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

Design Document

Contents

1 Introduction

1.1 Project statement

This project is to create a dynamic and interactive TV dashboard on an HDMI dongle running an Android operating system. We will incorporate the CEC functionality of HDMI to turn your tv on and alert you for the purposes of an alarm clock. It will integrate a large list of API's to present items to the dashboard such as calendar events, weather, commute time, etc.

1.2 purpose

The purpose of this project is to create a commercial quality interactive assistant that runs on your TV. The assistant takes a number of commands from the users and executes them. We want to create a tool that can be useful and easy for everyone to use. Our client has 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 is to create an innovative application that someone would want to use everyday.

1.3 Goals

For this project we as a group have two main goals, to create a quality and useful application and to gain knowledge for the future. When making this interactive TV dashboard we will certainly be aiming towards making something really cool and useful but another big reward of the project will be the knowledge gained. We are excited to expand our knowledge of programming for android and learn about the plethora of technologies we will be incorporating into our project such as: voice control, hand tracking, 3rd party APIs, etc.

2 Deliverables

- 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

3 Design

3.1 System specifications

System is to run on the Android OS running on an HDMI CEC ready media box.

3.1.1 Non-functional

Non-functional Requirements

- Clean and simple dashboard user interface.
- Secure application, impossible to exploit.
- Applications scales multiple types of Android devices and screen sizes.
- Avoid legal and licensing issues.

3.1.2 Functional

Functional Requirements

- Integrate multiple API components into single application.
- Hand tracking control of the dashboard.
- Integrate voice commands to dashboard interface.
- Integrate 3rd party APIs to deliver information to the user.

3.1.3 Standards

In our code we will be integrating 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.

3.2 PROPOSED DESIGN/METHOD

The design will be composed of multiple bound services acting as facades to the multiple APIs. For example the email service will be able to 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.

3.3 DESIGN ANALYSIS

So far the basic services and core wrapper classes are designed. 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 currently 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 will be a package called manager which will contain the alarm clock manager class. The manager classes will be managing all features already in the framework.

4 Testing/Development

4.1 INTERFACE SPECIFICATIONS

Our client provided us with a concept screenshot of the application. He wants us to work towards designing an end user interface similar the image below. Outside of that there were no further exact specifications given regarding the interface.

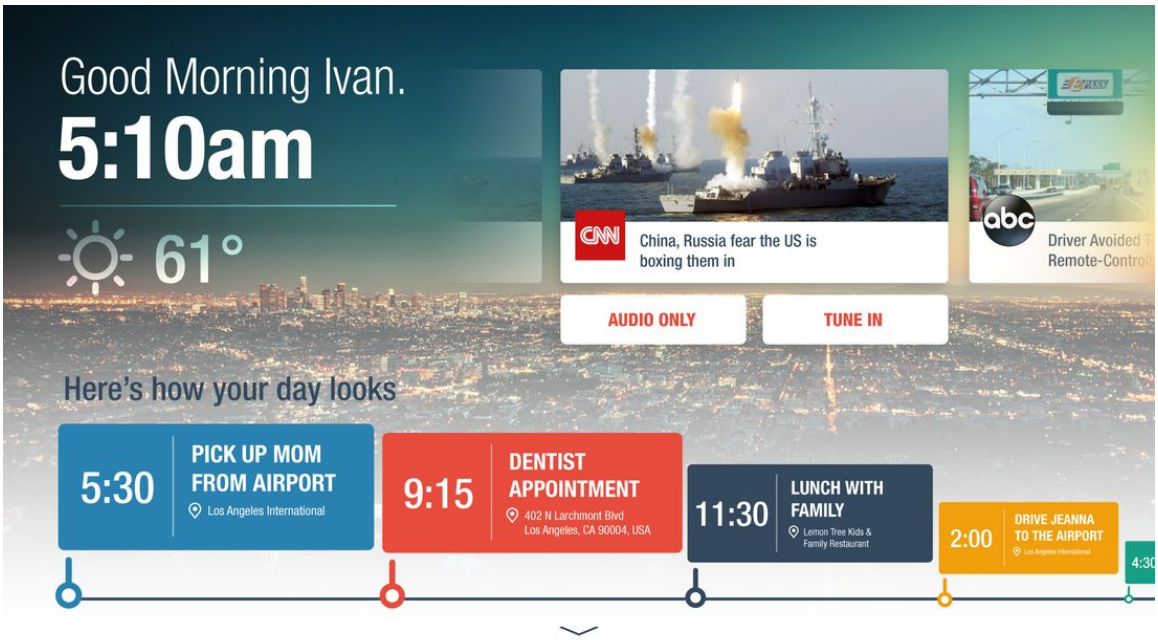

FIGURE 1

4.2 HARDWARE/SOFTWARE

The software we are using for this project is the Android OS, so any type of hardware we tested with so far 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 will run on which we just received. The new hardware that our final product will run on is a Android Kitkat media box that has the CEC functionality of HDMI.

4.2 Process

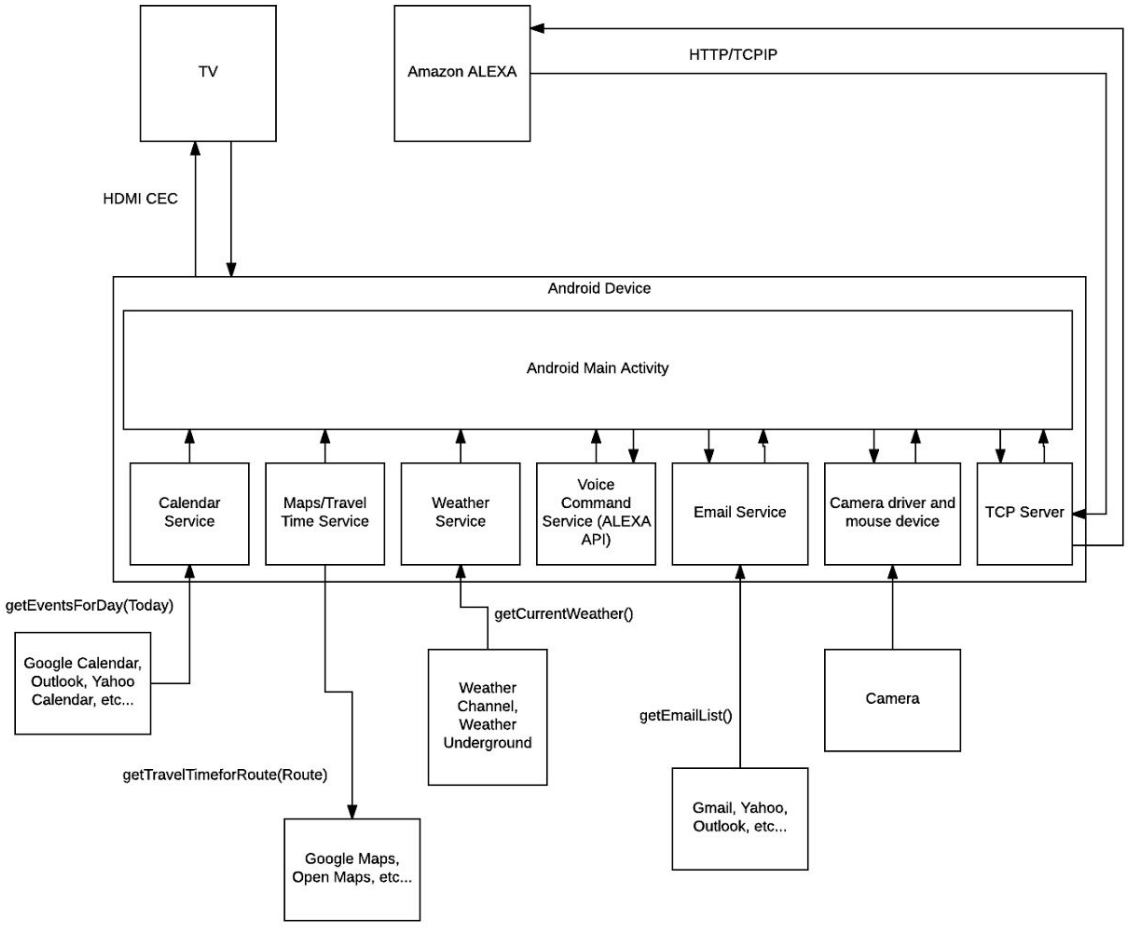

Figure 2

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(start, finish) - 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.

5 Results

The only part of this project we can test is design of the project. The project uses services as a facade to the various APIs and a group of core classes provide the wrappers for the objects provided in the services. The services are fully modular and can easily be added modified to provide very flexible code.

6 Conclusions

So far we have determined the hardware that we will be targeting and the specific APIs to be implemented on the Android system. The HDMI-CEC interface on these boards functions with CEC compatible TVs and all of the primary Android CEC functionalities that we plan to use have been tested and they function successfully. With the Kit-Kat Android TV systems the Camera we had initially began with will not work at this point; however, we were not to particular on the camera interface at this point in the project.

7 References

1. https://source.android.com/devices/tv/HDMI-CEC.html
2. https://developers.google.com/gmail/api/guides/
3. https://developers.google.com/gmail/api/quickstart/android
4. https://github.com/zh-wang/YWeatherGetter4a
5. https://developers.google.com/maps/documentation/android-api/
6. https://developers.google.com/maps/documentation/directions/

8 Appendices

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.