Team: DEC1716

Client: UAVX LLC

Adviser: Swamy Ponpandi

Team Members:

Marco Restaino (Webmaster)

Jackie Larin (Communications Leader)

William Tangney (Team Leader)

Team Email: dec1716@iastate.edu

Team Website: http://dec1716.sd.ece.iastate.edu/

Interactive TV Dashboard

Final Report

Contents

1 Introduction

This project was to create a dynamic and interactive TV dashboard on an HDMI dongle running an Android operating system. It integrated a large list of API's to present items to the dashboard such as calendar events, weather and commute time.

1.1 PURPOSE

The purpose of this project was to create a quality interactive assistant that runs on your TV. The assistant takes a number of commands from the users and executes them. We wanted to create a tool that can be useful and easy for everyone to use. Our client had the ultimate goal of selling this application to a commercial electronics provider to be included in some of their products. As a team our ultimate goal was to create an innovative application that someone would want to use everyday.

1.2 GOALS

For this project we as a group had two main goals, to create a quality and useful application and to gain knowledge for the future. When making this interactive TV dashboard we aimed towards making something really cool and useful but another big reward of the project was the knowledge we gained. We were excited to expand our knowledge of programming for android and learn about the plethora of technologies we incorporated into our project such as: Alexa voice control, CEC functionality, 3rd party APIs, etc.

1.3 DELIVERABLES

Our initial deliverables are listed below:

- Fast loading intuitive application
- Integration of hand tracking for control of the dashboard
- Integration of voice commands for control of the dashboard
- Application running on an HDMI dongle using the Android operating system
- Integration of 3rd party API's to deliver information from people's services like Google Maps
- Integration of HDMI CEC to have a "Wake Up" functionality for alarms
- Make secure system that is impossible to exploit

As we continued to work on the application our deliverables changed because we had to remove the hand tracking for the dashboard. For the rest of the deliverables we were to able to achieve them.

1.4 NEED STATEMENT/MARKET RESEARCH

According to a 2015 study regarding television statistics in U.S. households approximately 64% of households contain a television in the master bedroom of the home. We found this number to be quite staggering, as that is considerably over half of the homes in the United States contain a television in the primary bedroom. This is the market we are targeting, as utilizing the TV that is already in so many homes for the purposes of a lifestyle utility hasn't really been done yet. We feel that using the television as the primary interface for this system is different from other similar applications in the market and provides a very nice overall experience for the end user.

2 Design and Implementation

In the rest of this document, section 2 describes the design and implementation of the project in detail. Section 3 focuses on the design changes and challenges of the project. In particular section 3.1 highlights the major design changes during both semesters and how we overcame them. Section 4 describes how we tested our application as the semester progressed.

2.1 System specifications

The System runs on the Android OS running on an HDMI CEC ready media box. In the case of our project we used the Ugoos AM2 media box as provided by our client, it provided us with the HDMI CEC functionality and all other requirements that we needed to fulfill the scope of our project. This allows the user of the system to access our application with the primary interface being a television, the only requirement for that television is that it supports an HDMI connection.

2.2 Full Target Hardware Specifications

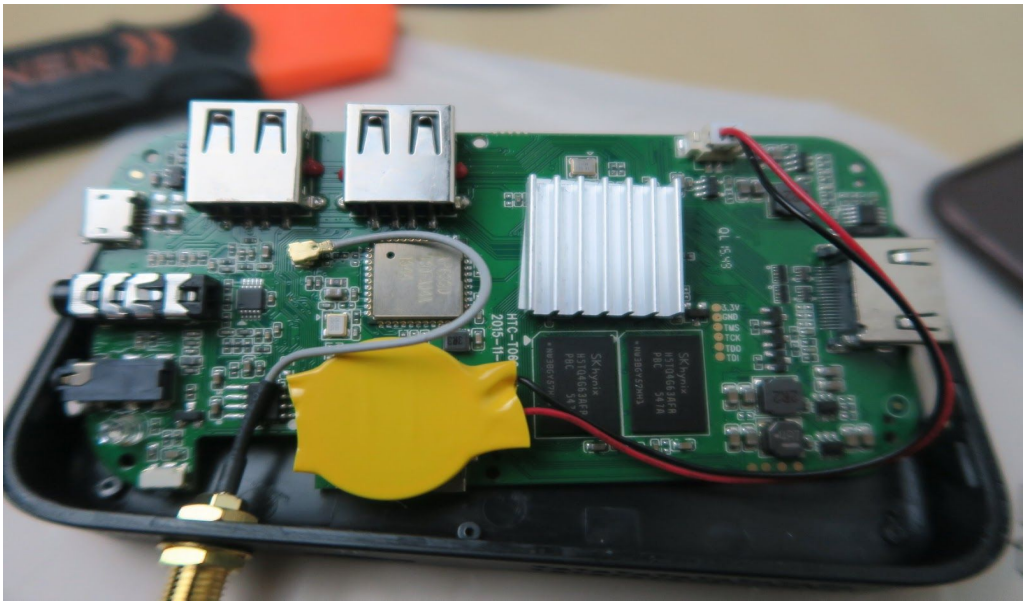

Figure 1

Operation System	Android 5.1.1
Language	Chinese,English...Multi-Language support
CPU	(S905)Quad-core ARM Cortex-A53 up to 2.0GHZ (DVFS)
GPU	Penta -core ARM Mali 450 GPU up to 750MHz+ (DVFS),OpenGL ES1.1/2.0, Open VG 1.1 , 2250Mpix/sec and 165Mtri/sec
SDRAM	DDR3 1GB
Flash	8GB
Network	2.4GHz/5GHz WiFi 802.11 b/g/n.
WIFI Module	AP6330
The extended storage	TF Card, up to 32GB
Power Supply	DC 5V/2.5A via Micro USB
Interface:	
HDMI	HDMI(1.4 and 2.0) to support maximum 4K@60fps output
Port	2*USB 2.0 Host, 1*IR Jack, 1* AV Jack
Power Supply	1*DC Micro USB

Figure 2 Hardware Specifications

2.3 Additional Hardware

We used an Amazon Alexa Dot to support the voice interaction functionalities of our system. A detailed image of the device can be seen below in *Figure 3*.

Figure 3 Amazon Alexa Dot

2.4 Interface specifications

Our client's main specifications our application were detailed in a screenshot provided to us. Our client had it mocked up by a graphic designer who works for the company and told us to work towards an end goal of having a similar looking UI with it's containing functionalities. The image the client provided us with can

be seen below in *Figure 4*.

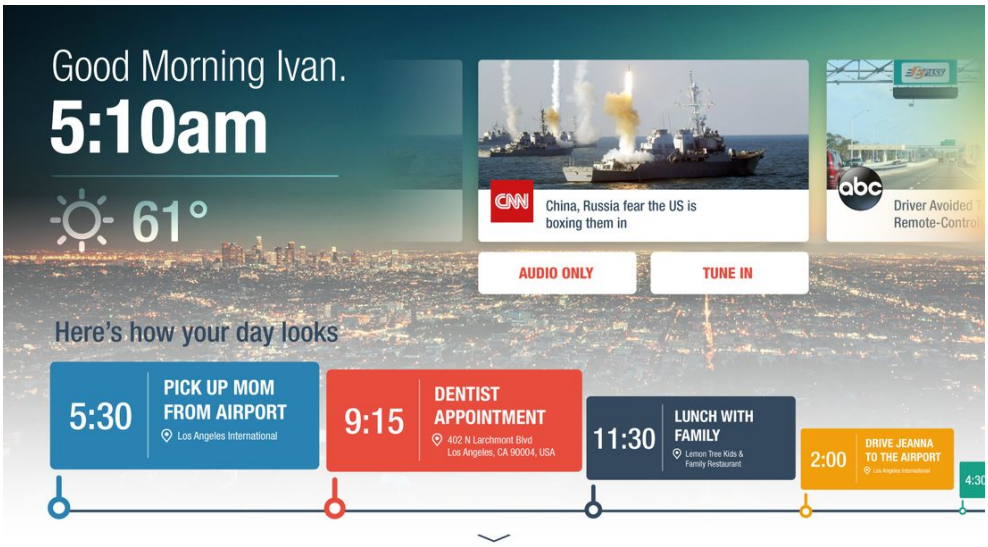

Figure 4 User Interface Design

2.5 Non-functional Requirements

- Clean and simple dashboard user interface
- Secure application
- Applications scales multiple types of Android devices and screen sizes
- Avoid legal and licensing issues
- Create an easy to use application

2.6 Functional Requirements

- Integrate multiple API components into single application
- Integrate voice commands to request information from the dashboard interface
- Integrate 3rd party APIs to deliver information to the user

2.7 Standards

In our code we integrated multiple APIs. These APIs are all licensed such that they can be integrated into an application that is released under either a proprietary or open source license. Piracy of code is strictly prohibited in applications developed under proprietary licenses and is a violation of all legal standards as well as standards set forth by IEEE and ABET.

2.8 Design Implementation Overview

The design is composed of multiple bound services acting as facades to the multiple APIs. For example the email service we provide a common base wrapper for all of the email APIs that will be integrated with this

design. The individual services will provide separate facades for different types of APIs to be integrated. There are specific wrapper classes so that all of the different APIs have a common core class. These classes are very flexible and provide a very simple interface to be used by higher level classes. This design gives a simple hierarchy for each of the classes to provide a high level common API for each of the services. The design is broken down into services, which contains services, and core which are the core wrappers. The alarm clock feature is already built into the framework so there is a package called manager which contains the alarm clock manager class. The manager classes manages all features already in the framework.

2.9 Implementation Details

For our project we have many different technologies and libraries being used in the background to ultimately provide the user with the simple interface and interaction. Our system can be divided into these major components and modules.

2.9.1 Android Main Activity

The Android main activity is where all of our information from the various major services come together. This is where we handle all the elements of the UI as well. This activity is where we combine all of our services to run them in the background and display the results to the user on the dashboard.

2.9.2 Major Information Services

We have four major information services in our system: calendar, weather, maps, and email. These are responsible for fetching the data from our various implemented APIs. In addition to these services we have one that updates all that retrieved and makes it accessible by the Alexa part of our system.

2.9.2.1 Google Maps

Google Maps was used as the service provider for all the mapping functionality of our application. Our implementation of this takes in two locations as parameters and calculate the commute time between the two locations via a car. This is used to display the commute time and route to the end user on the dashboard's user interface.

2.9.2.2 Google Calendar

The Calendar service is implemented on the local machine. It uses the user's synchronized google calendar through a CalendarProvider contract query. The query results came in the form of a cursor with the events that matched the query placed into a list to be viewed in the main activity in the main activity's onSelectedDayChanged function. Because the calendar is local, the response is instant, and the size of both the table and results are small, the calendar was queried without an async task or any threading implementations. Because all the user's calendars: google, local, and outlook, are synced locally there was no need for implementing multiple providers.

2.9.2.3 Google Email

Google Email was The email service is implemented with custom providers in mind. The email service has defined an interface for the custom providers to adhere. The custom providers are necessary because a developer may not assume that an email app, such as gmail, is installed on a local device. We only decided

to implement a google gmail provider because of the application's dependance on google services being available. This implementation provided a level of abstraction from the application and allowed the future development of email client implementations. The gmail provider itself uses multiple check functions to insure its availability on the system. It checks for the installation of google play services, network connectivity, user permission to access their email account, and API authentication from google. The provider the queries google's RESTful gmail api which will return a list of IDs of gmail labels, that represent inboxes, and gmail messages, that represent messages. The API must be queried a second time with the IDs and the user name to retrieve the specific values of the the remainder of the email fields. These results are put into a list and parsed in the onPostExecute function of the of the MakeRequestTask, an implementation of an async task. The provider makes a callback to the service and the service will make a callback to the main activity updating the UI's email list and showing the data retrieved.

2.9.2.4 Yahoo Weather

For the weather service we had to use the Yahoo weather API. Our initial plan was to integrate the Google weather API to have exclusively Google services throughout the project, but after some research last semester we discovered that the Google API is no longer available for use. We are able to get information from this service via the "YQL" query structure. From there the API returns a JSON object that we then parse to grab the relevant forecast information that we need to present.

2.9.2.5 Alexa Accessible Data Updating

In addition to the services we use to obtain data, we also have a service class that is used for the purposes of updating the data our Alexa skill presents. This functionality is detailed further in section 2.9.4 where we talk about the implementation details of the Alexa part of our system.

2.9.3 HDMI CEC

The HDMI CEC functionality of our project is responsible for controlling the television from interaction with the user. This feature sends a CEC signal via HDMI connection to a television that can receive those signals. This was implemented by adding a call to send the "TV Power On" signal to the TV when we had an alarm trigger in our system.

2.9.4 Amazon Alexa

Our system utilizes multiple tools within Amazon Web Services and Amazon Alexa Services to deliver the information from our application to the user via a voice response. We created an Alexa Skill using the Alexa Skills Kit to trigger our Amazon Web Services lambda function. This function is written in Node.js and it is responsible from getting the data we have stored in the AWS DynamoDB service and creating an audible output for the user. The other portion of the Alexa integration is the information service class in our Android application, this is responsible for updating the data in AWS DynamoDB so that the updated information can be retrieved by the lambda function. This ensures that what you see on the screen and the information read back to you from the Alexa device is always the same and current. In addition to these functionalities, the Alexa also handles authentication between our Android application and Alexa software utilizing the Amazon Cognito and Amazon IAM services.

2.9.5 System Diagram

If you look at *Figure 5* you will see a diagram that shows how each component listed above will interact within the system.

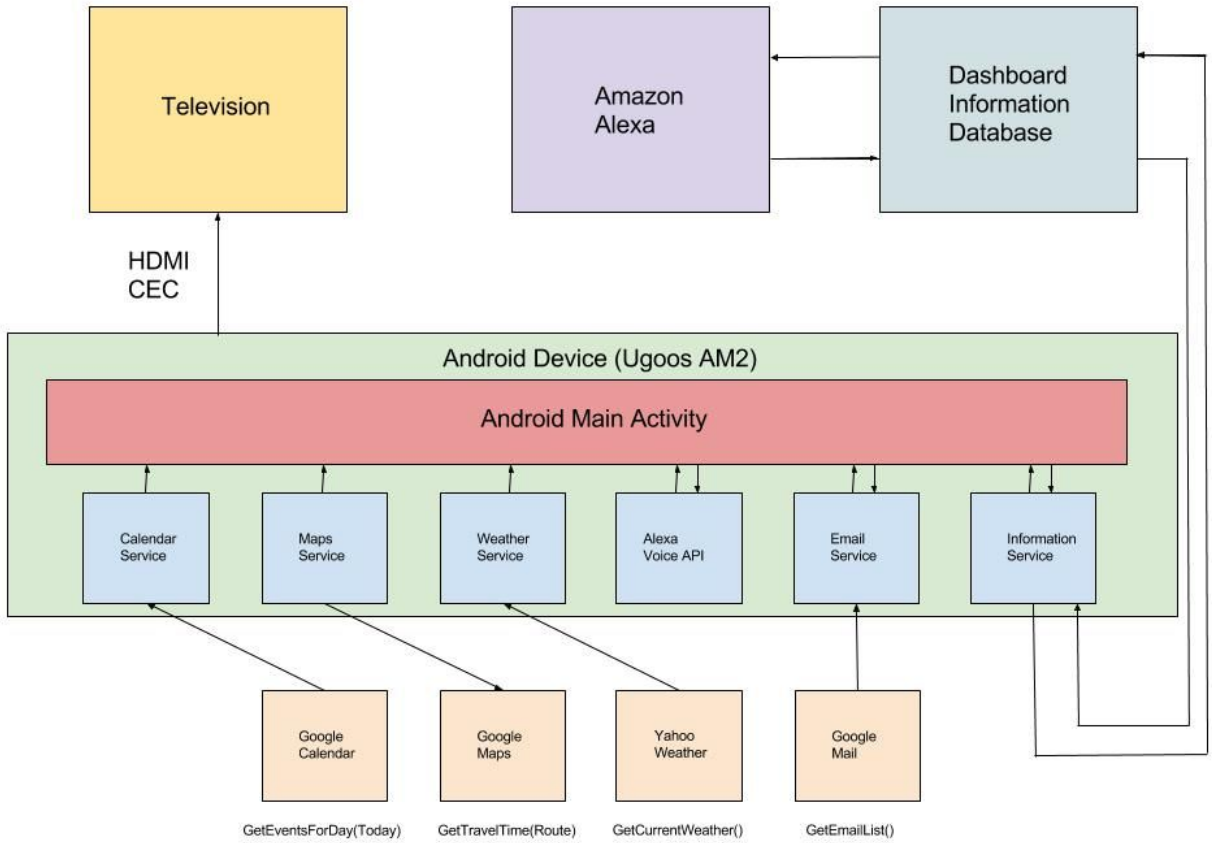

Figure 5 System Diagram

2.9.5 INFORMATION SERVICE METHODS

getEventsForDay() - Sample events will be prepared in the calendar service and the list of events will be compared against the list from the calendar API.

getTravelTimeforRoute() - The calculated route will have to be checked against that of the API that is used to call or other APIs to validate its accuracy within a certain tolerance. The shortest route heuristics can sometimes produce different answers so it's best to compare within a tolerance.

getCurrentWeather() - Check the current weather with calls from other weather reports. Compare within a tolerance. It can also be checked against a digital thermometer for temperature.

getEmailList() - Check if the retrieved list of emails matches that of the user's email client.

updateInformation() - Updates the information presented to the user from the dashboard to the Amazon

services that allow the Alexa to access it.

2.9.6 Our System User Interface

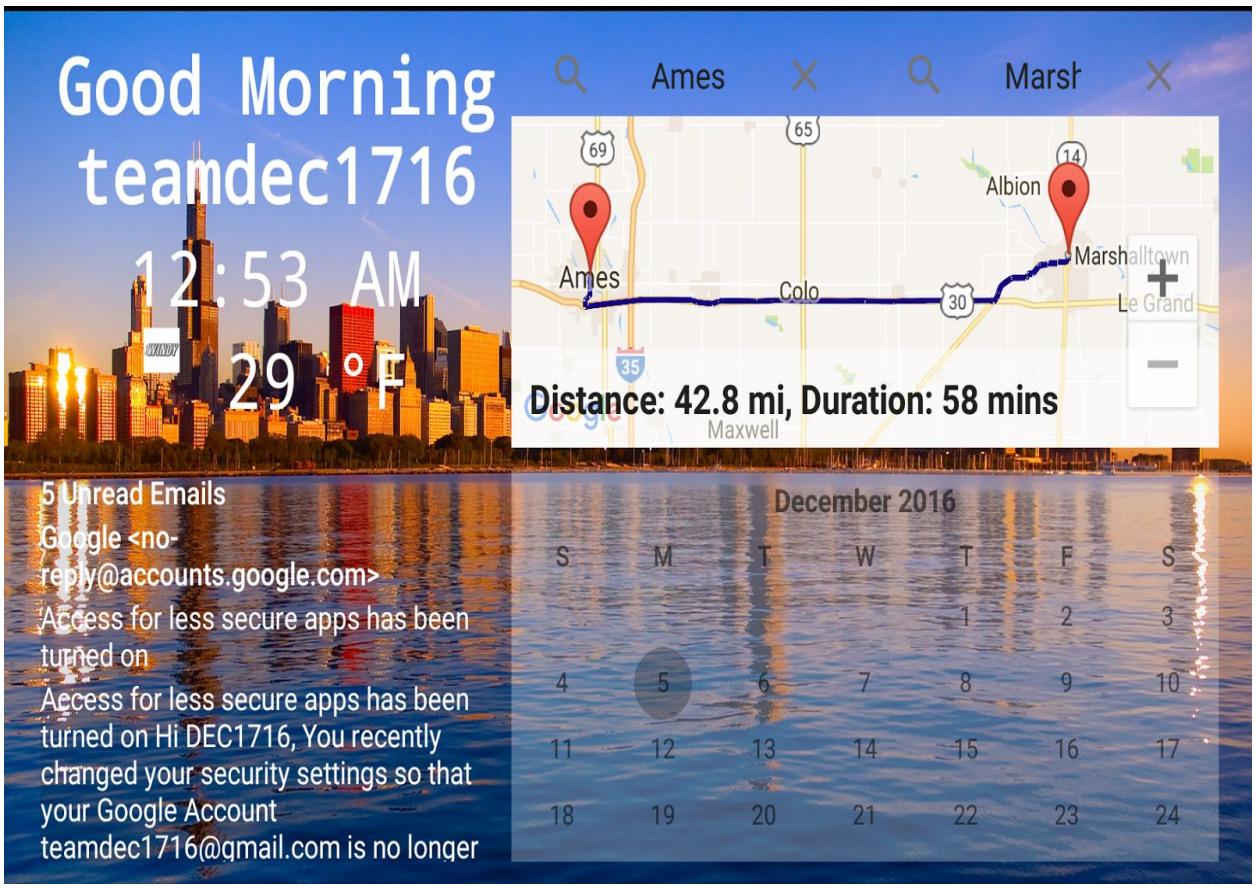

Figure 6 Interactive TV Dashboard User Interface

3 Design Challenges and Changes

The software design phase is an essential phase of the development life-cycle. The challenges we faced prompted us to play closer attention to the design process to better understand and apply well known design processes and professional practices to overcome the challenges listed below. As engineers nothing will go as planned so we have to learn how to move forward when faced with challenges and changes.

3.1 Design Challenges

For this project we faced many design challenges along the way, some of the main challenges that we had to overcome were:

- The target hardware that we were provided with from our client had it's fair share of hardware and software limitations which lead to some design challenges and even lead to us have to make some overall design changes to accommodate. We had to scrap the hand tracking functionality of our project that was initially proposed because after some testing we realized the hardware really could not handle that functionality from a performance standpoint, this was not too big of a setback for

the project because at that point in our development we decided that particular accessibility option was not needed to fulfill the goals of our project.

- We had many design challenges in the realm of getting everything to work well together. What that means is we had many APIs pulling in different information to the dashboard at once and there were plenty of challenges for us as relatively inexperienced Android developers to overcome.
- As time went on in our project our client became very busy in his personal life and lost the ability to meet with our team on a regular basis, this was difficult from a design standpoint because we had much less of the client's guidance. This made it harder to stay organized and make sure we were fulfilling what the client requested.
- The major design challenges we faced throughout the entire development process of our project were mainly associated to the lack of knowledge on the technologies used. Most, if not all of the system and services we used and integrated were completely new for the team and required a lot of time and research into them to properly utilize them in our application. One major example of this is the implementation of the Alexa services to our application, our plans for this system with our lack of experience turned out to be infeasible and the actual implementation would have to be much different. The Alexa services are very strict on how you are able to use their system, this lead to us having to learn a lot about all of the Amazon Web Services and how they all needed to combine to provide us with the functionality we needed for our application. This major design change/challenge is detailed in the next section, 3.2.

3.2 Previous Iterations and Design Changes

We had to make some pretty serious design changes as we were going through our development of this project, the hardware and software limitations that we were presented with caused many issues for some of the design elements we had originally planned for. In addition, there were some design decisions we made as a team and with our client to increase the overall effectiveness of our application. In previous design iterations we had planned for the implementation of hand tracking to control the mouse driver in our application. There were two main reason we decided against this inclusion and moved on to new design iterations with it removed. The first reason being, after doing some initial testing we discovered the board we were provided to run our software was simply not going to be capable of this type of accessibility. The second reason is our main reason for removing it, that was the fact that for the scope of our project, which is a simple easy to use dashboard, we felt that hand tracking would make it much more complex and frustrating to control. We were confident when making the serious design change that the voice interaction system would be a more effective means of interactivity with the system, because it is more simple and lightweight which is on theme with our goals for the project. The most major design changes for our system definitely came from the Alexa services that were implemented as part of it. None of us had any experience with this technology so when we designed the "ideal" way the Alexa would interact with our system we did not realize that the way we had planned wasn't really feasible in practice. First, we had planned for the Alexa device to be directly interfaced with our application and that we would able to implement an Android Alexa API to create the functionalities for our application. It turned out that the type of interfacing we had in mind was simply not available or possible. This lead us to a design change where we realized we would need a data layer between our application and our Alexa to "interface" with our system. That change lead to us attempting to implement a SQL server to update and store information being pulled into our application. This too ended up being very infeasible, within all of Amazon Web Services including the Alexa services they only really play nice with each other. What that means is trying to integrate an outside system within an Alexa function turned out to be very difficult and also seemingly not possible. All of those

changes to the Alexa integration eventually lead to our final design which was to use all Amazon services for the data handling our system, including the use of Amazon DynamoDB, Amazon Cognito, Amazon IAM, Alexa Skills Kit, and AWS Android SDK. The final design choice of using all these Amazon services to work together to provide the service we need for application was a major challenge and took a large amount of time to fully understand how all these things needed to work. In the end we do feel as though the implementation of all these services together did make the Alexa portion of the project work very well and ultimately provide the best and most refined solution for our goals.

4 Testing Process/Results

Testing is required for an effective performance of software application. In the sections below we talk about how we tested our application. We wanted to ensure that the application didn't result in any failures.

4.1 Hardware/software

The software we used for this project is the Android OS, so any type of hardware we tested with has been loaded with that OS. To begin our project we started testing with Raspberry Pi's that we loaded Android onto. This was just a placeholder until we all decided on the hardware our final version runs on, which we just received. The new hardware that our final product runs on is a Android 5.1.1 media box that has the CEC functionality of HDMI (Ugoos Am2).

4.2 Major Services General Testing

- Email Services and Providers:
  - Compare list of Emails to list of emails directly from 3rd party app. I.e. Compare gmail list to list from gmail app.
- Route planning:
  - Compare to other heuristics. This can only be tested in a tolerance because brute force shortest path on a map takes a significantly long time.
- Weather:
  - Compare to another weather service and compare weather data within a tolerance.
- Calendar Events:
  - Add events to google calendar and make sure all events are accurately retrieved.

4.3 Json Data Testing

For this project we incorporated a number of APIs, with the standard of most APIs returning data in the form of JSON this was a big part of the testing for our system. When we were integrating these services into our system, to test to make sure our API calls were returning the correct JSON data we needed for our functionalities we used systems in place to check our API query responses. For example, when we were building out the weather service functionalities of the system we used the Yahoo weather API. This API has a console to get sample responses from queries in the form of JSON, we used this system to test our query structure and make sure that the API calls from within our system are obtaining the correct data. The system we used to test our queries can be seen in *Figure 6*.

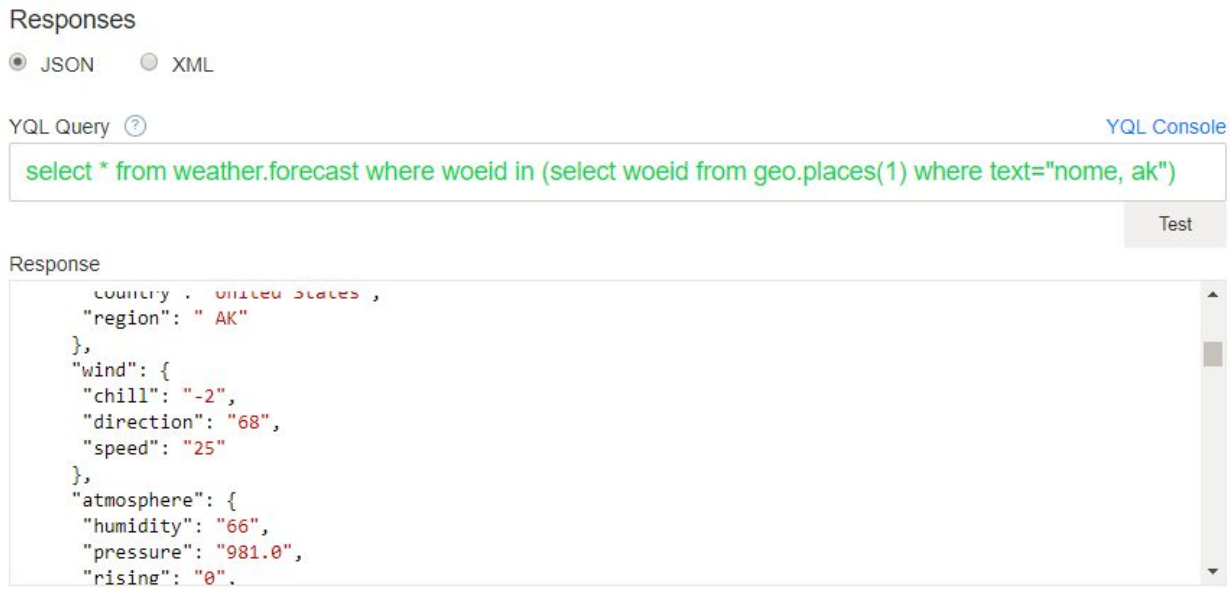

Figure 7 Weather Query Test

5 Related Products/Literature

In order to see the importance of our project, we decided to do research on the topic of Home Automation. We wanted to see what other projects or research is being developed that closely relates to our project. With the research we were able to see different approaches people are taking to develop something similar to our project and we were able to see the market for application.

5.1 Open Smart Hub : Home Automation

The Open Smart Hub project is a device that allows you to interface with various smart home type devices, such as smart switches. There were two majors features of this system that were of interest to our project and drew us to look further into it, those were the integration of a weather API and the use of an Amazon Alexa device for voice interaction with the system. These are two keys features we are including in our application so we thought this project overview would be great to give us some insight for our project development. In addition to the the two major features, there was also some integration with alarm related event triggering which is a major feature of our application and helped us with some ideas for ours. The commonalities between our project and the Open Smart Hub project are clear, therefore we had the thought that the potential benefits from researching it would be there.

The project documentation includes two major components, the hackster.io page and the GitHub for the project. The hackster.io page includes some diagrams and a good overall general overview of the system while the GitHub provides some more detailed comments on specific aspects of the system. Looking at the

hackster.io page gave us access to the diagram of the system the developer of the Open Smart Hub project provided with the general overview of the system. This diagram was really great to see the interconnections for the components of the system, it definitely helped us in the way we were thinking of structuring our system. This overview also gave us insight as to what weather API the Open Smart Hub project had used, in this case it was the WeatherUnderground API that was used to deliver forecast information to the system. This was one of the weather APIs that was researched before we chose which API we were going to integrate. The GitHub was something we also looked at for things to consider for our application. The GitHub provided the look into the functionality of the clock and event triggering functionality for the Open Smart Hub system, this helped guide us in our designs for our system, we did have to incorporate the extension to this with the CEC functionality of our system. Lastly, this project gave us a little bit of insight on the integration of the Alexa services for our system, this is something that we still have in development as no one in the group has experience with the Alexa API. This system provided us with a good "overview" of how the Alexa should integrate into one of these types of system. Overall, we think that this project provided useful information that was very helpful in understanding how we should go about building our system.

The Open Smart Hub project left us with a great deal of information that is and was helpful for the development of our project, in addition to that, it got us thinking about things we would be able to integrate in the future, such as smart switches. Our system is designed to be an alarm clock and we have built functionalities to go along with that, one thing that this project us got us thinking about for adding more to the project was the possibility of turning on lights to go alongside the alarm clock functionality. We feel as we are coming to the late stages of development of our project that the Open Smart Hub project gave us the knowledge and ideas to approach the technological hurdles of our project development more effectively and leave us with ideas for more possible additions to our existing application.

5.2 Bluetooth Remote Home Automation system using Android Application

This research paper discusses the design and implementation of a system intended to control electrical appliances and devices in house. The application allows the user to turn on and off light switches as well as measure temperature and humidity level in the house. One of the main purposes of this article was to create a low-cost design and a user-friendly interface that can be installed easily. These main purposes are exactly what we want to do for our project so it was interesting to read about their approach. The interesting thing about their approach was that they used Bluetooth wireless connection to enable the system to communicate with the graphical user interface on a laptop or smartphone. This allows them to not have to use cables which could have been a good idea for our project. Their approach with a complete success and in the future the window GUI will be implemented with speech recognition voice control. Our design will contain basic voice recognition commands which we are implementing by using an amazon Alexa.

5.3 Smart Home: Integrating Internet of Things with Web Services and Cloud Computing

This research paper explores the concept of smart home through integrating IoT with web services and cloud computing. Their system architecture must fulfill the requirements of measuring home conditions, processing instrumented data, and monitoring home appliances. Their approach Our approach consisted of

embedding intelligence into sensors and actuators using Arduino platform, networking smart things using Zigbee technology. Their approach is very different from ours we are creating an android application with a series of different services and using CEC as the protocol for sending events from a device to the television.

5.4 Home Automation and Security System using Android ADK

This paper puts forward the design where home appliances are connected to the ADK and communication is established between the ADK and android mobile device. Their application's main functions are light control, door control, smoke detection and temperature sensing. Their system is smart because it can activate the alarm when smoke is detected and auto on/off lights during late hours and it even has voice navigation. The user interacts with the android phone and send control signals to the Arduino ADK which in turn will control other embedded devices/sensors which is a little similar to our design.

Appendix 1: Operation

This is a guide on how to setup and use our application "Interactive TV Dashboard".

A1.1 Download Android Project and Environment

Our project can be downloaded with permissions of UAVX at:

https://bitbucket.org/interactivedashboard/code-repository

Once the project is downloaded you have access to our project which can be viewed, built, and tested from with the Android studio IDE which can be found here:

https://developer.android.com/studio/index.html

A1.2 Build and Run project on targeted device

Once you have access to our project in Android studio you can build/run it on our targeted device. This just requires that you connect the device, via a USB connection, to the computer and within Android studio select that device as the build target. This will then build and install the APK file on the device which gives you access to the application.

A1.3 Alexa Commands

All of the Alexa functionalities are accessed through our Alexa Skill "Interactive TV Dashboard", because the publishing rules for Alexa skills are strict we were not able to publish the skill. This means we only have the skill accessible through one of our team members Amazon account. This is something we hope to be able to publish at some point in the future to make it more accessible.

All of the interactivity of our system can be handled using Alexa commands. You can use this service and device to get audible readouts of the information presented on screen.

Once you have our Alexa Skill loaded up on an Alexa device you can start to use it with our system. The invocation name for our Alexa skill is, TV. That means that to invoke a response from our system you can say “Alexa, ask TV…”. Following that invocation of our system there are a list of intents to access certain responses from our system. They can be seen in *Figure 7.*

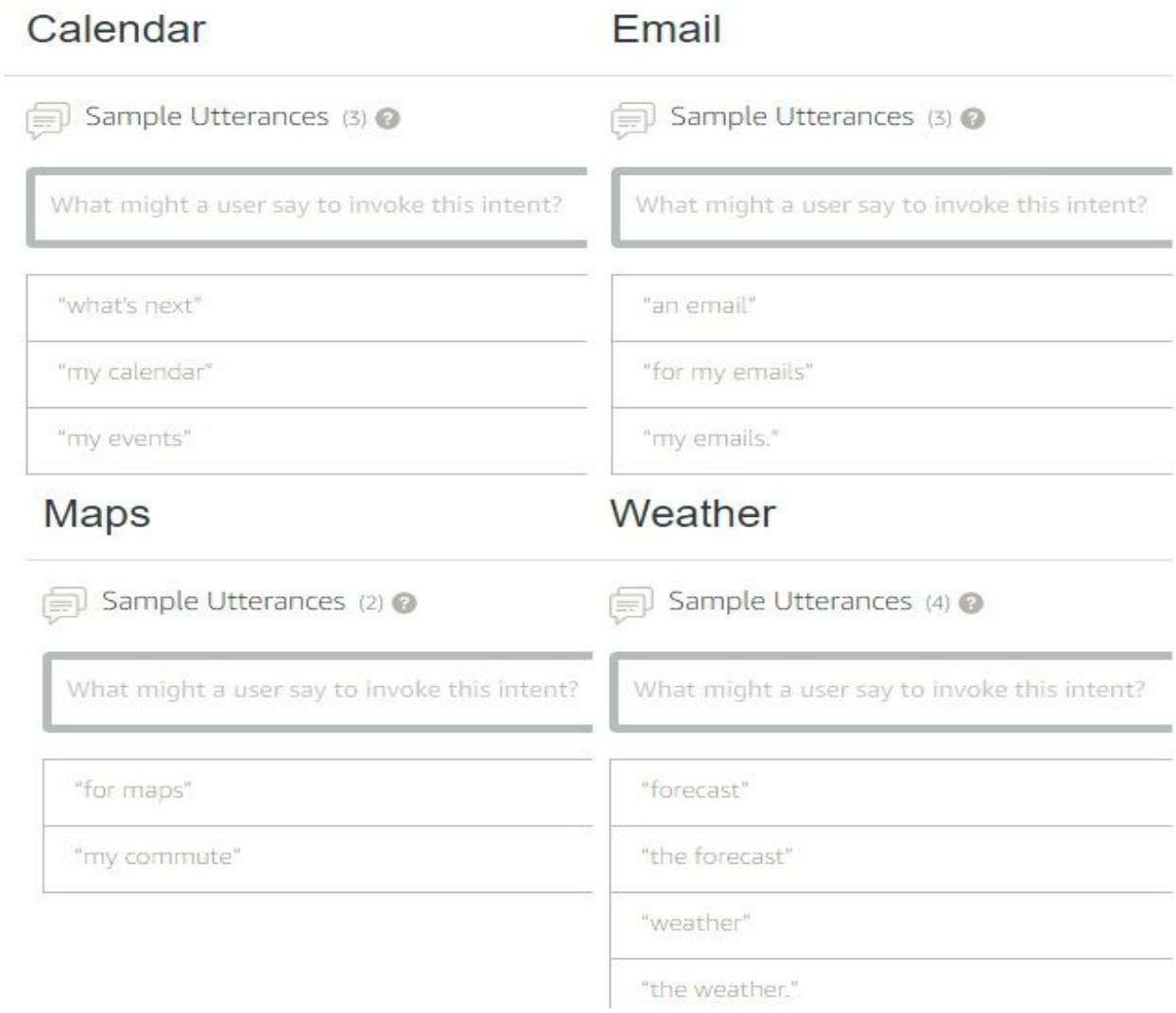

Figure 8 Alexa Skill Intents

A1.4 User Interface

From here all of the data will be presented to you on the user interface, this is pulled from your Google account so you need to designate the Google account that is associated with the android device it is running on. This option will pop up as a dialog when you first start the dashboard. There is no further instruction for use as all the other interactions with the user interface of the application are user friendly and logical.

Appendix 2: Conclusion/Other Considerations

This project was certainly a learning experience for our group in many ways, not only did we have to gain a great deal of knowledge with the technologies we used but we also got to gain exposure into the full design process of a project from start to end. This includes the documentation and meetings and all the things associated with a project that requires this level of design, we felt this to be the most useful thing we gained from going through the process of this project. While the technical knowledge and skills we obtained in the process are certainly important, we feel that the long-lasting positive impact this will have on us as engineers is going through the full design process. There were certainly long frustrating nights working on the project, but those were always followed but moments of triumph that left us with something to be proud of. Overall, through all the highs and lows that we experienced in the design and development of this project we feel like we came out as more capable and experienced engineers on the other side.

Appendix 3: Future Work

There are plenty of things that we are thinking about for future work that did not fall within the scope of our project or our time constraints for this project. One major thing we think could be a good addition to the system in the future is the addition of more interactions with the Alexa system. Our current system is designed in a way to allow for the additions of more Alexa interactions fairly easily, so this is something in the future that can be added to in order to create an even better overall user experience. The idea of customizable dashboard elements is something that was also appealing to us and something we think would be good addition to the system to make it more customizable for the end user. Overall, from all the research we did on smart home and home automation left us with plenty of ideas for future work on this system. We designed our current application with modularity in mind and making sure to document well, so the additions of these future functionalities will be able to be integrated without any issues. The ultimate goal for this type of application would be for it to get as close to a smart home type product as possible while still falling within the scope of the system.

Appendix 4: References

https://www.hackster.io/anthony-ngu/open-source-home-hub-26e5c7

https://source.android.com/devices/tv/HDMI-CEC.html

https://developers.google.com/gmail/api/guides/

https://developers.google.com/gmail/api/quickstart/android

https://github.com/zh-wang/YWeatherGetter4a

https://developers.google.com/maps/documentation/android-api/

https://developers.google.com/maps/documentation/directions/

http://www.sciencedirect.com/science/article/pii/S1877705811018911

http://www.creditdonkey.com/television-statistics.html

http://ieeexplore.ieee.org/document/6574587/

http://ieeexplore.ieee.org/document/6735443/