

**CMPE 277: Smartphone App Development**

**Cognitive Home Security – Project Report**

Submitted to

Prof. Chandrasekar Vuppalapati

By

Abhijeet Rawat

Anvit Saxena

Gaurav Kumar Gupta

Date of Submission: 05/10/2017

**Table of Contents**

**1. Project Description** **4**

**1.1. Raspberry Pi setup and connection** **4**

**1.2. Connecting to cloud** **5**

**1.3. Android Phone Application** **5**

**2. Requirements** **7**

**2.1. Software Requirements** **7**

**2.2. Hardware Requirements** **7**

**3. Mobile UI Design Principles** **9**

**3.1. Storyboards** **9**

**3.2. Wireframes**………………………………………………………………………………………………………………………………………………1**0**

**4. High Level Architecture Design**……………………………………………………………………………………………………………..**12**

**4.1. Raspberry Pi setup and connection** **12**

**4.2. Connecting to cloud** **13**

**4.3. Android Phone Application** **13**

**5. Component Level Design** **15**

**5.1. Raspberry Pi 3** **15**

Inputs 15

Outputs 15

**5.2. Windows Laptop** **15**

Inputs 15

Outputs 15

**5.3. Android Phone** **15**

Inputs 15

Outputs 15

**6. Workflow** **17**

**7. Mobile and Cloud Technologies used** **18**

**7.1. Cloud Technologies** **18**

**7.2. Mobile Technologies** **18**

**8. Interfaces: RESTful and Server Side Design** **20**

**8.1. RESTful** **20**

**8.2. Server Side Design** **22**

**9. Client Side Design** **23**

**10. Testing** **24**

**11. Automation** **26**

**12. Design Patterns used** **27**

**13. Profiling** **28**

**14. References** **30**

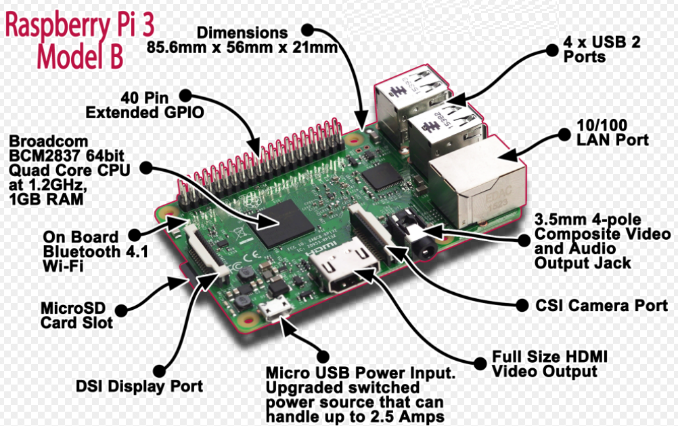
**15. Screen Captures** **31**

**1. Project Description**

Our project titled ‘Cognitive Home Security using IoT and Cloud Vision’ is focused on providing security at homes by image detection method that uses Google cloud vision API. The motivation behind this project was the increasing threats and security concerns at homes. Since, the internet of things is becoming popular these days, the concept of Smart Home came into existence. People now a day are planning to turn their home into Smart Home by using IOT. And hence, we have developed a system that uses Raspberry Pi 3, a camera sensor and a push button to click images and to further detect it using Google’s own machine learning concept. The machine learning concept provided by google helps us in detecting and analyzing the images. The images are clicked and stored in the cloud which is provided using Firebase and these images are further displayed to an Android phone application. The images are displayed along with the detected attributes in an image like the face attribute, an object or a thing. This project is divided into three different parts which are explained below:

**1.1. Raspberry Pi setup and connection**

This part deals with the hardware configuration of the project. Since, we have used Raspberry Pi 3, we have used it to connect to a bread board for establishing circuit connection. We connected camera sensor to the Raspberry Pi which is connected to bread board using jumper wires. A resistor and a push button is connected to the bread board. Android Things is the operating system installed in Raspberry Pi. The Pi is connected to the system using Android Debug Bride (ADB) which is a command line tool in the Android studio that lets you to connect to any devices like emulator or any external hardware device. The fixed IP was provided to Raspberry Pi to ensure connection between both the devices. The command used to connect the Pi to the system is ‘*adb connect <IP of Raspberry Pi>*’ and to check whether the device is connected or not the command ‘*adb devices’* is used. Below figures depict Raspberry Pi 3 device (left) and a sample push button (right) we have used in our project.



**1.2. Connecting to Cloud**

On the event of the button push, the image is directly uploaded to the cloud. This part deals with the cloud connection. We have used Firebase database provided by google to store our data. This database is connected to our project and also connected to our Android application where the data is displayed. The google cloud vision API is used to detect and analyze images and helps understand the content of an image by encapsulating powerful machine learning concepts in an easy to use REST API. We have used cloud to store out data because it is easy and hassle free. The uploaded in the cloud and displayed on the phone happens in real time such that as soon as the user clicks the push button, the image is displayed on the application immediately. We have also enabled the logs in our android project which shows when the image is being uploaded to the cloud and it also gives any error message if the upload is failed.

**1.3. Android Phone Application**

This part deals with the configuration of an Android Phone Application that is used to display the image and other data from the firebase database in the cloud. This application’s layout just has the image view to display the images and the text view to display the attribute of the analyze image using vision API. The data comes directly from the firebase database in the cloud and is displayed in the application. The architecture diagram and the screen captures is shown further in the report. This application can be run both in the emulator as well as android phone. The application loads the previously captured images also whenever the app is closed and opened again. The images and the data is directly loaded from the firebase database in the cloud. The firebase database also stores the data in the form of logs where we can see the image name as well as the timestamp of the image clicked. The image along with the timestamp is shown in the application.

**2. Requirements**

1. **Software Requirements**

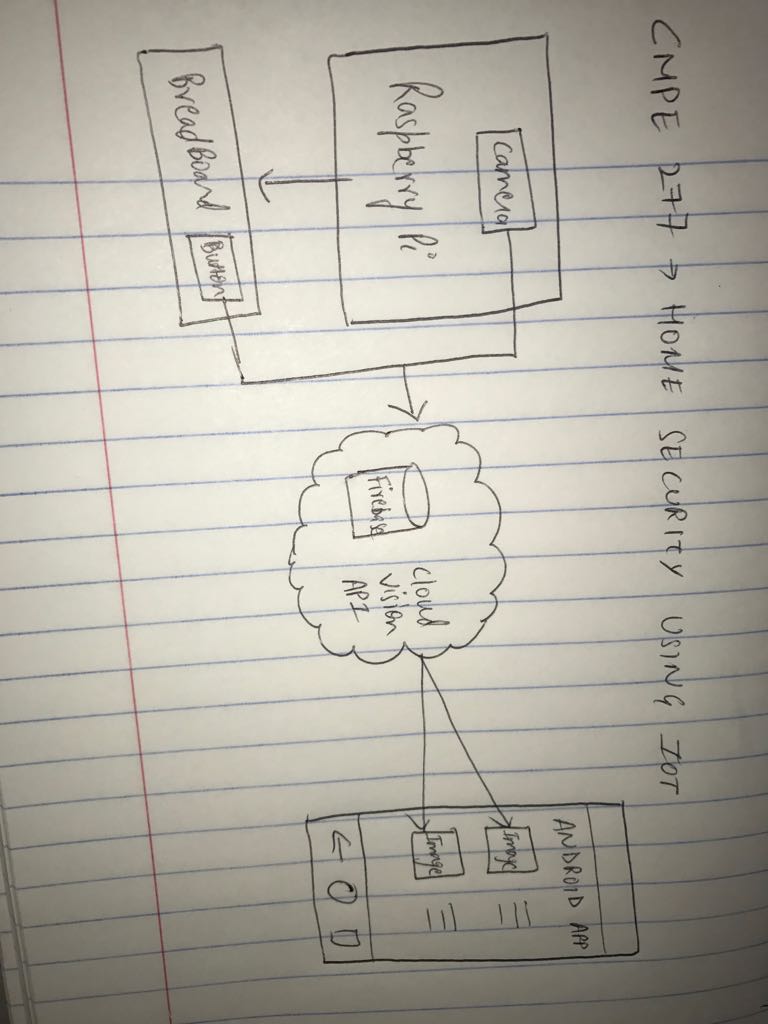
* Android Studio 2.2+: Android Studio is an IDE which is used to develop android applications or android projects provided by Google.
* Windows 10: Windows 10 operating system is used for this project which is the operating system by Microsoft.
* Any android operating system (ice cream sandwich and higher) is installed in android phone which helps in running the android application.
* Android things operating system which is Google’s operating system for IOT is installed in the Raspberry Pi 3.
* Firebase Database which is used to store data in the cloud.
* Google Cloud Vision API which is used for developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API.

1. **Hardware Requirements**

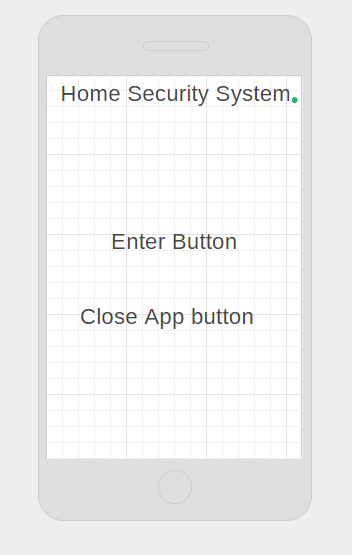
* Raspberry Pi 3: This device is a microcontroller which is used as a mini computer. This device is connected to the system.
* Camera Sensor: Raspberry Pi camera module is used as a camera sensor to capture images on the event of a button push.
* Push Button: A push button is used to connect to the bread board for making the button click event to enable camera sensor to click images.
* Breadboard: This is a board which is used to make an electrical circuit connection.
* Jumper Wires: These are used to interconnect the components of the breadboard.
* Resistor: This device has a resistance which helps in the passage of an electric current between breadboard and Raspberry Pi.
* Android Phone: An android phone which is used to install the android application to display the image and the data from the cloud.

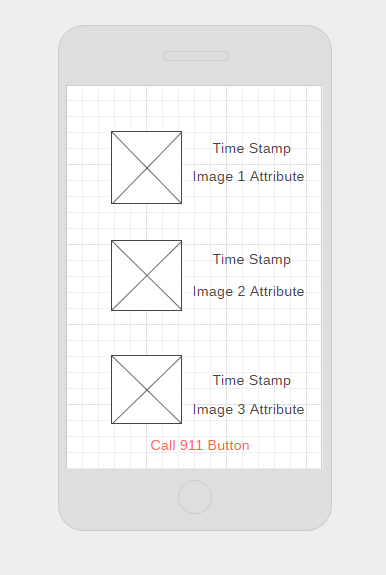
3. Mobile UI Design Principles

**3.1. Storyboards**



**3.2. Wireframes**

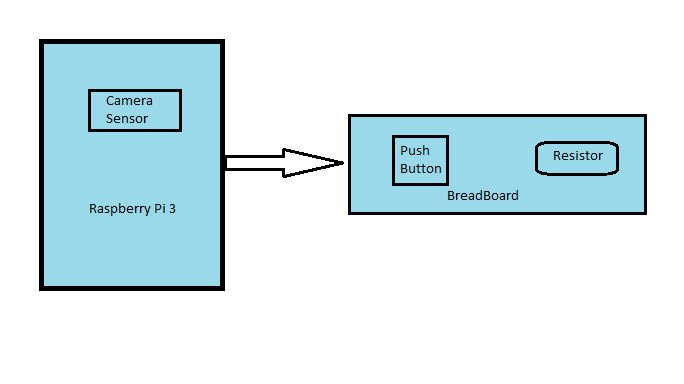




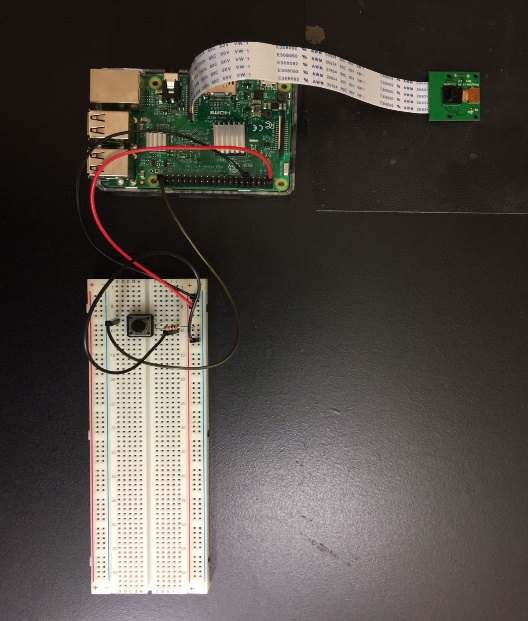
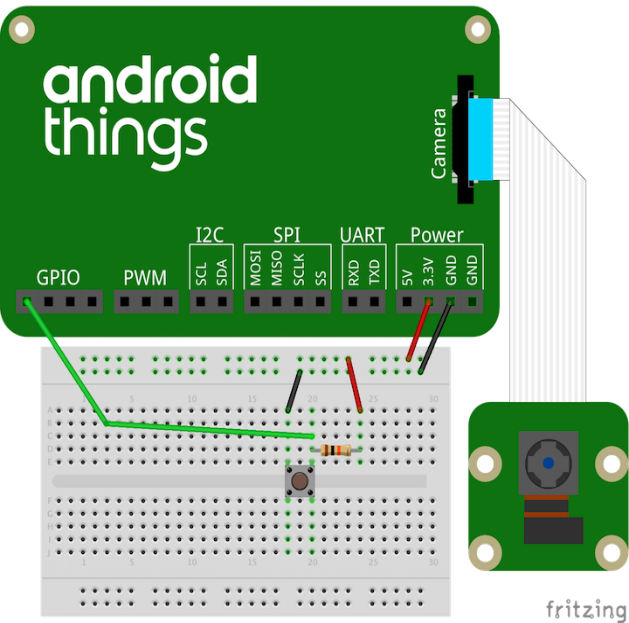
**4. High Level Architecture Design**

**4.1 Raspberry Pi setup and connection:**

The figure below shows the architecture design of Raspberry Pi connected to the Breadboard using jumper wires. Camera sensor is connected to raspberry pi device as shown below.

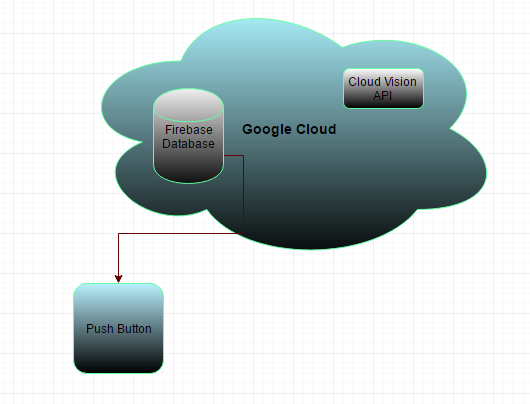


Below two figures show the actual connection we implemented on our project using Raspberry Pi and breadboard (left) and the circuit diagram of raspberry pi (right).



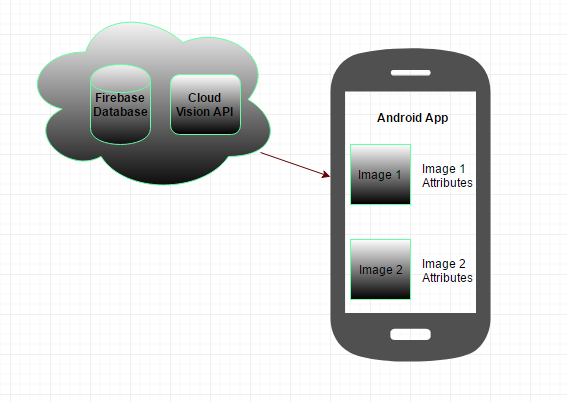
**4.2 Connecting to Cloud:**

Below figure shows the architecture design of the cloud connection. In case of the button push, image is clicked and stored in the firebase database. Cloud Vision API is used to analyze the image using Google’s machine learning module.



**4.3 Android Phone Application:**

Below figure shows the connection of Android phone via an android application that receives the real-time data from the firebase database on click of the button. This image is displayed using image view and the cloud vision API analyzes the image and displays in the text view of the android application layout.



**5. Component Level Design**

Components of the system include the Raspberry Pi 3, Android Phone and a Windows Laptop. Below is the component level design of all these components:

**5.1. Raspberry Pi 3**

Raspberry Pi is used to connect to system and it helps in enabling the camera sensor to capture image on click of a push button. The Raspberry Pi acts as a microcontroller that is connected to breadboard for electrical circuit connection.

**Input**

* Camera Sensor
* Connection to Breadboard

**Output**

* Image capture and uploading to cloud

**5.2. Windows Laptop**

The windows operating system enables laptop is used for creating the project in Android studio for both Android things and android phone application. It is also used to enable connection between Raspberry Pi and the system using adb.

**Input**

* Raspberry Pi
* Camera Sensor

**Outputs**

* + Data in the Firebase database cloud.
  + Console logs in the Android Studio.

1. **Android Phone**

An Android phone device is used to receive the image from the firebase database and display it accordingly in the app. The data received will be real time live data so that as soon as the camera clicks the image, it is loaded in the android phone application.

**Inputs**

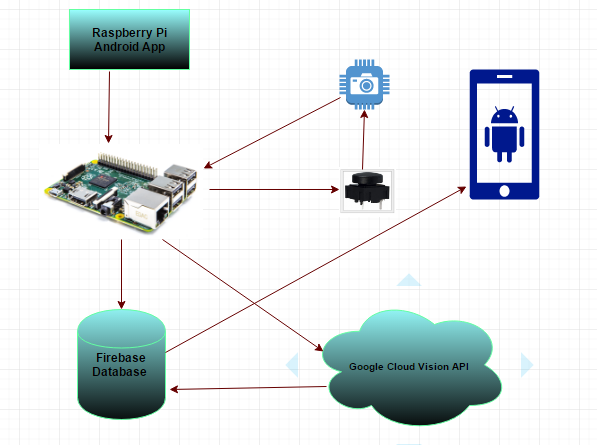
* Image data from the cloud
* Machine learning analyzed image attributes from the cloud.

**Outputs**

* Image in Image view
* Text in text view.

**6. Workflow**

Detailed steps to setup the Raspberry Pi and breadboard are included in the README document. The steps to use cloud vision API and Firebase database is also included in the README document. A brief overview of the entire workflow is given below:



The above workflow diagram explains the entire sequence flow of our project. The Raspberry Pi has an Android app which is connected to the camera sensor and push button. On the event of button push, the image is captured and sent over to Firebase database. This image is then analyzed using Google Vision Cloud API and displayed in the Android Phone Application.

**7. Mobile and Cloud Technologies used**

**7.1. Cloud Technologies**

We have used Google Cloud Platform as the cloud service provider for our project. The data is stored in the Firebase database which is a Google’s product. To demonstrate the concept of using machine learning to analyze images we have used Google Cloud Vision API. Below is the description of each one in detail:

* + Google Cloud Platform: Google Cloud Platform or GCP is a cloud computing platform provided by Google which offers infrastructure as a service to build and develop a large range of programs. We have used Google cloud API to integrate our project with Google Cloud Platform for storing and analyzing data. We have used Google Cloud Platform because it provides secure, cost effective and high performance infrastructure. The Machine Learning concept by Google is also the primary focus of using Google Cloud.
  + Firebase Database: Firebase is a company owned by Google that allows developers to store and sync data across multiple clients. We have used firebase as our database to store the image data. We have used Firebase project within our application to store and retrieve data from the database.
  + Google Cloud Vision API: Google Cloud Vision API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy to use REST API. This is provided by Google to analyze an image. We have used this API in our project to analyze our images from the firebase database and to display the image’s attributes into our android phone application in the form of text view.

1. **Mobile Technologies**
   * Recycler View: Recycle view an efficient view model which allows user to render large data set efficiently. It provides flexibility and option to customize the list of large datasets.
   * Internet Permissions: To connect to internet and perform network operation, android application should have permission to access the internet. Internet permission is defined in application manifest file.
   * Intents: In android, intent is defined as intention to perform an action. Using the intent, application can start an activity, service or send a broadcast message.
   * Views: Android View is a building block for user interface. View takes rectangular area on screen and draws a user interface component.

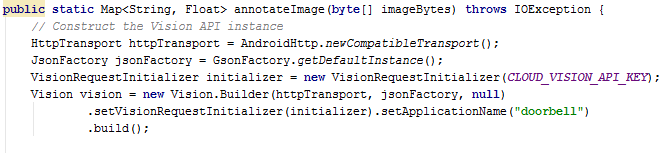
**8. Interfaces: RESTful and Server Side Design**

**8.1. RESTful**

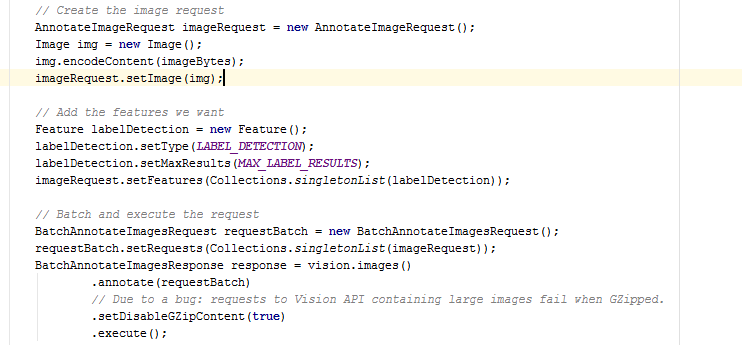
Our app has been implemented using REST software architecture principles.

For uploading:

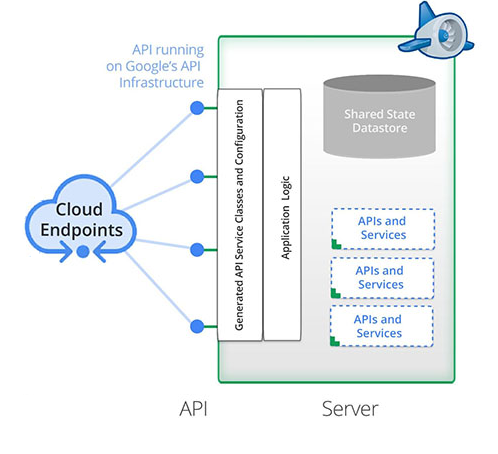




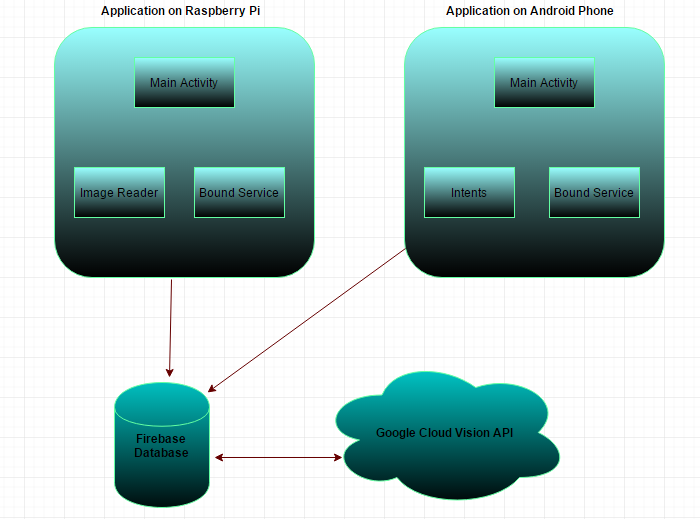
For saving:



**8.2. Server Side Design**



**9. Client Side Design**



**10. Testing**

The complete testing strategy for this android included:

1. Operational Testing
2. Integration Testing
3. UI Testing
4. Smoke Testing

**10.1. Operational Testing:**

Operational testing comprised of code walkthrough, Installation and breadboard setup verification. It is also known as acceptance or functional testing.

**10.2. Integration testing:**

The Integration testing ensures the working of different modules when put together. The project comprises of 2 Android applications and 2 cloud modules. Since, both app independently communicate with the cloud, we tested the following scenarios: -

1. Testing working of app 1 with cloud modules
2. Testing working of app 2 with cloud modules
3. Testing app 1 and 2 along with cloud modules.

**10.3. UI Testing:**

UI testing stands for User Interface testing. It comprises of asserting the visual components of the application. The focus is on verifying that each visual component performs the basic indented action. Simple gestures and actions like weather the button or other component are touch enabled and is the UI negatively affected by change in orientation. This can be done manually or using automated testing tools like selenium. Some of the scenarios tested manually are as follows: -

1. Ensuring buttons are working.
2. Testing UI with different orientation

**10.4. Smoke Testing:**

Smoke testing (also confidence testing, sanity testing, build verification test (BVT) and build acceptance test) is preliminary testing to reveal simple failures severe enough to (for example) reject a prospective software release.

Few smoke testing scenarios followed by us are as follows: -

1. Verify that camera can take picture.
2. Verify that picture is getting stored in Firebase
3. Verify that Google vision API is generating annotations
4. Verify that HomeOwner app can connect to Firebase database
5. Verify that HomeOwener app shows all the data

**11. Automation**

For automation we are using the following module :-

1- Google Cloud

2- Selenium - Selenium is a portable software-testing framework for web applications. Selenium provides a record/playback tool for authoring tests without the need to learn a test scripting language (Selenium IDE).

3- Seledroid - Seledroid is a test automation framework which drives off the UI of Android native and hybrid applications (apps) and the mobile web. Tests are written using the Selenium 2 client API.

4- AWS Jenkins for running jobs

**12. Design Patterns used**

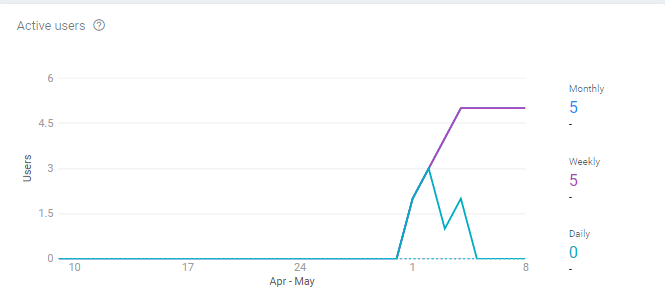
A **design pattern** is the re-usable form of a solution to a design problem. A repeatable solution to a software engineering problem.

Some of the patterns used in Android app:

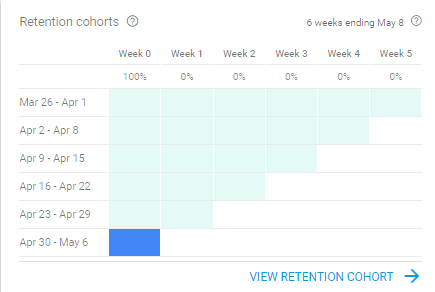
1. MVC
2. Observer
3. Behavioral Pattern-Command
4. Facade

**13. Profiling**

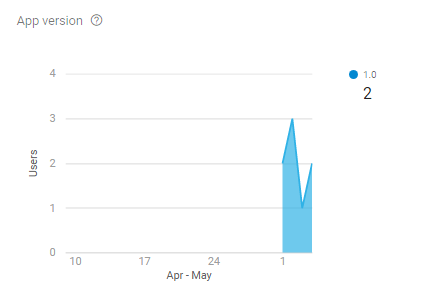
Active Users:



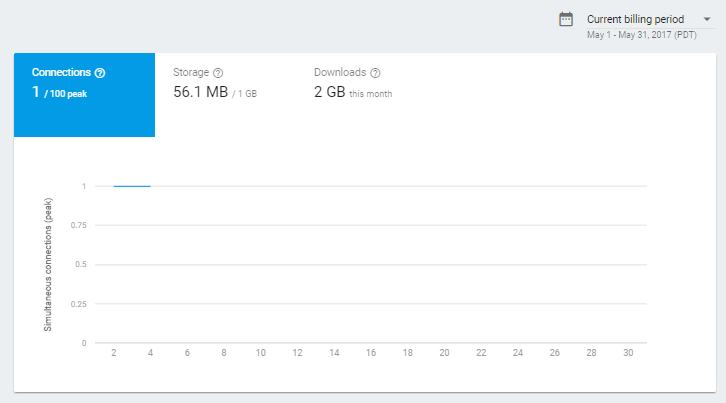
Retention Cohorts:



App version:



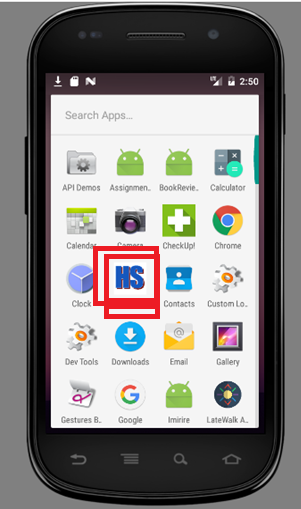
Connections Graph:

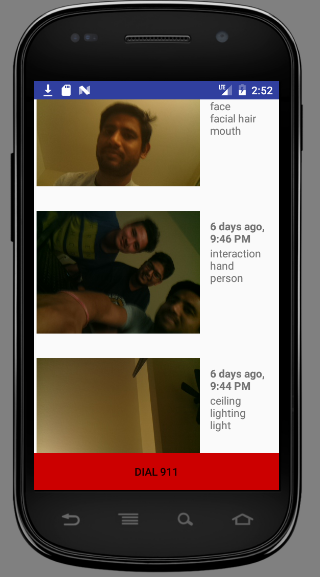
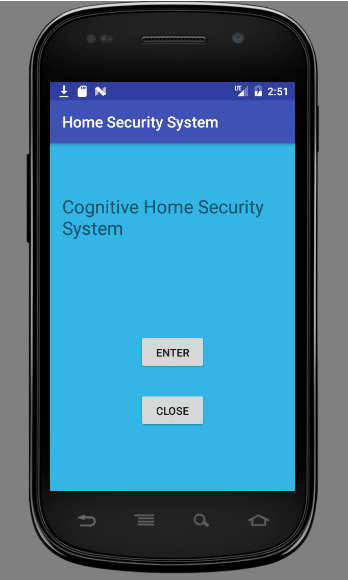


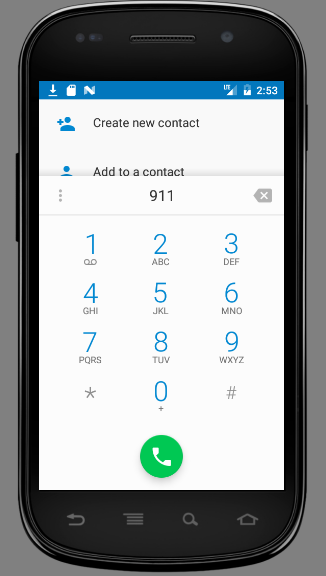
**14. References**

1. <https://developer.android.com/things/hardware/raspberrypi.html>
2. <https://www.raspberrypi.org/learning/getting-started-with-picamera/>
3. <https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera>
4. <https://cloud.google.com/vision/docs/tutorials>
5. <https://firebase.google.com/docs/samples/#android>

**15. Screen Captures**







1. **Contributions**
   * App development: Abhijeet, Anvit and Gaurav
   * Configuring Cloud Module: Abhijeet
   * Configuring Raspberry Pi: Anvit and Gaurav
   * Manual testing and profiling: Abhijeet
   * Presentation: Gaurav
   * Design document and Report: Abhijeet and Anvit