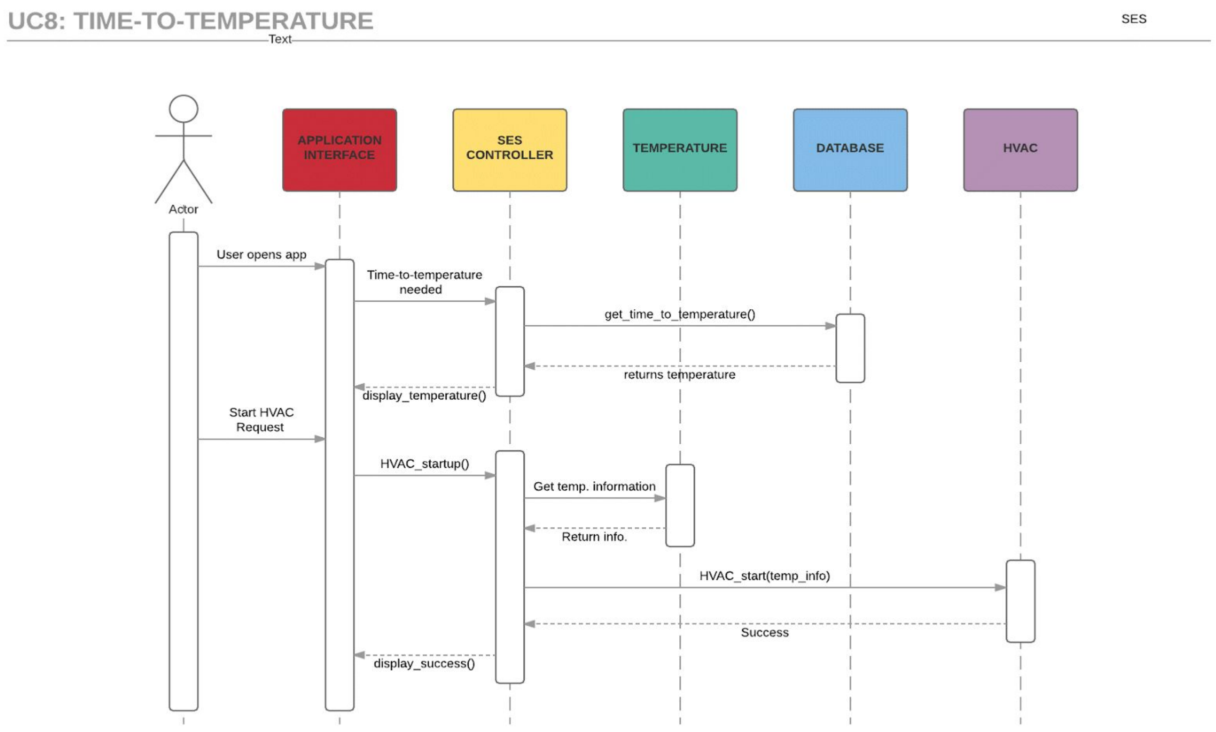
The user can hear notifications from the SSS on the SAS.



SPEAKERS FOR SSS SEQUENCE DIAGRAM
SAS
CONTROLLER
SMART SPEAKERS
ALEXA API (USE CASE 5)
USER
Audio Request for Schedule
Verify that request is for schedule
Verification
Gather schedule data
SSS
Schdule Data
Audio message of schdule events
(UserWakesUp = TRUE) Gather Schedule Data
Gather schedule data
Schdule Data
Audio message of schdule events

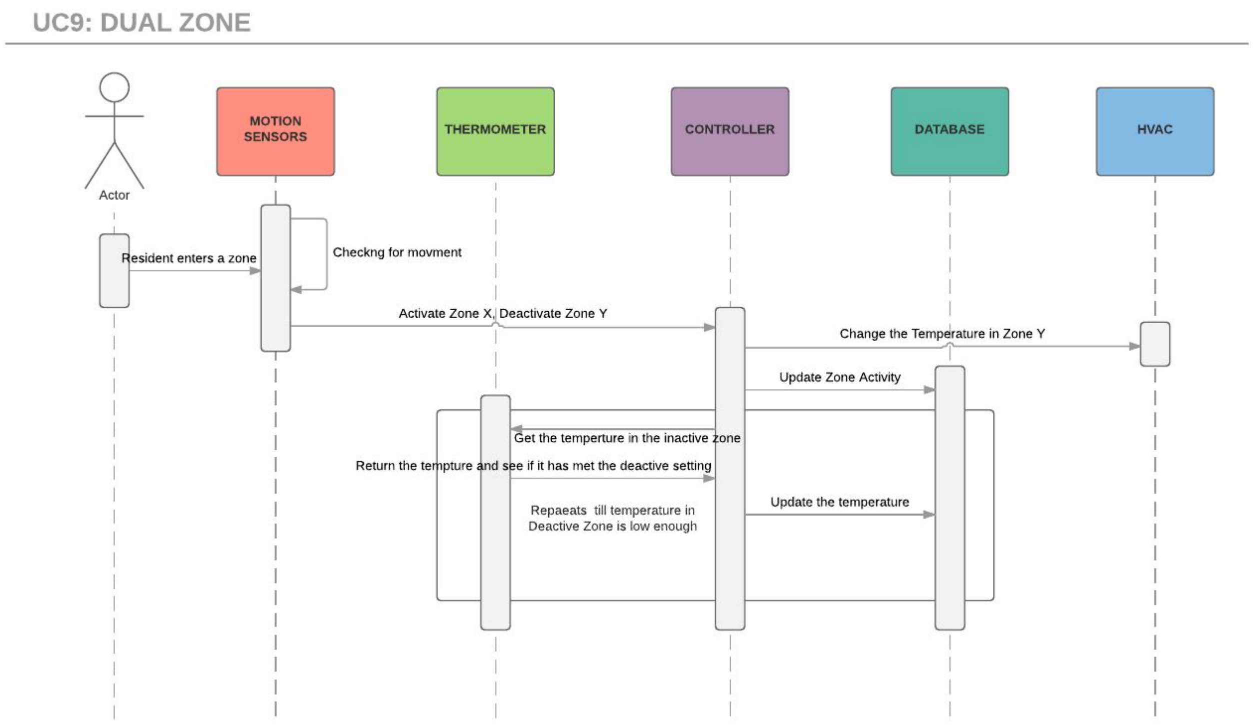
Use Case 8 - Temperature Ready

User is returning home from work/errands and checks application for the time-to-temperature prediction and initiates heating/cooling when time-to-home is approximately the same.



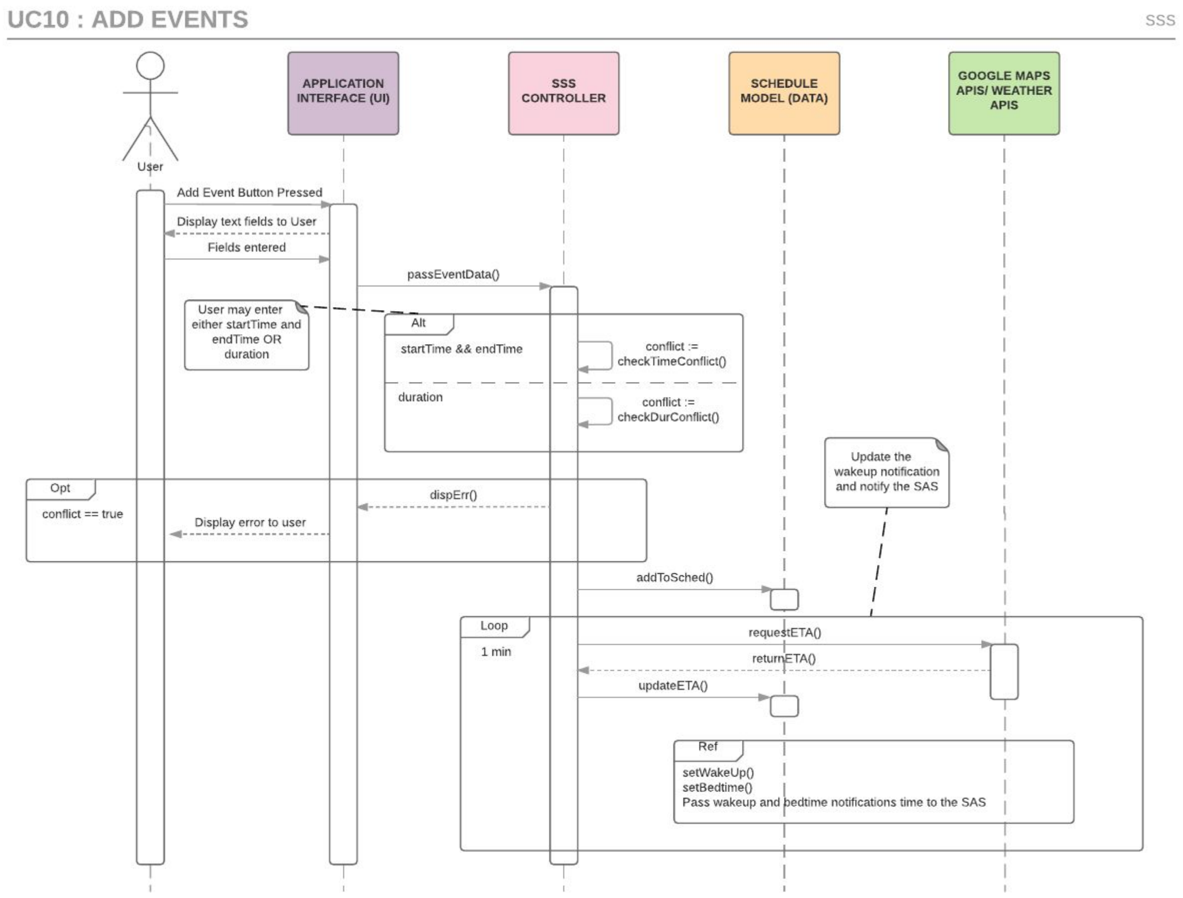
Use Case 9 — Dual Zone Activation/Deactivation

No movement has been detected for a certain duration of time in one or both of the zones in a dual heating/cooling home and the temperature adjusts to minimize energy consumption and lights are shut off in the respective zone.


UC9: DUAL ZONE
Actor
MOTION SENSORS
THERMOMETER
CONTROLLER
DATABASE
HVAC
Resident enters a zone
Checkng for movment
Activate Zone X, Deactivate Zone Y
Change the Temperature in Zone Y
Update Zone Activity
Get the temperture in the inactive zone
Return the tempture and see if it has met the deactive setting
Update the temperature
Repaeats till temperature in Deactive Zone is low enough

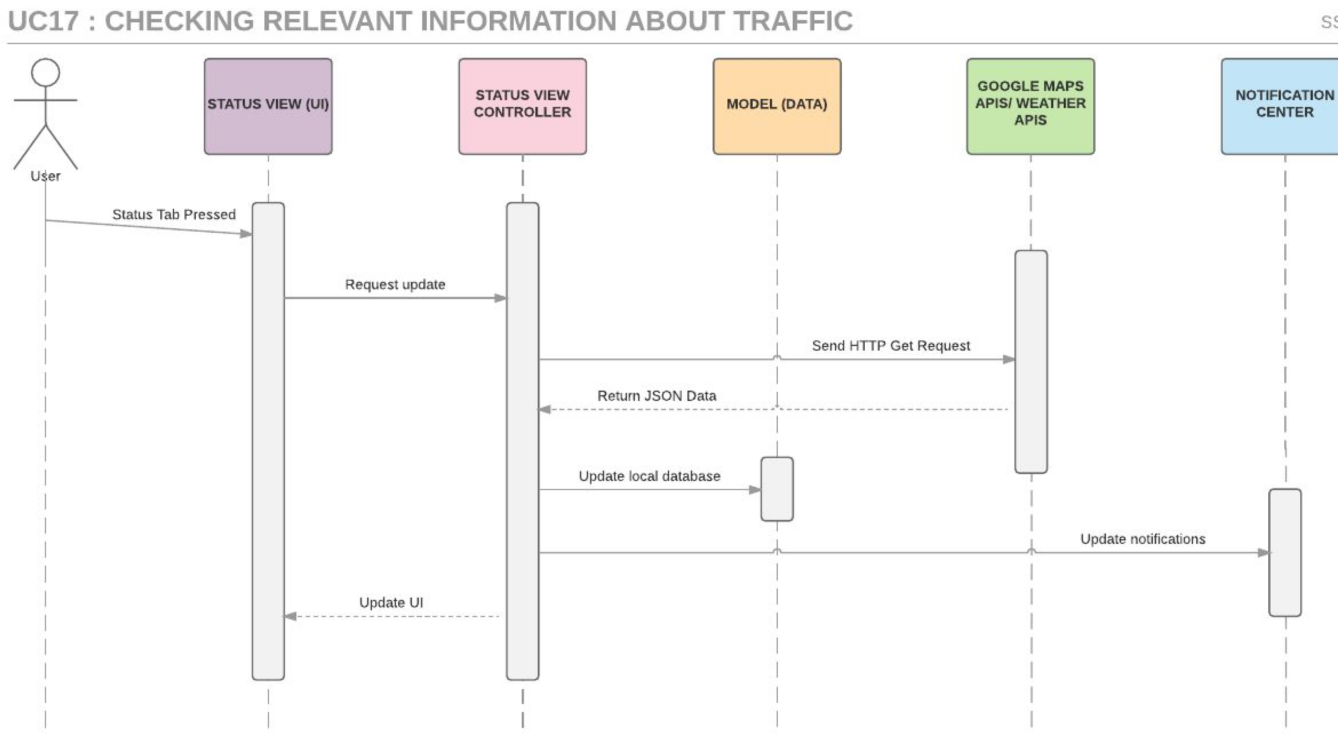
Use Case 18 — Schedule Event Addition

User can modify his schedule, destination, add holiday, change alarm sound.


UC10 : ADD EVENTS
SSS
User
APPLICATION INTERFACE (UI)
SSS CONTROLLER
SCHEDULE MODEL (DATA)
GOOGLE MAPS APIS/ WEATHER APIS
Add Event Button Pressed
Display text fields to User
Fields entered
passEventData()
User may enter either startTime and endTime OR duration
Alt
startTime && endTime
conflict := checkTimeConflict()
duration
conflict := checkDurConflict()
Opt
conflict == true
dispErr()
Display error to user
Update the wakeup notification and notify the SAS
addToSched()
Loop
1 min
requestETA()
returnETA()
updateETA()
Ref
setWakeUp()
setBedtime()
Pass wakeup and bedtime notifications time to the SAS

Use Case 17 - Relevant Traffic Information

User can view weather condition, traffic on his route, time taken to his next destination when he opens the app.



Description of Design Principles

Our design principles are a hybrid of low coupling principles, expert doing, and high cohesion. Each object is an expert at a small task and communicates its information to other objects. For example, in use case 9 there are many actors, like the physical sensors, thermometer, and HVAC [8]. Each piece of hardware has his its own object that controls its physical responses. In order to preserve the cohesiveness of each object, the controller class was given a significant amount of responsibility.

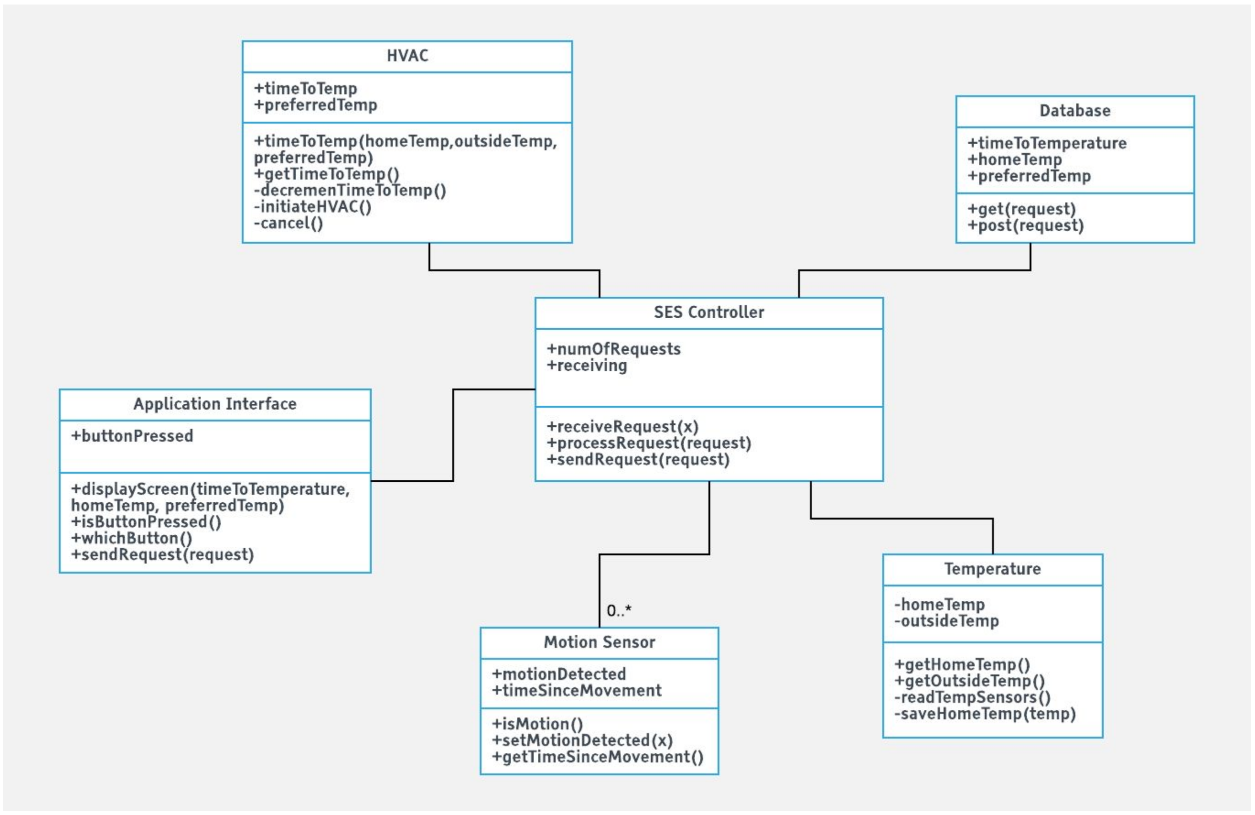
Each of our subsystem is its own stand alone system. Technically, each subsystem can function minimally by itself. Each subsystem sends data to other subsystems to complete certain tasks; the SSS would send audio data to the SAS to play the audio through the speakers The controllers of each our subsystem employ tight coupling however each of the objects that communicate with the controller employ loose coupling. The job that controllers play is generally to identify the type of information it receives from the actor and relay it to different objects. In theory controller is high cohesion but since it handles many different type of information it's more logical to label the controllers as low cohesion.

Overall we are type 1 and type 3: responsibility of knowing and responsibility of communicating respectively.

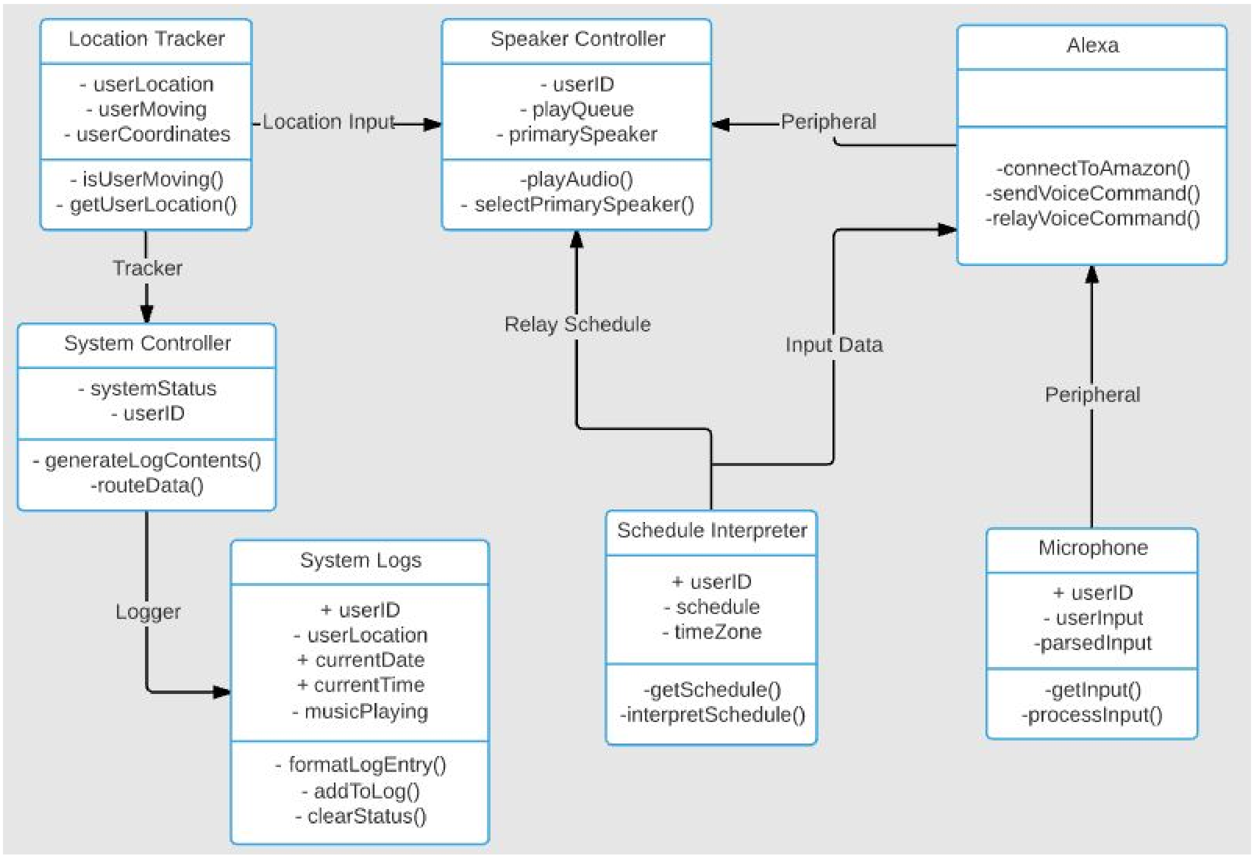
8. Class Diagram and Interface Specification

A. Class Diagrams

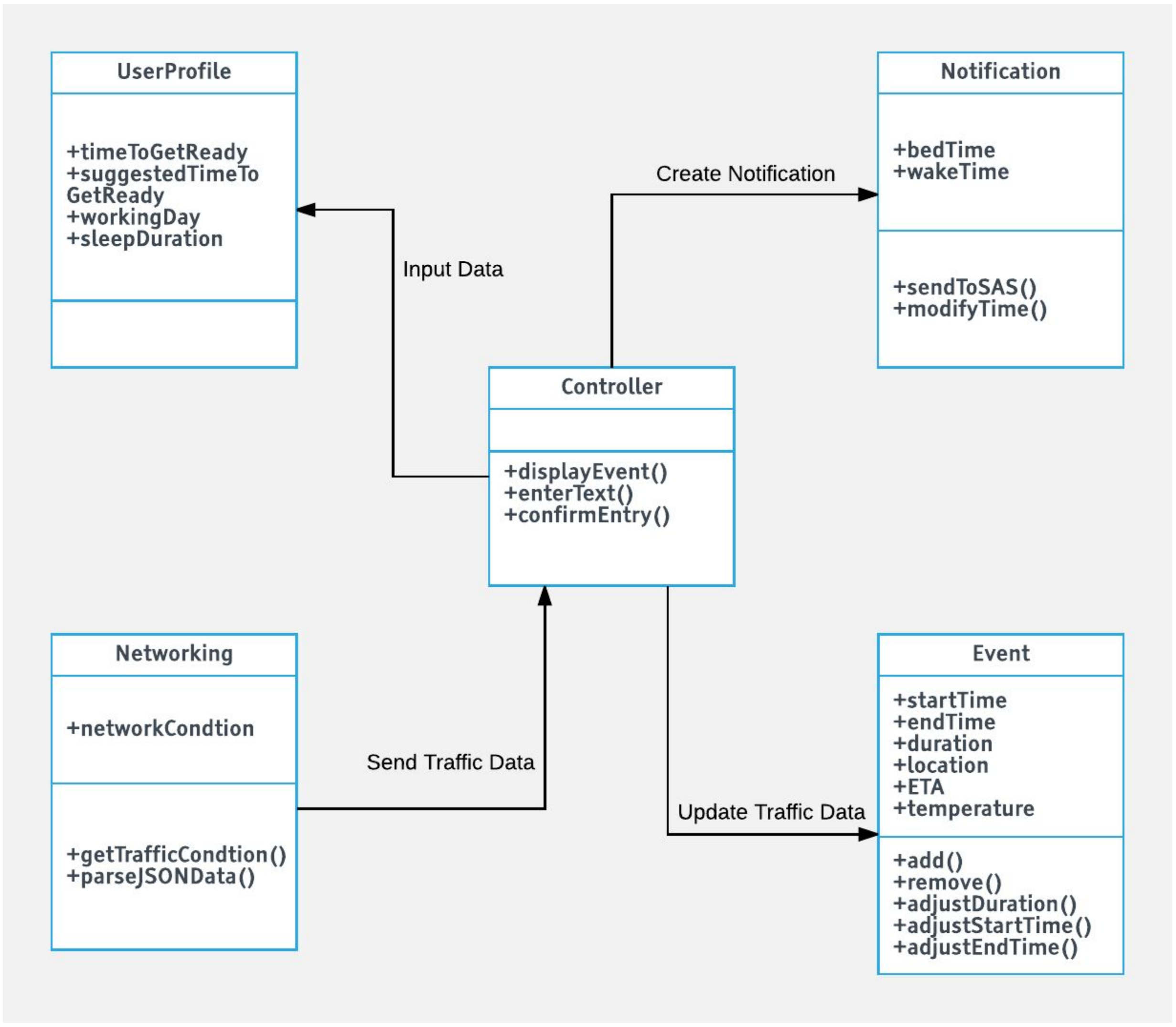
SES Class Diagram



SAS Class Diagram



SSS Class Diagram



B. Data Types and Operation Signatures

Class: SES Controller

The SES Controller controls all requests and controls the exchange of information between different classes.

Attributes:

+ numOfRequests: int --- Current number of requests being received by the controller

Operations:

+ processRequest(request): boolean --- Receives a request struct and extracts relevant information. Returns boolean 1 if processed correctly.
+ receiveRequest(x: bool): boolean --- Sets the controller's 'receiving' data item to one or zero which controls flow of requests. Returns 1 for success.
+ sendRequest(request: Request, targetClass): bool --- Reroutes request to target class for handling.

Class: HVAC

This class directly controls the heating/cooling system and calculates important data used for many use cases.

Attributes:

+ timeToTemp: int --- The time needed to go from the current home temperature to the preferred temperature.
+ preferredTemp: float --- User's preferred home temperature

Operations:

+ timeToTemp(homeTemp: float, outsideTemp: float, preferredTemp: float): float --- Calculates the time needed to transition between current temperature and preferred temperature.
+ getTimeToTemp(): float --- Returns timeToTemp value

-decrementTimeToTemp(): boolean --- Decrements timeToTemp as time progresses after the user initiates heating/cooling

- initiateHVAC(): boolean --- Initiates the physical heating/cooling system
- cancel(): boolean --- Stops heating/cooling process from starting

Class: Database Controller

The database stores all important data for later use

Attributes:

+ timeToTemperature: float --- Described in the previous class and stored for data persistence

+ homeTemp: float --- Temperature inside the user’s home

+ preferredTemp: float --- User’s desired home temperature

Operations:

+get(request): boolean --- Used to request information from the database

+set(request): boolean --- Used to update or create new data in the database

Class: Application Interface

Displays and provides an interface with the application/website

Attributes:

+ buttonPressed: boolean --- Boolean value that keeps track of when a button has been pushed

Operations:

+ displayScreen(timeToTemperature: float, homeTemp: float, prefferedTemp: float): void --- Sends and displays important data to the GUI

+ isButtonPressed(): boolean --- Returns 1 if user has pressed a button, else returns a 0

+ whichButton(): string --- This function returns the string name of the button that was pressed

+ sendRequest(request): boolean --- Sends a request struct to the controller for processing

Class: Temperature

Interfaces with hardware and API’s to obtain relevant temperature information

Attributes:

- homeTemp: float --- Temperature in user’s home

- outsideTemp: float --- Temperature outside user’s home

Operations:

+ getHomeTemp(): float --- Returns homeTemp

+ getOutsideTemp(): float --- Returns outsideTemp

- readTempSensors(): float --- Interfaces with hardware to retrieve temp. data

- saveHomeTemp(temp: int): boolean --- Changes homeTemp variable to the argument passed in the function

Class: Motion Sensor

Interfaces with motion sensors and keeps track of time dependent data

Attributes:

+ motionDetected: boolean --- Variable that is 1 when motion is detected and 0 when no motion detected

+ timeSinceMovement: float --- Stores how much time has passed since the last movement was detected

Operations:

+ isMotion(): boolean --- Interfaces with hardware and returns 1 when PIR sensor is set off

+ setMotionDetected(x: boolean): boolean --- Sets motionDetected to 0/1 based on the function parameter

+ getTimeSinceMovement(): float --- Returns timeSinceMovement

Class: Location Tracker

Interface with SAS controller and provide user location

Attributes:

- userLocation : holds float or double(TBD) based based on user location
- userCoordinates : (int array), coordinate location of the user in the house

Operations:

- isUserMoving(): boolean - returns boolean value whether the user is stationary or in motion
- getUserLocation():int/float - return back the approximate location of the user in the home

Class: Speaker Controller

Attributes:

+ userID : (int/long/hash key), ID of user who is currently using the system

- playQueue : (struct), list of the names of the next files to be played on the speaker system
- primarySpeaker : (string), closest speaker to the user

Operations:

- playAudio(): int - play audio that has been cached on the Speaker Controller. Similar to a print queue
- selectPrimarySpeaker(): finds the key of the speaker closest to the user

C. Traceability Matrices

SAS Traceability Matrix

	Classes						
Domain	**Location Tracker**	**Alexa**	**Microphone Controller**	**Schedule Interpreter**	**System Logs**	**System Controlle r**	**Speaker Controller**
User Function Input			The microphone is used to gather user input		The logs will maintain and/or display important user info		
SAS Controller	The controller will read the data from the location tracker	Alexa replies to voice command are relayed to controller			The logs will maintain and/or display important controller info	The SAS controller is the system controller	
SAS Operating Interface						The system controller provide informatio n to the UI	
Smart Speakers					The logs will maintain and/or display important speaker info		The speaker controller controls the speakers
Alexa API		The Alexa API allows the system to integrate with Alexa					
Indoor Positioning System	Indoor Positioning System includes all the key function of the				The logs will maintain and/or display important user info		

	tracker.						

SSS Traceability Matrix

Domain	Classes				
	User Profile	**Controller**	**Notifications**	**Event**	**Networking**
Application Interface		Controller get data and display UI			
Controller					Parse JSON Data return from networking class
Schedule				List of events	
Wake Up Notification				Send user notification from event lists	
Bedtime Notification				Send user notification from event lists	
Google Maps API					Networking class get data from GG maps and add them to events

SES Traceability Matrix

Domain	Clases					
	HVAC	**Application Interface**	**Motion Sensor**	**SES Controller**	**Temperature**	**Database**
Request		Get the views from the application controller				Post requests to the database
Application Interface				Is main iterator	Display data from temperature	Parse JSON data returning from db
Sensors		Displays where interaction is				Posts to the databases at time intervals
SES Controller	Interacts with HVAC to send boolean values	Forwards views	Forwards data to database connection		Gets data points from temperature objects	Is the primary communicator of the database controller.
Lighting		Displays				

		where heat points are				

9. System Architecture and System Design

A. Architecture Style

We have a Control Center which all subsystems will communicate with. Each subsystem with upload its data, get relevant data of other subsystems through Control Center. Also we adhere to the MVC design pattern.

The model represents the data, and does nothing else. The model does NOT depend on the controller or the view.

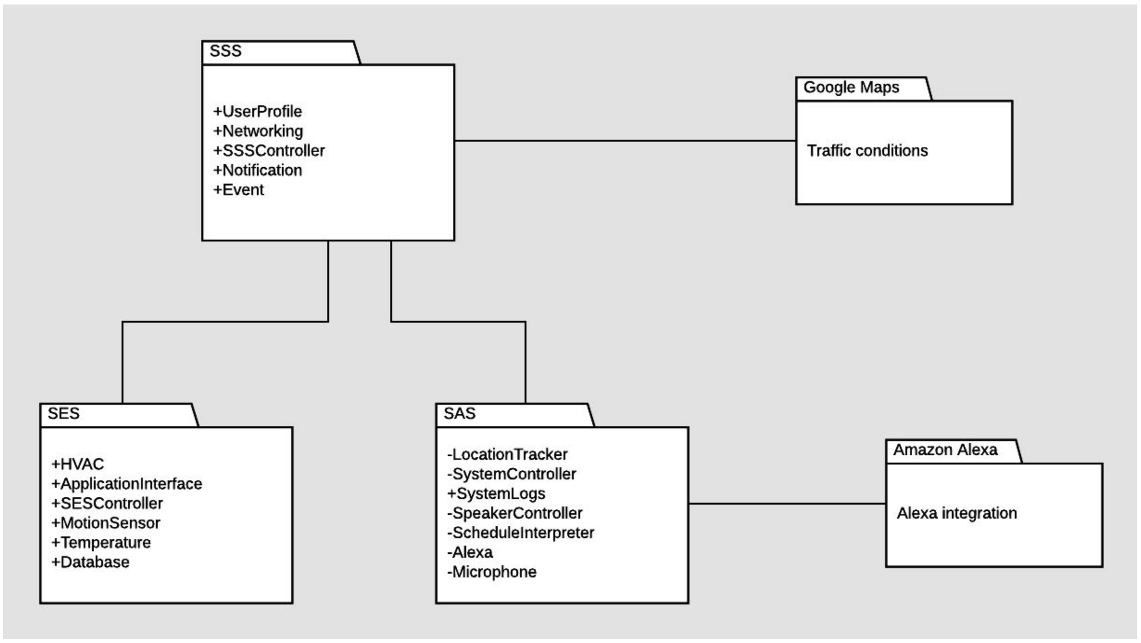
The view displays the model data, and sends user actions (e.g. button clicks) to the controller. The view can be:

- independent of both the model and the controller
- the controller, and therefore depend on the model

The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model. In some cases, the controller and the view are the same object.

The SAS's Indoor Positioning System will also have its own independent server strictly for user tracking processing purposes. Unlike the database, this machine will not be connected to the Wide Area Network. But the biggest difference between our Control Center and this Indoor Positioning System Server is that there will be a significant amount of processing of information which will occur instead only pushing/pulling data.

B. Identifying Subsystems



C. Mapping Subsystems to Hardware

In general the project adheres to a client/server model. The client could be any device: a laptop, mobile phone, really anything with a browser. The Control Center server program needs to be hosted either on one of our laptops or we can get hosting from AWS or Heroku; while the Indoor Positioning System's Server will need a dedicated server with a little more "horsepower." All three subsystems at least for the dashboard part of the application are mapped to this client/server model.The SAS will require a speaker module. This system will also require a minimum of 1Mbps steady WAN throughput at all time to function properly with a FastEthernet(100Mbps) LAN connection with the other subsystems and the database. Wireless client machines will need a minimum of 802.11n wireless interface card to function without any hiccups. For our purposes we used a raspberry pi to run our subsystem and connect to Amazon services for Alexa.

D. Persistent Data Storage

The database schema of choice is a Relational Database. This is because we will have an organized element look up table. Some data for the SES subsystem will be Current Temp: float, Preferred Temp: float, Time to Temp: float, Motion Detected: boolean, timeSinceMovement float, Outside Temp: float. The SAS user tracking software will have reference points stored in the database. These reference points are used to pinpoint user location. SSS will save a list of events.

E. Network Protocol

The Indoor Positioning System will use C sockets to communicate between the user's device and the positioning central server. The reason we chose this was to more finely control the hardware and scalability for the future. The max number of user audio tracking devices will directly correlate to the machine that the SAS is running on. For example as the number of user increase dramatically the machine can(TBD) fork new processes to handle new users and track them. The number of forked processes will depend on the power of the host machine. This provides the user with an efficient way of scaling the the SAS system. When referring to the OSI model we are mostly concerned in fine tuning the Network, Transport, Session, and to a lesser extent the Data Link layer. In terms of the Data Link we are pulling the Wi-Fi signal strength from the network interface card. For networking we are going to stick with ipv4 as their isn't any real reasons to use ipv6 on the local area network as the positioning system will not access any wide area network. In terms of the Session layer we will be using the connection oriented protocol TCP with respects to the data packets traversing the network. FInally, for Session we will use a large unregistered port for the socket communication so we do not interfere with any pre existing user devices/services. The user will be able to change the port number if it just so happens if the port number is in use.

For the SSS we will use HTTPs request to get data from Google Maps API [3]. We will also use OpenWeatherMap's API[18] to retrieve temperature information used in the SES system. In between our objects http requests will be used to transfer information and route requests. In general the rest of the system will be using HTTP get and post requests to send data in between a central server, the app, and unity.

F. Global Control Flow

Smart Energy System: **Control Flow**

A majority of our code is event driven. As seen in our use cases, most of our scenarios depend on the user interacting with the system or some sort of inactivity. For example, when no motion has been detected for a fixed amount of time in an area, the heating/cooling minimizes energy consumption. The system depends on time for certain functions like this that are used to judge when it's acceptable to alter the temperature in a home. One of our main features, time-to-temperature, is an estimation of time which allows the user to choose when to execute the event of starting the heating/cooling system.

Smart Audio System: **Control Flow**

Our system is event driven and mostly triggered by the user and/or other subsystems. Whenever the user activates the SAS by the Audio Functionality or other subsystems want transmit audio SAS is called upon to complete the function. Our system is not directly reliant on time but the SSS is and when a time event occurs on the SSS, the SAS is called upon to do a certain function from ringing the alarm to playing the user's schedule. For the tracking between user device and speaker location we will be using multiple forked processes for concurrency. Forking will be an optional expandability option for users who really want to scale up this system. The beauty of our system is that not every single machine will need to be upgraded; only the Indoor Positioning System's server.

Smart Scheduling System: **Control Flow**

Our system is driven based on user inputs. The alarm will sound base on the weather conditions, traffic conditions, the time for user to get ready, the time user have to be at work. User input his schedule, then the system fetch all related information, and finally it sets the alarms.

G. Hardware Requirements

For the SES system to work we need several types of sensors. For some of use cases we need to detect whether or not there are people in the area and we will use PIR sensors. The main features of our system require sensors that measure inside home temperatures and humidity. To get the temperature of the house a thermometers data will be needed. As well two machines that have web browsers.For the SAS system, we will need speakers which will be connected to Alexa and the Speaker controller.

Project Management for Part 2

This week we had refocused ourselves for a more code first outlook as well as reminding ourselves of the goal. After reading an expert from the group management paper: "The State of Cooperative learning in Postsecondary Settings" by David W. Johnson [12], we did an activity at our round table meeting where we asked two questions openly to the group. This was a useful exercise in which we were able to get on the same page. We also went back to the drawing board in terms of which technologies to use and how to code up the project. We meet together on friday night to work these ideas out as well as work on the report.

10. Algorithms and Data Structures

Algorithms

Smart Audio System: Algorithms

The SAS tracks the user's location in the home and automatically plays audio on speakers closest to him while also avoiding conflicts with other user's audio playing. To implement this feature the SAS will have to constantly gather user geo location(with respect to the mapped location). The user's phone or tablet will transmit this statistics to a computer(raspberry pi for our case). We do this by having listening sockets on the raspberry constantly listening to new user locations and each location will coordinate with a certain speaker. Our Indoor Positioning System will provide the user location and transmit that over the local area network to the raspberry pi. This will occur simultaneously.
The inefficiencies are apparent if the user wants to scale the SAS to include more users. To combat inefficiencies and allow for future expandability we plan to have sockets opened in concurrent forked processes for future expandability use only. However for a limited number of users, the user does not have to initiate a concurrent version of the SAS.

Smart Scheduling System: Algorithms

The SSS uses array index lookup in order to insert events into the schedule. If an event already exists at the specified array index, then the event creation fails. The SSS will fetch predicted traffic condition at a particular time from Google Maps API [3]. The time of the wakeup notification will be calculate by adding the traffic travel duration calculated by Google Maps [3] to the user's timeToGetReady attribute in the UserProfile object. In addition, the SSS uses a networking algorithm to get JSON and parse JSON data from Google.

Smart Energy System: Algorithms

For this first demo we plan on using an algorithm based on Matlab's Thermal House Model [9]. Home temperature data will be read in from temperature sensors and outside temperature will be retrieved from a weather API. This data will be used to calculate the approximate time it will take to reach the users preferred temperature if the HVAC was started at any given moment. The flow of heat will be calculated based on these two temperature and some general parameters similar to most HVAC systems. On top of that we will be incorporating Newton's Law of Cooling [10] which states that the heat transfer is proportional to the difference in temperatures.

Data Structures

In general, all systems will use a database. The question then becomes what datastcuters is the database using? In particular MySQL uses a combination of binary trees and hashtables. Because we are using a database the data structures are is abstracted away from us so we do not have to worry about updating.

Smart Audio System: Data Structures

The SAS will have to keep track of geo location points throughout the home. To store such data points we will use a hash table with chaining. So each geo location will getted mapped to a certain index in the hash table and each location in the table will correlate to a speakers in the vicinity of the geo location. However, given a smaller location we can make due with logic operators. Given more than one speakers in a single geolocation, speakers will be chained to the position of the table and audio will be played throughout all.

Smart Scheduling System: Data Structures

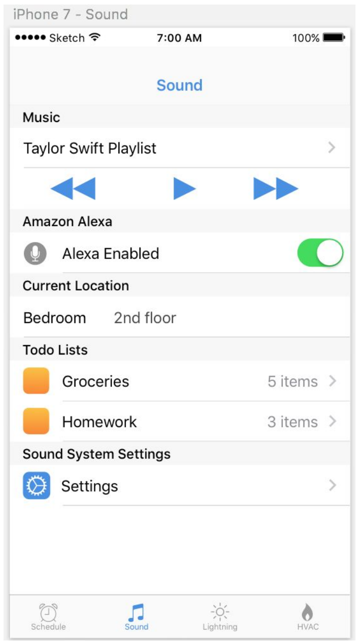
The SSS stores events as objects. The data is manipulated as strings since a Google Maps API call returns strings in a JSON format. These objects are saved in a public Schedule object, which contains an array of events.

Smart Energy System: Data Structures

Temperatures will be stored in vectors. This is because after every cycle the most recent temperatures will be added to the ends of the list dynamically. If we want to access temperatures a few cycles earlier the time complexity is O(1) whereas stacks, queues and linked lists are O(n). We won't be performing many other operations on the data besides dynamically inserting, deleting and and accessing data. Therefore, vectors are the most efficient.

11. User Interface Design and Implementation

11.1 Preliminary Design


iPhone 7 - Sound

●●●●● Sketch 7:00 AM 100%

Sound

Music

Taylor Swift Playlist

Amazon Alexa

Alexa Enabled

Current Location

Bedroom 2nd floor

Todo Lists

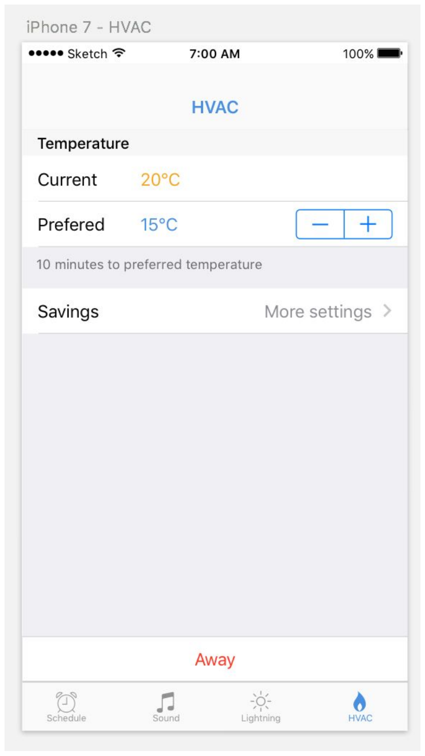
Groceries 5 items

Homework 3 items

Sound System Settings

Settings

To access everything for Sound, click on Sound in sidebar. Once in it you'll have all the options to do what you want with the SAS. Click on settings and you can access and modify all its settings and preferences there. Hit the back button to get back to the dashboard. That's one click to get to all the functions and two clicks to all the settings. The Settings is filled with simple to use toggle switches and character input for specific settings management. It also provides basic troubleshooting detail whether you are connected to the rest of the network.



Temperature

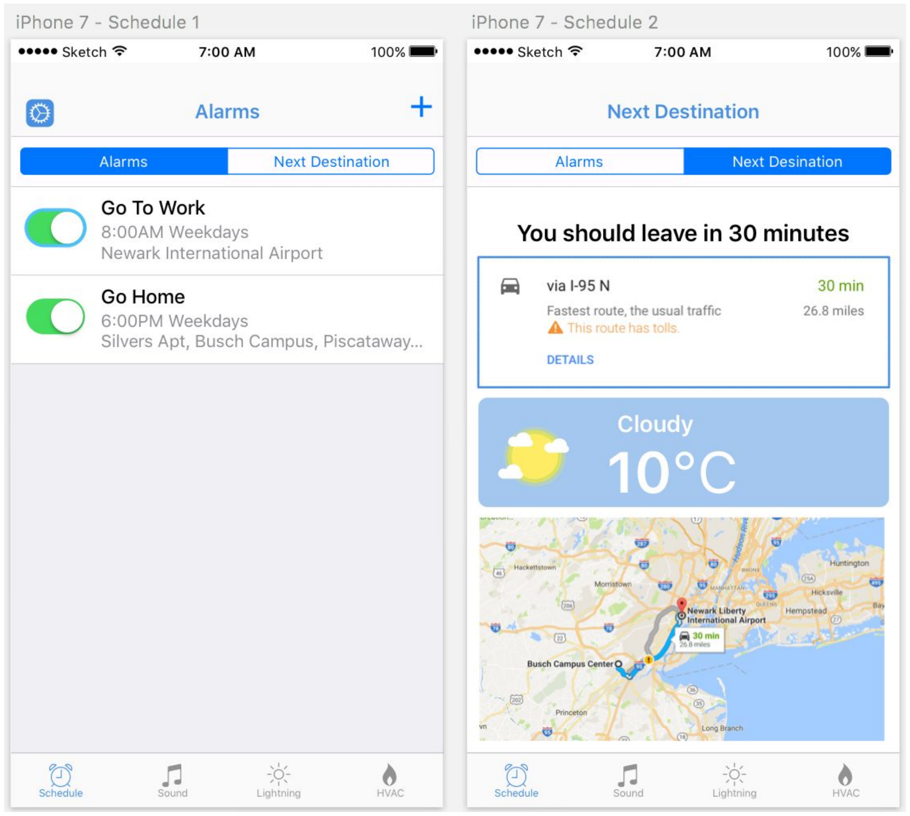
Current 20°C

Prefered 15°C

10 minutes to preferred temperature

Savings More settings

Based on use case 8 we can see what the GUI will look like. After opening the application the user would first click on the HVAC button. Once in HVAC the user will have options to edit their preferences. In this scenario the user would be heading home and want to have their house temperature be their preference on arrival. After clicking Time-to-temperature the user has the option to start the heating/cooling by pressing the start button or wait if their time-to-home is much larger than the time-to-temperature.


iPhone 7 - Schedule 1

●●●●● Sketch 7:00 AM 100%

Alarms

Alarms | Next Destination

Go To Work
8:00AM Weekdays
Newark International Airport

Go Home
6:00PM Weekdays
Silvers Apt, Busch Campus, Piscataway...

Schedule Sound Lightning HVAC

iPhone 7 - Schedule 2

●●●●● Sketch 7:00 AM 100%

Next Destination

Alarms | Next Desination

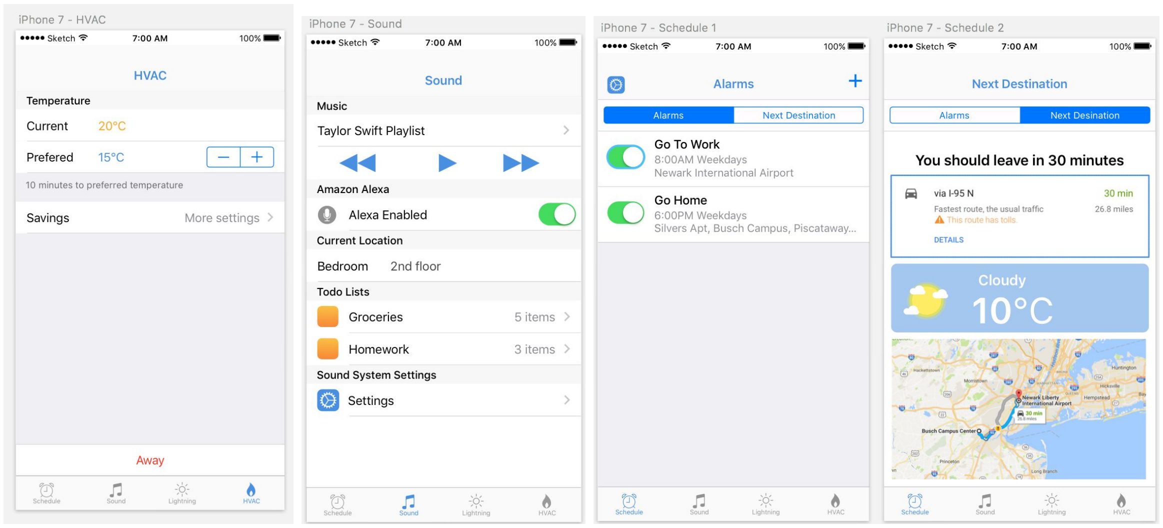
You should leave in 30 minutes

via I-95 N 30 min
Fastest route, the usual traffic 26.8 miles
⚠ This route has tolls.
DETAILS

Cloudy
10°C

Schedule Sound Lightning HVAC

Navigation Sample Mockup : https://gfycat.com/ScientificRingedAstrangiacoral


iPhone 7 - HVAC

●●●●● Sketch 7:00 AM 100%

HVAC

Temperature

Current 20°C

Prefered 15°C − +

10 minutes to preferred temperature

Savings More settings

Away

Schedule Sound Lightning HVAC

iPhone 7 - Sound

●●●●● Sketch 7:00 AM 100%

Sound

Music

Taylor Swift Playlist

Amazon Alexa

Alexa Enabled

Current Location

Bedroom 2nd floor

Todo Lists

Groceries 5 items

Homework 3 items

Sound System Settings

Settings

Schedule Sound Lightning HVAC

iPhone 7 - Schedule 1

●●●●● Sketch 7:00 AM 100%

Alarms

Alarms | Next Destination

Go To Work
8:00AM Weekdays
Newark International Airport

Go Home
6:00PM Weekdays
Silvers Apt, Busch Campus, Piscataway...

Schedule Sound Lightning HVAC

iPhone 7 - Schedule 2

●●●●● Sketch 7:00 AM 100%

Next Destination

Alarms | Next Desination

You should leave in 30 minutes

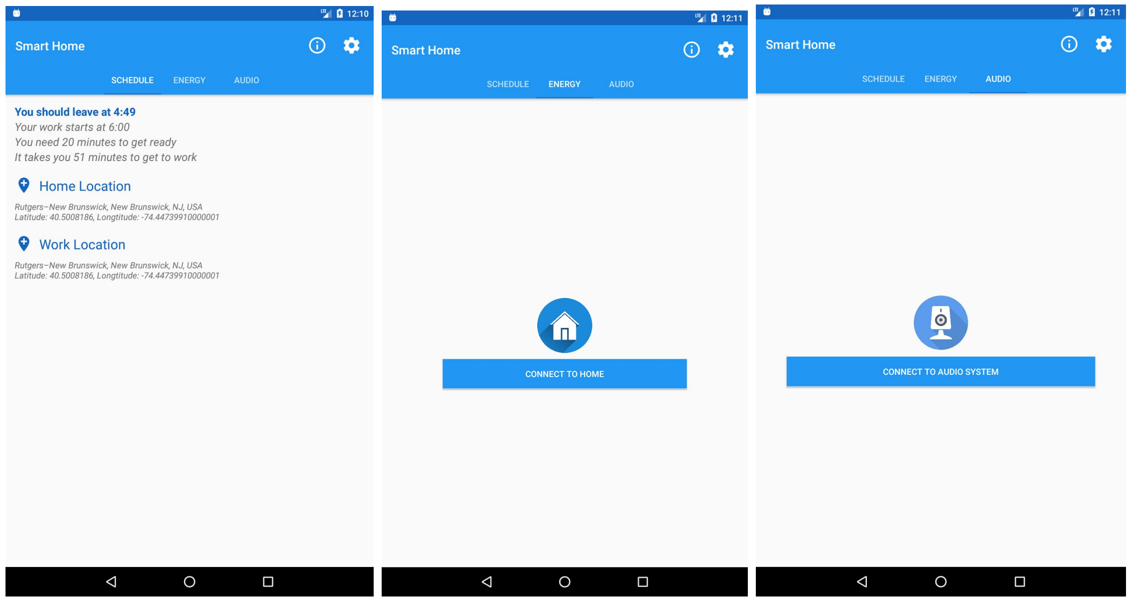
via I-95 N 30 min
Fastest route, the usual traffic 26.8 miles
⚠ This route has tolls.
DETAILS

Cloudy
10°C

Schedule Sound Lightning HVAC

We have not made any significant changes to our graphical user interface. However, we are working to streamline our interface to make operations as easy and concise as possible. We are making progress in eliminating technical jargon in the user interface for ease of use for the user. We want the user to take advantage of all our features and not get lost in technical jargon. We are going to implement settings option for each of our subsystem that the user can easily navigate through and understand what each option does with little or no research. Android[11] may look a little different.

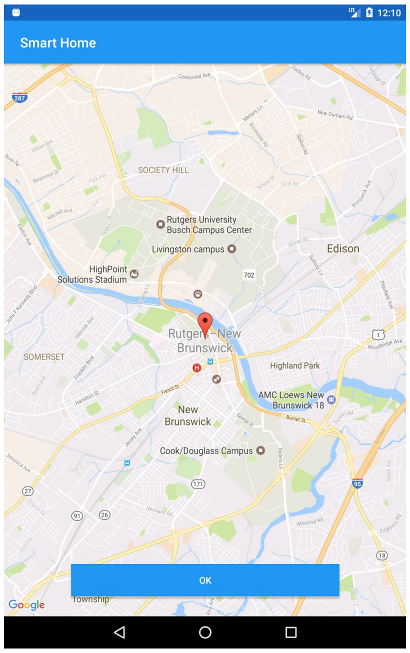
11.2 Secondary Design

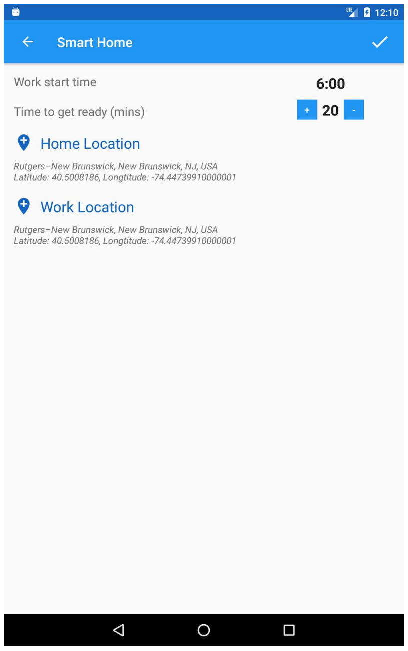


You should leave at 4:49
Your work starts at 6:00
You need 20 minutes to get ready
It takes you 51 minutes to get to work

Home Location
Rutgers–New Brunswick, New Brunswick, NJ, USA
Latitude: 40.5008186, Longtitude: -74.44739910000001

Work Location
Rutgers–New Brunswick, New Brunswick, NJ, USA
Latitude: 40.5008186, Longtitude: -74.44739910000001





Work start time 6:00

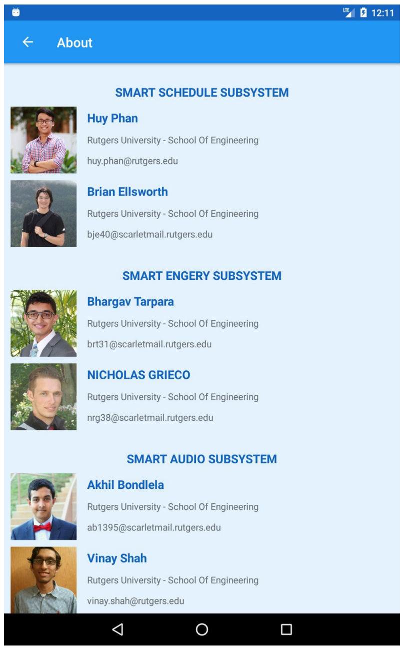
Time to get ready (mins) + 20 -

Home Location

Rutgers–New Brunswick, New Brunswick, NJ, USA
Latitude: 40.5008186, Longtitude: -74.44739910000001

Work Location

Rutgers–New Brunswick, New Brunswick, NJ, USA
Latitude: 40.5008186, Longtitude: -74.44739910000001



SMART SCHEDULE SUBSYSTEM

Huy Phan
Rutgers University - School Of Engineering
huy.phan@rutgers.edu

Brian Ellsworth
Rutgers University - School Of Engineering
bje40@scarletmail.rutgers.edu

SMART ENGERY SUBSYSTEM

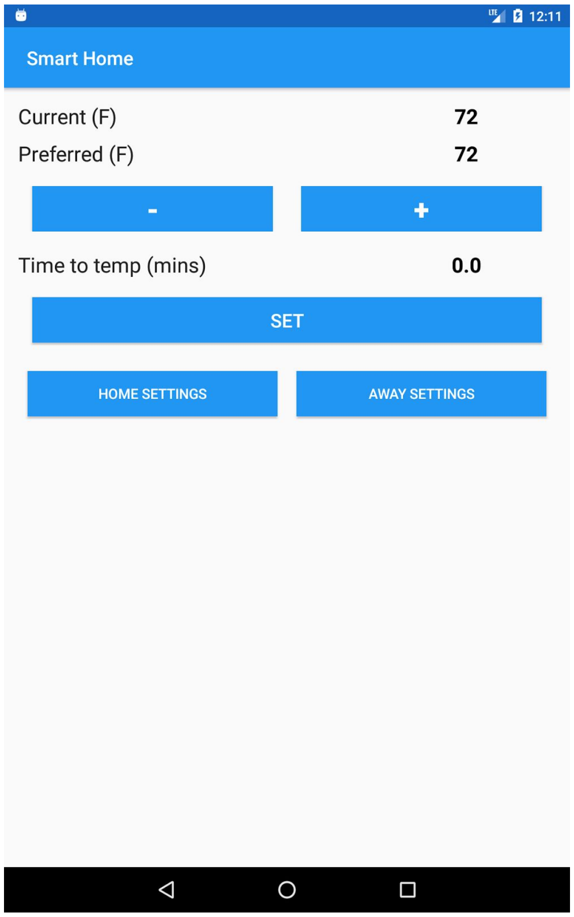
Bhargav Tarpara
Rutgers University - School Of Engineering
brt31@scarletmail.rutgers.edu

NICHOLAS GRIECO
Rutgers University - School Of Engineering
nrg38@scarletmail.rutgers.edu

SMART AUDIO SUBSYSTEM

Akhil Bondlela
Rutgers University - School Of Engineering
ab1395@scarletmail.rutgers.edu

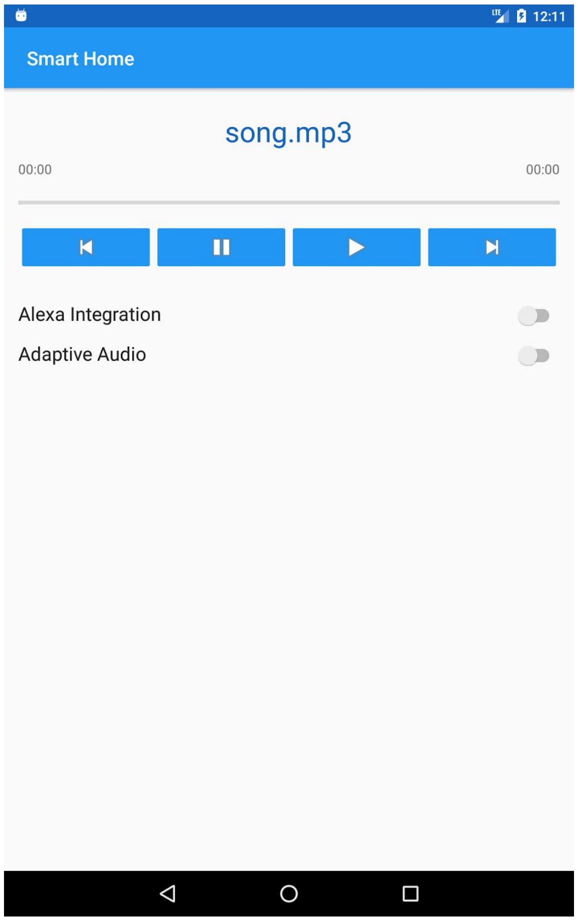
Vinay Shah
Rutgers University - School Of Engineering
vinay.shah@rutgers.edu



Current (F) 72

Preferred (F) 72

Time to temp (mins) 0.0



song.mp3

00:00 00:00

Alexa Integration

Adaptive Audio

12. Design of Tests

Casual Test Cases

Use Case 1: Todo List
Simulate a wake up alarm and check to see whether the SAS notices the wake up and plays the todo list

Use Case 2: Audio Room Selection
Attempt to change audio playback from one room to another manually while in the middle of a playback such as a song.

Use Case 3: Audio Selection Conflict Control
Attempt to play audio on a speaker where audio is already being played by another user.

Use Case 4: User Audio Tracking
Move location of the user's phone from the vicinity of one speaker to another and see if the audio playback transitions to new speakers while stops playing on the original

Use Case 5: Amazon Alexa Integration
Attempt to access Amazon Echo services with voice command and attempt to get a response through the speakers

Use Case 6: Speakers for SSS
Trigger other subsystems to attempt playback audio messages through speakers.

Use Case 7: Monitoring for SES
Check if log file is there.

Use Case 8: Temperature Ready
Check if time to temp is a positive number and that they match at some point.

Use Case 9: Dual Zone Inactivity
Run a test where sensor input is nothing in one zone and active in another.

Use Case 10: Activate Dual Zone
Check if the system responds to no activity

Use Case 11: Sensor Activation
Check if the system responds to activity

Use Case 12: Wake Up Adjustments
Check temperature at a given time point to see if it matches setting.

Use Case 13: Going to Bed Adjustments
Check temperature at a given time point to see if it matches setting.

Use Case 14: Empty Home
Check if button for empty home is a clickable object.

Use Case 15: Energy Saving
Check if graph is generated and function generating the graph is working.

Use Case 16: Schedule Initialization
When the user first uses the app, a schedule is imported from the user's local calendar application. The user should be able to add events initially to the schedule

Use Case 17: View Scheduling Factors
When the SSS provides a bedtime notification time, the user should be able to view the factors that led the SSS to make its decision. This includes viewing traffic data from Google Maps including anticipated travel duration.

Use Case 18: Schedule Modification
The user should be able to modify existing schedule events. This includes modifying parameters of each event: start time, end time, duration, location, estimated time of arrival, and temperature.

Use Case 19: Wake-up Notification
The user must be awoken at the time specified by the notification. The notification is set by assessing the Google Maps traffic data.

Use Case 20: Event Addition to Schedule
Events that the user enters into the application must be successfully saved into the schedule. Schedule conflicts must be handled and reported to the user.

Use Case 21: Bedtime Notification
The user must be provided with a notification at night so that he can go to bed at a time that allows him to wake up after a user-specified duration.

Descriptive Test Cases

Use Case 1: Todo List
An alarm wakeup is simulated with a filled todo list

Success
-SAS should be triggered by a wakeup
-SAS should playback todo list through the speakers

Failure
-SAS remains dormant during a wakeup event
-SAS is triggered by wakeup event but plays nothing from speakers

Use Case 2: Audio Room Selection
Audio playing on one speaker is changed to playback on another speaker

Success
-Playback stops on the first speaker and starts playing on the chosen feature

Failure
-Nothing happens
-Audio plays on both speakers
-Audio switches to wrong speakers

Use Case 3: Audio Selection Conflict Control
Simulate a conflict of audio playback

Success
-If conflict occurred due to movement playback on speakers is first come first served basis
-If user manually changes playback to speaker, user with highest priority is given playback access

Failure
-User priority is used to judge playback rights due to movement of user
-Manually changed playback conflict isn't resolved based on user priority

Use Case 4: User Audio Tracking
User walks from one speaker location to another during playback

Success
-Audio follows user as he/she walks from one speaker location to another while cutting playback

Failure
-Audio playback remains stationary as user moves.
-Delay of audio transition is arbitrarily long

Use Case 5: Amazon Alexa Integration
Activate Alexa

Success
-Alexa remains always listening and replies to voice commands

Failure
-Alexa does not capture most voice command
-Alexa transmits commands but doesn't reply back

Use Case 6: Speakers for SSS
Other subsystems to attempt playback audio messages through speakers

Success
-Other subsystems can triggers SAS and relay audio through speakers

Failure
-Other subsystems cannot communicate with SAS
-Other subsystems can associate to SAS but cannot play audio

Use Case 8: Temperature Ready
User presses button to initiate heating/cooling

Success
- HVAC system starts
- Time-to-temperature countdown is initiated

Failure
- HVAC fails to start

- Time-to-temperature stays static

Use Case - 9: Dual Zone Inactivity
No movement for a set period of time

Success
- HVAC decreases heating/cooling power
- Lights shut off

Failure
- HVAC stays at same power or is increased
- Lights stay on

Use Case - 10: Activate Dual Zone
Users notifies system to adjust lighting and temperature in a zone they are moving to preferred settings

Success
- Room requested will start time-to-temperature process
- Lights will turn on in respective area

Failure
- HVAC does not start
- Lights are not turned on

Use Case - 11: Sensor Activation
User's enter a room and lights are turned on if it is dark (low luminosity)

Success
- Light turn on

Failure
- Lights don't turn on when motion is detected
- Lights turn on when no motion is detected

Use Case - 12: Wake Up Adjustments
Time specified before user wakes up the SES initiates

Success
- Temperature is users preferred temperature when they wake up

- Lights gradually turn on when motion is detected

Failure

- Preferred temperature is not reached upon waking up
- Lights do not turn on

Use Case - 13: Going to Bed Adjustments

Temperature adjusts to energy efficient temperature around normal sleep hours. Lights turn off when user notifies the system through the application

Success

- Temperature is users preferred temperature when they go to sleep
- Lights turn off when the user indicates they are going to sleep and there is no/minimal movement

Failure

- Temperature is not what the user would prefer during sleep
- Lights do not turn off

Use Case - 14: Empty Home

User notifies the system they are leaving their home

Success

- HVAC decreases power to optimal temperature
- Lights shut off

Failure

- HVAC stays on or at inefficient levels after user left
- Lights stay on

Use Case - 15: Energy savings

User checks their energy savings for a given time duration

Success

- Energy savings are calculated
- Energy savings can be displayed on the app

Failure

- Energy savings not calculated properly
- Energy savings won’t show up on the app

Use Case 20: Event Addition to Schedule

Success:

-The user presses the “OK” button and the new event object is saved in the schedule object.

-The user presses the “cancel” button and the new event is not saved.

-Event data is saved properly.

-New events with notification or temperature settings should send a signal to the SAS or SES, respectively.

Failure:

- The user presses the “OK” button, but the new event object is not saved in the schedule object.
- The user presses the “cancel” button, but the new event is saved into the schedule.
- Event data is not saved properly. It appears as null or as invalid data values.
- New events do not properly signal the SAS to play a wakeup notification or bedtime notification.
- New events do not properly signal the SES to change the temperature for an event with a temperature setting.

Use Case 19: Wakeup Notification

Success:

- The SSS is unable to appropriately set the wakeup time based on the user data and the data from Google Maps.
- The user’s location is properly conveyed from an Event object to Google Maps.
- Google Maps returns an appropriate travel duration value.
- The SAS plays a notification sound at the time of the wakeup notification.

Failure:

- The SSS is able to appropriately set the wakeup time based on the user data and the data from Google Maps.
- The SAS does not play a notification sound at the time of the wakeup notification.
- The SSS is not able to communicate with Google Maps in order to receive a time estimate for the approximate travel duration.

-The SAS plays a notification sound at the wrong time.

Integration Testing

The test will simulate the event of the user waking up. The SSS will monitor the traffic and wake up the user at the optimal time. The SSS triggers wakeup events on other subsystems. The SES will then turn on the lights in the user's room and make the temperature optimal in the area where the user has breakfast. The SAS will play the alarm to wake the user and then read the user's todo list for the day.

Given Conditions

-User has a set breakfast location
-SHS knows of user's commute schedule
-User's todo list is filled with testable instructions

Success

-Wakeup was triggered on SSS with respect to traffic condition on commute
-Wakeup is triggered on SSS and SSS triggers wake up on SES and SAS
-SES adjusts climate appropriately in breakfast area shortly before impending event
-SAS rings wakeup alarm and plays todo list shortly after

Failure

-Nothing happens
-Wakeup event doesn't take in account of traffic condition
-Each subsystem doesn't perform their respective tasks
-Subsystem perform tasks at incorrect times and act disjointed

13. History of Work, Current Status, and Future work

A. Merging the Contributions from Individual Team Members

Subgroup 1: Akhilesh Bondlela, Vinay Shah
Subgroup 2: Brian Ellsworth, Huy Phan
Subgroup 3: Bhargav Tarpara, Nicholas Grieco
Subgroup 1 is responsible for coding the Smart Audio System.
Subgroup 2 is responsible for coding the Smart Scheduling System.
Subgroup 3 is responsible for coding the Smart Energy System.

The application was done in Java using Android Studio.

Each subgroup work on their system, we will merge everything into one Android Application, which acts as a control center.

B. Project Coordination and Progress Report

The use cases that will be implemented, at least for the first demo, will be the most critical to our functionality. Probably the most difficult endeavor we are on is figuring out how to implement the use case in actual code. Simple designs on the user's end has lead to a complex design on the backend. We are trying to tackle how to implement our use cases so that it can be implemented in the user's day to day operations and not create other complications of its own from it. Our user interface, as pasted above, has been successfully agreed upon. We believe we have developed an intuitive approach to use the SHS. Each subsystem has their own independent tab in the app sort of like a sub-app per subgroup.

C. Plan of Work

Task	2/26	3/5	3/12	3/19	3/26	4/2	4/9	4/16	4/23	4/30	5/7
Report 2 Part 1	🟧										
Report 2 Part 2		🟧									
Report 2 Part 3			🟧								
Report 3						🟧	🟧	🟧	🟧		
Demo 1		🟧	🟧	🟧	🟧						
Demo 2						🟧	🟧	🟧	🟧		
Archive Documentation										🟧	🟧
Server Backend Endpoints		🟧	🟧								
Database Connection to server via API		🟧	🟧	🟧	🟧						
General App UI							🟧	🟧	🟧		
SES frontend							🟧	🟧	🟧		
SAS Frontend							🟧	🟧	🟧		
Integration of Alexa API	🟧	🟧									
Integration of in home positioning API							🟧	🟧	🟧		
Prototype of an individual speaker with Alexa functionality	🟧										

D. Breakdown of Responsibilities

Name	Modules/Class Objects
Akhilesh Bondlela	SAS System Controller, Alexa Integration, Speaker Integration, Microphone Controller
Bhargav Tarpara	SES Controller, Main Activity, and Zone functions
Brian Ellsworth	SSS App Testing, Integrate SSS App to the Main App
Huy Phan	UI Design[13], SSS App (Coding, All classes)
Nicholas Grieco	SES App testing, Time to temperature function
Vinay Shah	SAS System Controller, User Location Tracker, System Logs, Schedule Interpreter

14. References

[1] 802.11n https://www.en.wikipedia.org/wiki/IEEE_802.11

[2] Amazon Alexa API https://developer.amazon.com/public/solutions/alexa/alexa-voice-service/content/avs-api-overview

[3] Google Maps APIs https://developers.google.com/maps/

[4] Internal Positioning(FIND) API https://www.internalpositioning.com/api/

[5] Operational Contract https://www.comptechdoc.org/independent/uml/begin/umlopcontract.html

[6] Nest https://nest.com/downloads/press/documents/thermal-model-hvac-white-paper.pdf

[7] Newtons law of cooling http://www.math.tamu.edu/~mvorobet/Math308/Fall07/Lecture18.pdf

[8] HVAC https://en.wikipedia.org/wiki/HVAC

[9] Matlab Sim https://www.mathworks.com/help/simulink/examples/thermal-model-of-a-house.html

[10] Newton's law of cooling https://en.wikipedia.org/wiki/Newton's_law_of_cooling

[11] Andriod https://developer.android.com/studio/index.html

[12] The State of Cooperative Learning http://biologytransformed.org/wp-content/uploads/2014/07/Johnson20071.pdf

[13] UI Design https://www.sketchapp.com