Spring 2013

Cloud Services for an Android Based Home Security System

Karthik Challa
San Jose State University

Recommended Citation

Challa, Karthik, "Cloud Services for an Android Based Home Security System" (2013). *Master's Projects*. 296.
DOI: https://doi.org/10.31979/etd.wuzr-kr66
https://scholarworks.sjsu.edu/etd_projects/296

Cloud Services for an Android Based Home Security System

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment of the

Requirements for the

Degree Master of Computer Science

by

Karthik Challa

Spring 2013

ACKNOWLEDGEMENTS

I would like to thank Dr. Soon Tee Teoh for being my project advisor and guiding me at every step of the way. He has been approachable and has provided me with encouragement, guidance and support throughout this project. I would like to thank my committee members Dr. Robert Chun and Steve Simler for their suggestions, valuable time and support. I would also like to express my gratitude to my family and friends for their continued encouragement and support.

ABSTRACT

Cloud Services for an android based home security system

This report talks in detail about an android based application designed for a home security system. The home security system is a tablet device developed using the android framework. The home security system makes use of sensors and a central device to secure an area. Currently the devices are standalone and require the users to be physically present to operate the devices with no interaction possible between two different devices. The system is also limited by its computational resources and storage capacity. For this project, I have developed a cloud based client server architecture to address these limitations and also to provide security and sharing functionalities along with remote diagnostics. This project has a device level framework to communicate with and exchange information with a cloud server. The project also addresses the primary limitations of cloud computing namely security, privacy and user control.

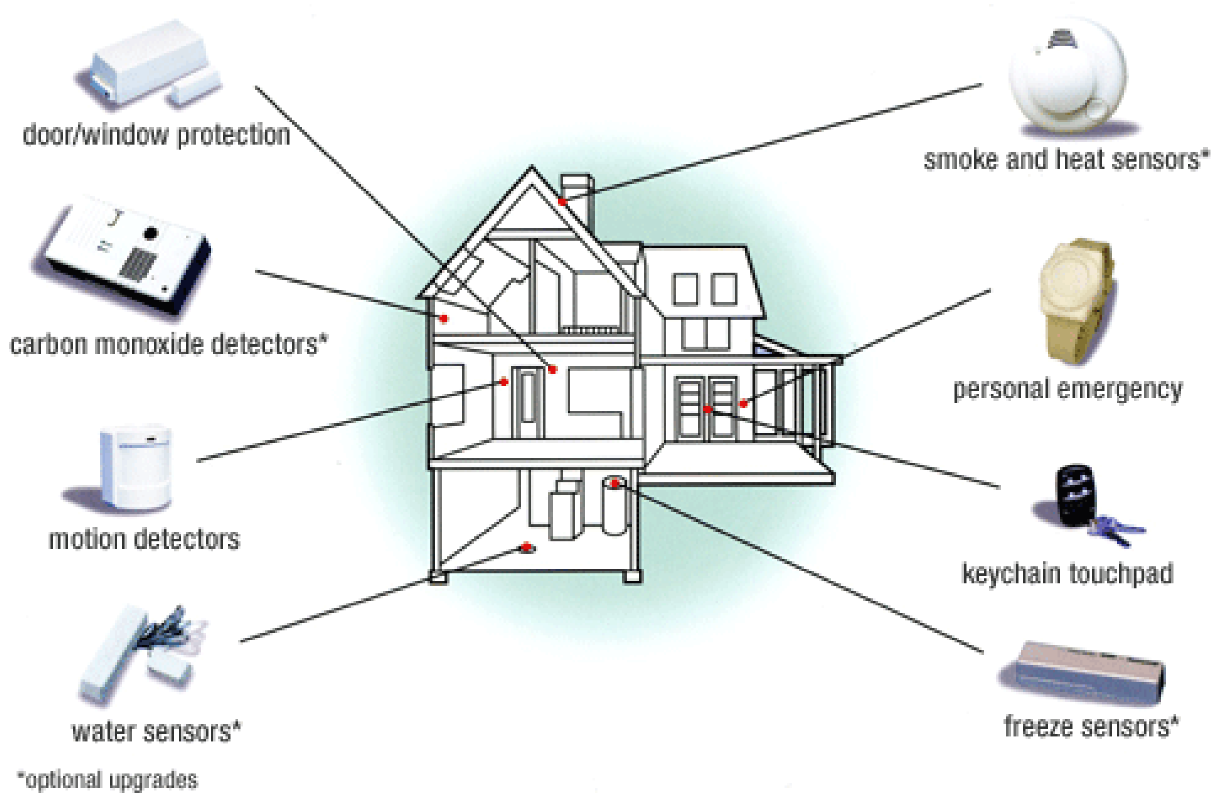
Contents

Table of Figures:

1. Introduction:

1.1 Home Security System:

Figure 1 : Security System



This project has been designed for an android based home security and automation system. The security system in question has been designed and built by Qolsys a startup based in Cupertino.

This system has been built completely on Android framework running on android 2.2.2. It is a 7 inch tablet device with built in radios and antenna to communicate between multiple types of devices using multiple communication protocols like zwave, wifi etc.

The device makes use of sensors to secure an area. These sensors communicate with the security system and depending on the mode and scenario various operations are performed. The end user can perform several operations like arming and disarming the system, selecting different arm modes etc by making use of the touch screen and a virtual key pad. The device also acts as smart-phone device and lets users play music, movies, create video messages, run other android applications etc.

Currently all the information is restricted to the device itself. All the database information related to sensors, user information etc is stored locally on the device. Also the user pictures and messages are stored in the device with no way of sharing them with others. Another limitation is that, the applications on the device are restricted by the available storage and if the device runs out of space, then files will have to be removed.

For this project, I have designed and implemented an android based application which uses cloud based architecture to address these limitations.

1.2 Project Scope and Features:

For this project, I have designed and implemented a client and server based android application to communicate with and exchange information with a remote cloud server. The application has been designed on android 2.2 and supports multiple features like user authentication , data encryption and security, secure access, image capture and secure, image upload and share and data repudiation.

I have implemented features to overcome the limitations of both the security system and the cloud servers. The application has been designed in a modular manner, with each module building on top of the previous one, and offering new functionality. The application supports a centralized user information repository which is hosted on Amazon Relational Database System.

The server side scripting has been done in PHP and this interacts with the remote database and also performs secure operations like encrypting the user credentials and salting the information to authenticate users using a hash based authentication.

Image capture and secure is a module which lets the users capture images and gives them an option to display the images on the device or to encrypt and hide the images by deleting them form the SD card or to decrypt and retrieve the images.

Image Upload and display builds on top of the image capture functionality, and supports the following functionality. It lets the users share the images online using a presigned link which can be used to view the picture on a browser. It also supports uploading the encrypted files securely to the server where they can be stored.

The application also supports a push based notification system designed by using Amazon Simple Notification Service, which lets users subscribe to a topic using their email address, and any messages posted by the administrator, are then send to the subscribing users.

I have also implemented a fail over recovery for the database using Amazon EC2 servers. This feature allows the server to set up a secondary server to act as a backup in case the primary server goes down. I have set up database synchronization between these two servers.

The potential uses for these features on the security system are as follows

a. The system takes pictures based on multiple factors like alarm conditions, user images etc. This feature will let the users upload pictures to a remote server and view them using their mobile phone without compromising on the security.

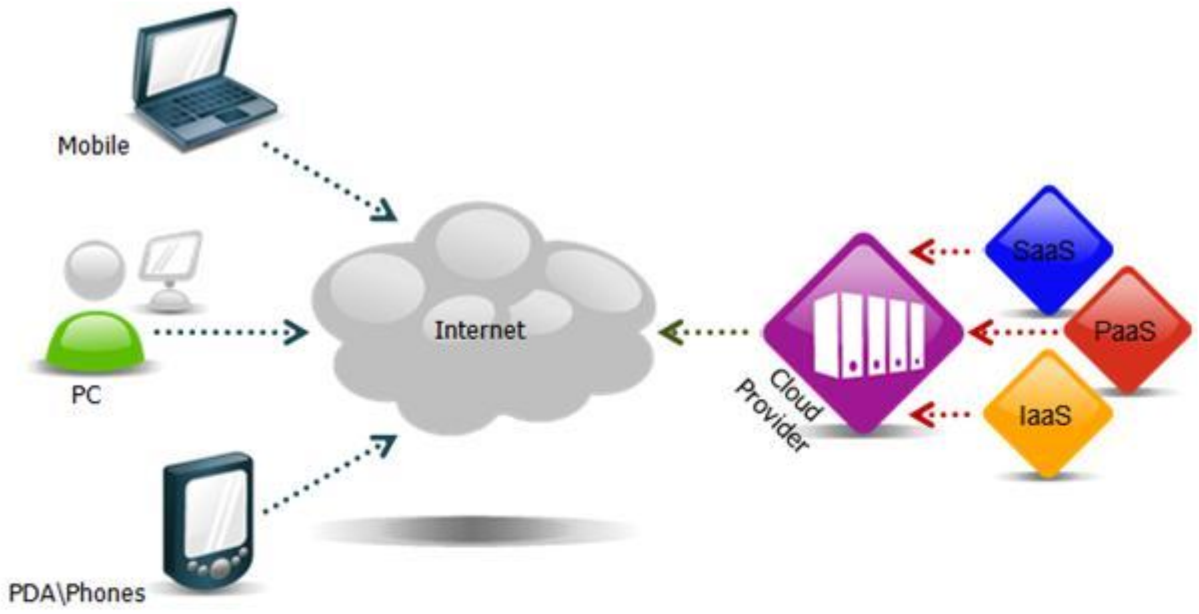
b. Having the sensor database and user info centralized on a remote db, will let the panel support remote backup and retrieval.

c. The push notifications can be used by dealers, to send specific messages to users based on specific issues.

2. Background Information:

2.1. Introduction to Cloud Computing:

Figure 2 Cloud Computing Overview



National Institute of Standards and Technology defines Cloud Computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.[1]

Cloud computing in essence translates to a virtual pool of resources and computational facilities which can be used based on the usage requirements. It abstracts the details of these resources, and the end users can just use them without worrying about the underlying architecture.

2.2 Advantages of Cloud Computing:

1. Inexpensive:

Cloud providers usually charge the users based on the usage, this reduces the upkeep costs for the organizations. Since the cloud providers take care of the licensing costs and are responsible for setting up the servers and maintaining them, this allows the organizations to just focus on the product development and not worry about the overhead associated with storage and computational facilities.

2. Scalability:

One of the primary reasons why organizations and developers are inclined to using cloud computing is scalability. Scalability allows the organizations to dynamically increase or decrease the servers being used without planning in advance. For instance, if a website has 500 hits each month, but suddenly sees a spike of users to say 1000, then the developer can easily increase the server limit, without worrying about the logistics of the upgrade. Similarly when there is a decrease in usage, the developer can easily scale back without having to pay the additional costs.

4. Ease of Configuration:

Cloud computing provides a set it and forget it approach, where a developer can setup one instance of his/her software on the server, and all the other clients associated with the server will automatically get the same version of the software. In the event that the developer updates the software version on the cloud server, all the clients are automatically updated. This allows the data on the all the clients to be in sync without worrying about version control.

2.3 Challenges of Cloud Computing:

1. Security:

Security and user privacy remains one of the biggest hurdles for cloud computing. Since all the user data is stored on the cloud server which belongs to a third party, many organizations are hesitant to store all their data with these providers.

2. Cannot be used for Real Time Applications:

Since the basic premise behind cloud computing is a client server model, this involves requests being send to the server, and the server processing those requests and sending a response out, this model cannot be used for applications which require a response in real time.

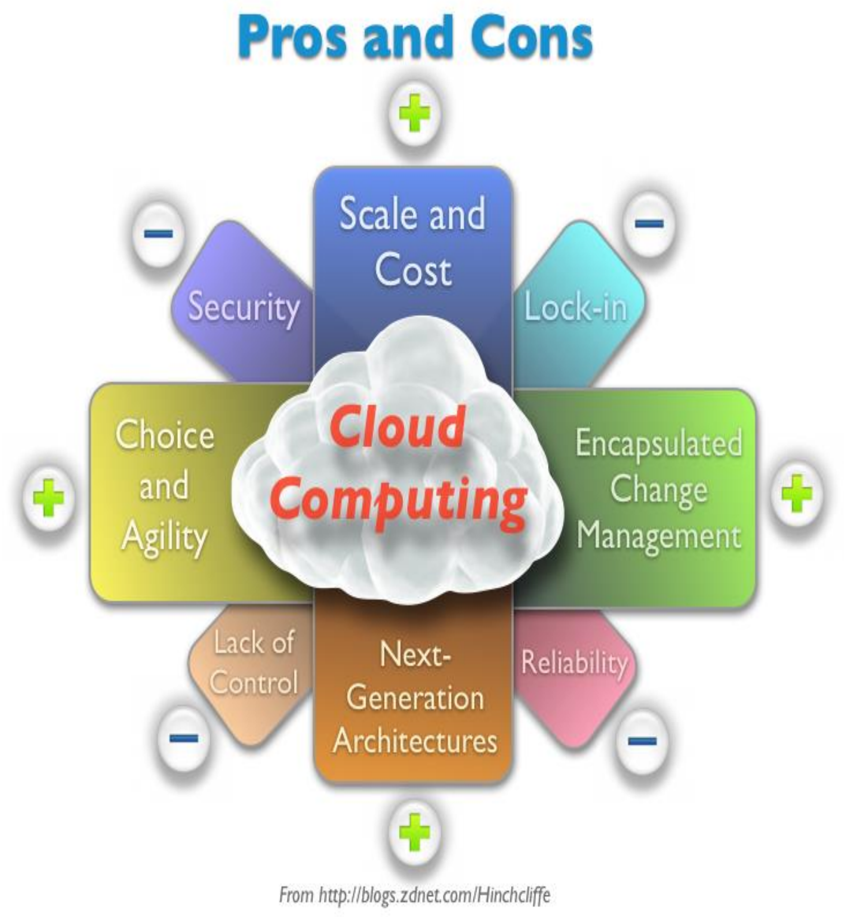
3. Downtime:

If of the major bottleneck for cloud computing is that the websites or software running on the cloud will depend on the cloud server. If the server goes down for some reason, then all the services making use of that service will also go down. Also since a large amount of data is usually stored on the servers, if the server goes down, then the data is also lost in these cases.

In my project, I have addressed some of these concerns especially the privacy and security and the dependency on one single server. For my project, I store all the user data in an encrypted format and use hash computations to verify the user credentials. This way I avoid the transmission of sensitive data on a communication channel. To address the limitation of dependency, I have set up data replication using Amazon Relation Database System and

MySQL. This synchronizes the data between the two, so I always have a backup of the database contents.

Figure 3 Pros and Cons of Cloud Computing

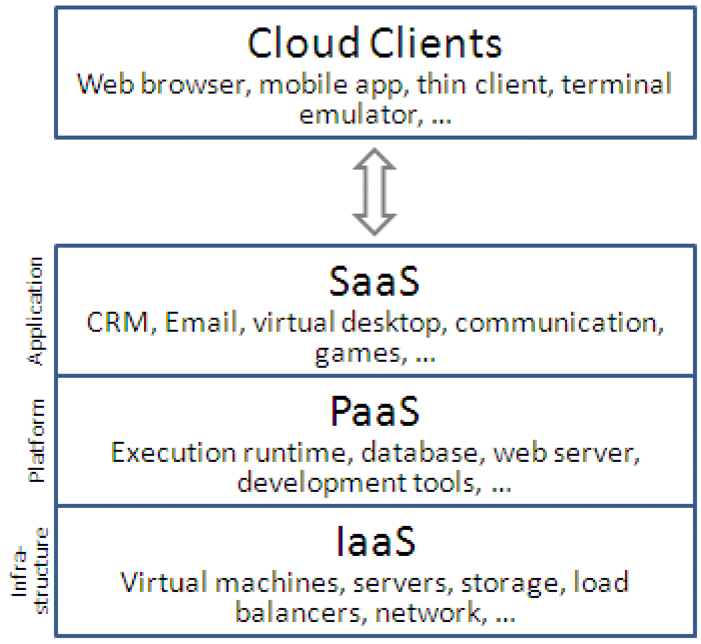


2.4 Cloud Computing Services:

Cloud computing services are broadly categorized into three types:

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

Figure 4 Cloud Computing Services


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure

2.4.1 Software as a Service (SaaS):

Software as a service is arguably the most popular distribution model of cloud computing. In this model, the developer designs and develops the software, and hosts it completely on the cloud servers. The developer then makes this product available over a network for end users.

The most common applications of this model include web sites like Yahoo, Google, Gamespot etc which rely on these cloud servers to process and store the huge amount of traffic and data it receives.

2.4.2 Infrastructure as a Service (IaaS):

Infrastructure as a Service is a cloud computing model primarily used by enterprises which need access to the cloud servers, but would like to configure and maintain them independently. In this model, the cloud provides provide the infrastructure i.e. machines, servers etc that they own as a service. The client takes the responsibility to configure and set up these machines along with the database and other resources.

The cloud provider is only liable to provide and maintain the hardware required while all the software is taken care of by the enterprise.

2.4.3 Platform as a Service (PaaS):

Platform as a Service is a cloud computing model where the developer provides the tools as a service and the consumer uses those tools to develop software. The consumer is also responsible for configuring and deploying the software. The cloud provider is responsible for providing the networks, storage and other services.

An example for this would be Google Cloud and Open Stack etc. which provide the users with tools and resources to build applications.

2.5 Amazon Web Services:

Amazon Web Services is a collective term which encompasses all the modules being offered by Amazon, under the umbrella of Cloud based services.

Amazon Web Services, at this time, offers twenty five different remote services each of which can be used independently depending on the desired functionality.

These services are primarily used by developers to design and build applications, making use of this service.

For my Master’s project, I made use of the following Web Services

1. Amazon Elastic Compute Cloud (EC2)
2. Amazon Simple Storage Service (S3)
3. Amazon Simple Notification System (SNS)
4. Amazon Relational Database Service (RDS)
5. Amazon Simple Queue Service (SQS)

3. Implementation

The project has been developed in a modular manner with each module building on top of the previous one. An overview of the web services and client side systems used in this project are as follows.

a. Amazon Elastic Compute Cloud (EC2): I have developed my server using Amazon's EC2. All the server side computations and secure connectivity are handled here.

b. Amazon Relational Database System (RDS): I have used Amazon's RDS to create and deploy the cloud database.

c. Amazon Simple Storage Service (S3): I have used Amazon's S3 as a remote storage end point to store all the client side data.

d. Amazon Simple Notification System (SNS): I have used Amazon SNS to implement a push notification system for the client system.

e. Server Side Scripting: I have developed all the server side scripts and computations using PHP as the scripting Language.

f. Client Development: Since the application is designed for an android based home security system, the client system has been designed on android the platform running Android 2.2 operating system.

g. Databases: For the server side computations, I have used MYSQL as the database system. For client side computations, I have used SQLite as the database handler.

Modules and their Implementation:

1. Creating and Instantiating the Cloud Server
2. Server Side Computation
3. Encryption and Salting at Server Side
4. Client Side Development
5. Image Capture and Secure
6. Amazon S3 Integration
7. Browser Display
8. Push Notification using E-Mail
9. Diagnostic Monitoring

3.1 Creating and Instantiating the Cloud Server:

To reduce the application size, and to maintain a centralized information flow, I have created a Linux based Web server using Amazon's EC2 to act as the server for my application. The steps involved in creating and instantiating the server are as follows:

1. Creating an EC2 Instance
2. Configuring an EC2 Instance
3. Security Groups and Permissions
4. Elastic IP
5. Secure Connections
6. Server Side Computation

Creating an EC2 Instance:

The first step is to login to the Amazon web services console, and open the EC2 dashboard. For this project, we will be using a Linux Based AMI to setup and run our server side scripts. We will also create a key value pair and a private key to securely connect to the instance.

Creating the Key Pair:

A valid key pair is required to connect to the EC2 instance. We need to create a pem key, which is a RSA key pair containing the private key of that instance.

To create a key,

```
C:> $admin | Format-List KeyName, KeyFingerprint, KeyMaterial [ ]
```

Key Pair Name: ec2-ubuntu

Fingerprint: 07:70:b9:17:9c:65:5e:7d:a5:40:82:d5:f0:67:bd:e3:d7:9c:62:ec

Snippet of the RSA key :

```
-----BEGIN RSA PRIVATE KEY-----

MIIEogIBAAKCAQEAijNvctEedgeY50VgqSDF0oXXverxve5q/yE/m6Nqt+fdr4vuNFV8WVR5PUZq

y03ZAMD9WHChB8ULhJ2hnEi+rUPwpEuamH0SwGvBbxMdGttviY87eTAPE/eWxD9CsaH/3MOD0hpp
```

Y0uccgU1XNZ65iJpcapEkXsmtdmr9NCmk6yHXplogPX/np3mo3+ZKZdh1Ts7jgqCDHEWh6
3gsMdn

____-

After creating the private key, we need to pipe it to an output file which can then be used as a private key. This file is saved with an extention of .pem.

```
$admin.KeyMaterial | Out-File -Encoding ascii admin.pem
```

Configuring the EC2 Instance:

Once we have created the EC2 instance, we need to configure the access to the instance. Each instance will have a unique instance id and a public DNS address which can be used to connect to the instance.

Security Groups and Permissions:

Before creating the instance, we need to create a security group to set up the required permissions. Since we want a secure connection to our instance and a secure communication between our client and server, we need to harden our instance accordingly. We need to permit SSH and connections on a specific range of ip addresses.

```
PS C:> Grant-EC2SecurityGroupIngress -GroupId sg-04037f34-IpProtocol tcp -FromPort 22 -
ToPort 22 -CidrIp 0.0.0.0/0
```

In the above statement, we are allowing connections from any ip address on port number 22 for the security group having the id 04037f34.

This lets us open secure shell connections between the EC2 server and the clients.

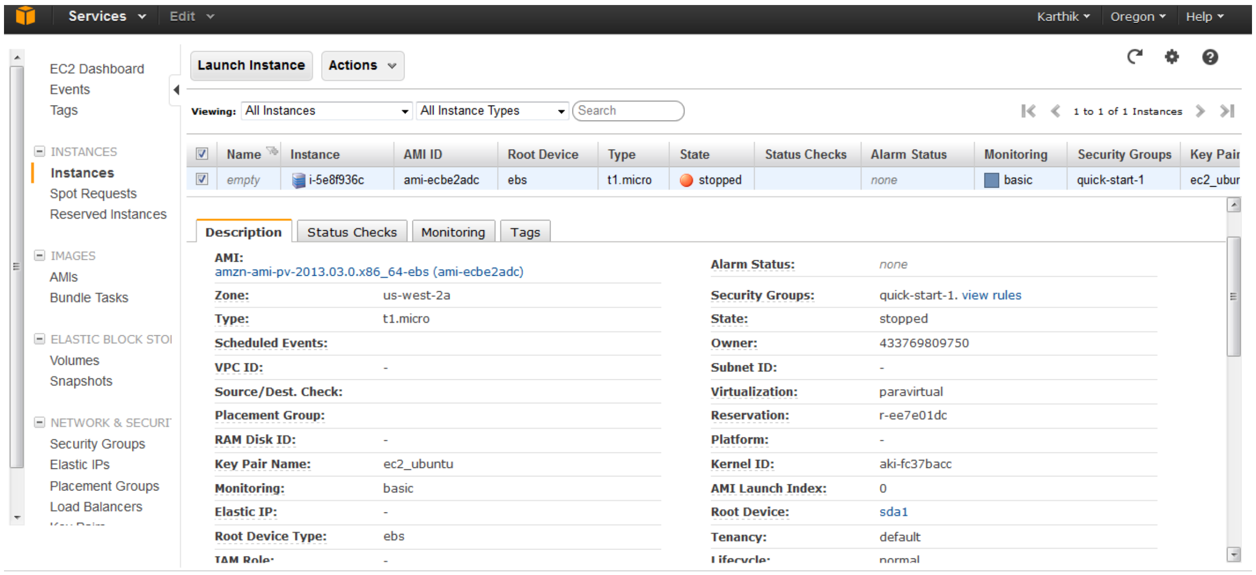
We will also create an inbound connection on port 80 to listen to http connections.

PS C:> Grant-EC2SecurityGroupIngress -GroupId sg-04037f34-IpProtocol tcp -FromPort 80 -ToPort 80 -CidrIp 0.0.0.0/0

The above permission, let's us communicate with the instance using http connection on port 80.

Fig: Instance Information

Figure 5 EC2 Instance Overview



✓	Name	Instance	AMI ID	Root Device	Type	State	Status Checks	Alarm Status	Monitoring	Security Groups	Key Pair
✓	empty	i-5e8f936c	ami-ecbe2adc	ebs	t1.micro	● stopped		none	basic	quick-start-1	ec2_ubur

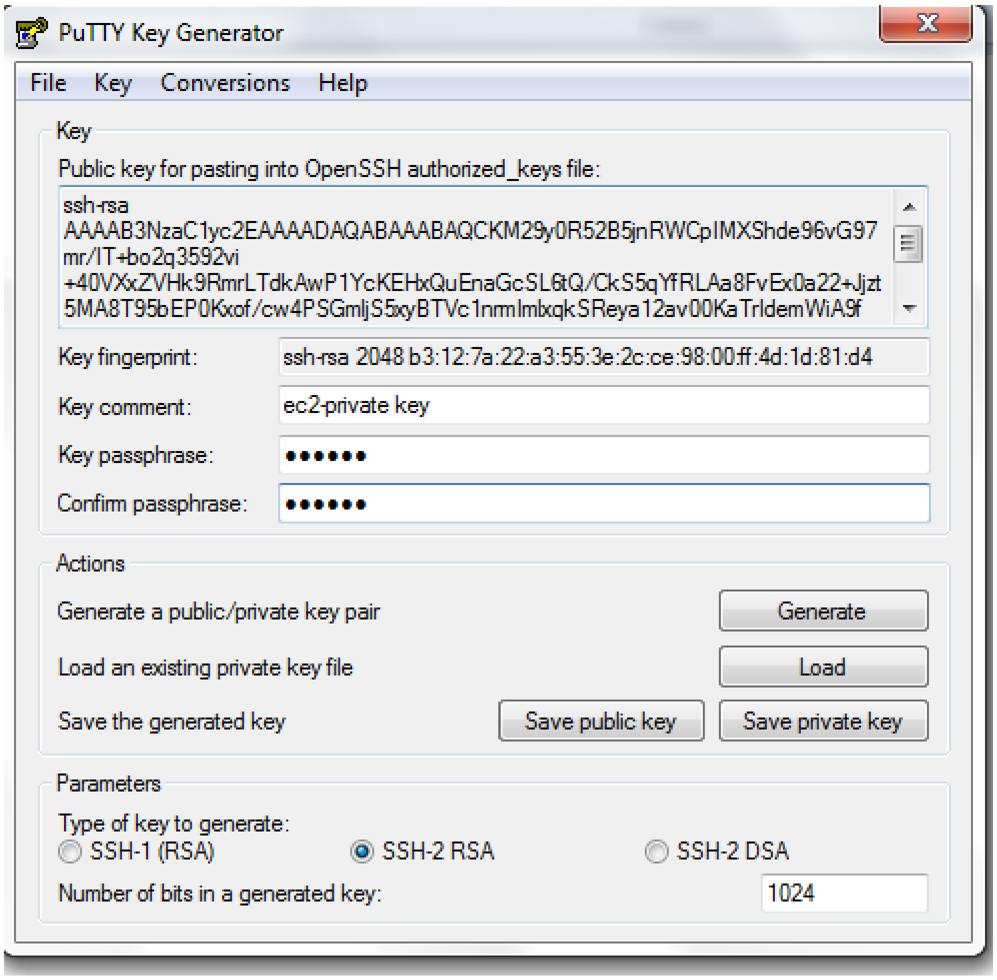
AMI:	amzn-ami-pv-2013.03.0.x86_64-ebs (ami-ecbe2adc)	Alarm Status:	none
Zone:	us-west-2a	Security Groups:	quick-start-1. view rules
Type:	t1.micro	State:	stopped
Scheduled Events:		Owner:	433769809750
VPC ID:	-	Subnet ID:	-
Source/Dest. Check:		Virtualization:	paravirtual
Placement Group:		Reservation:	r-ee7e01dc
RAM Disk ID:	-	Platform:	-
Key Pair Name:	ec2_ubuntu	Kernel ID:	aki-fc37bacc
Monitoring:	basic	AMI Launch Index:	0
Elastic IP:	-	Root Device:	sda1
Root Device Type:	ebs	Tenancy:	default
IAM Role:	-	Lifecycle:	normal

Elastic IP:

Each instance created on Amazon EC2, will have its own Public DNS address. This public DNS address is associated with the instance as long as it is powered on. When the instance is shut down or terminated, it loses the public dns address. This public DNS address is quite long and subject to change. To make accessing the instance easier, we can associate an instance with an elastic ip address. First we request for an elastic ip and associate it with the instance. After association, the instance can be connected to using the ip address.

Secure Connections: Once the instance is set up and connected, we need to establish a secure connectivity to work with the instance. For this we use putty. First we need to create a public/private key pair using the .pem file that was created earlier.

Figure 6 Public Key Generation



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCKM29y0R52B5jnRWCpIMXShde96vG97
mr/IT+bo2q3592vi
+40VXxZVHk9RmrLTdkAwP1YcKEHxQuEnaGcSL6tQ/CkS5qYfRLAa8FvEx0a22+Jjzt
5MA8T95bEP0Kxof/cw4PSGmljS5xyBTVc1nmlmlxqkSReya12av00KaTrldemWiA9f
```

Key fingerprint: ssh-rsa 2048 b3:12:7a:22:a3:55:3e:2c:ce:98:00:ff:4d:1d:81:d4

Key comment: ec2-private key

Key passphrase: ●●●●●●

Confirm passphrase: ●●●●●●

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

Now that the key pair generation is complete, we connect to the instance using putty or ssh.

```
ssh -i /path/to/private-key root@<ec2-instance-address>
```

3.2 Server Side Computation:

Once the server is setup and configured, we create and deploy PHP and MYSQ server on the device. Once this is done, I create the script files and save them in the www/html folder. Since I have setup permissions to access this folder from external applications, I save my scripts in this location, and access them.

The major operations carried out by the server are as follows:

1. User Registration and Verification
2. User Login.
3. Encrypting the User Credentials and hash verification to authenticate users
4. Connecting to the remote database and parsing data.

The server side scripts have been written using PHP and MySQL. The client initiates the communications with the server. To optimize the efficiency and to reduce lag times, all the computations are performed at the server.

Information exchange between the client and server is done using JSON parsing and using TAGS. I have written API calls which recognize the TAGS and perform operations accordingly.

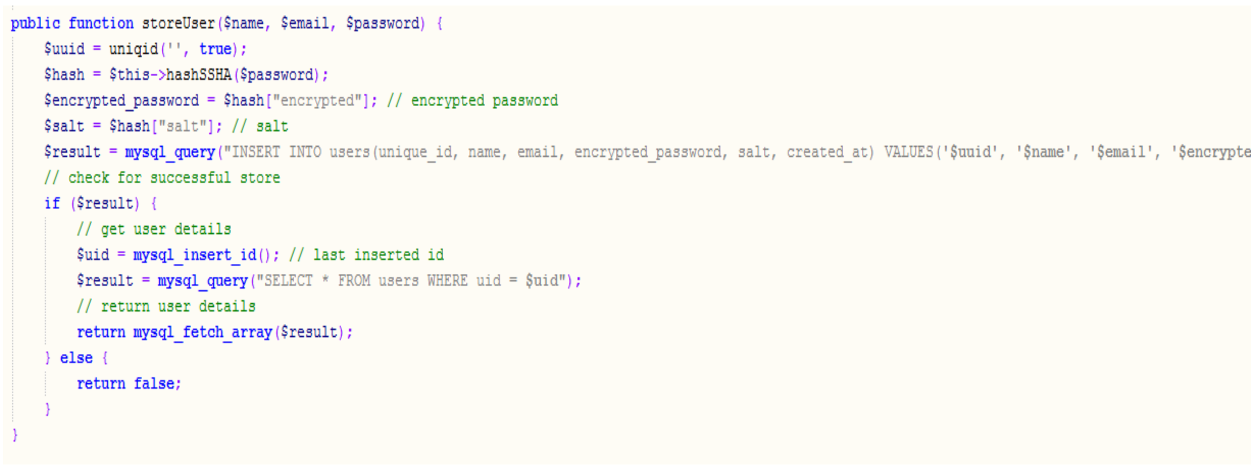
The server identifies two primary TAGS from the client to start the communication, the login tag and the registration tag. The login tag tells the server that the client would like to login using existing credentials, and the registration tag asks the server to open a channel

The server opens a connection with Amazon RDS to interact with the database. If the client wants to register, the client passes the users email address, username and password encoding them using JSON. The server on receiving these parameters processes the inputs and upon validation, communicates with the database to create the accounts. The server also maintains a time stamp of when the accounts were created or updated. One major limitation of using remote storage is that the user information is stored in the clear. This is not secure. To overcome this limitation, two operations are performed by the server before communicating with the database.

Encryption and Salt:

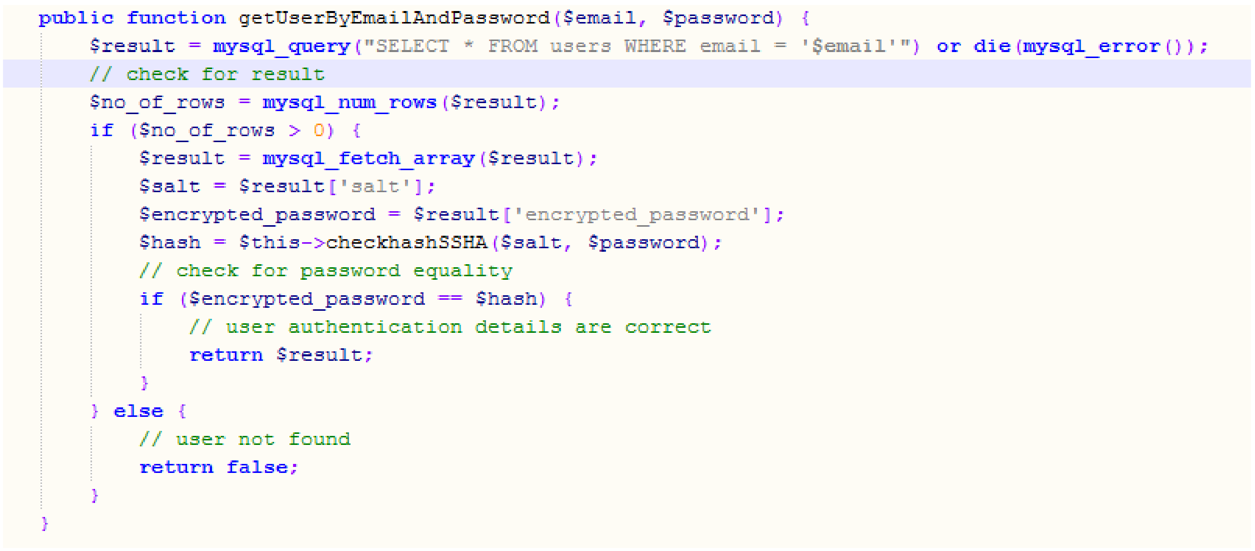
After the server processes the user info, it encrypts the user info by calling the encryption function. The function first generates a salt value for the user details, and computes the hash of the function using the SHA cryptographic function. Whenever the client sends the data to the server, this data is secured before being stored in the database.

Here is a code snippet showing the function to create a new user.



```
public function storeUser($name, $email, $password) {
    $uuid = uniqid('', true);
    $hash = $this->hashSSHA($password);
    $encrypted_password = $hash["encrypted"]; // encrypted password
    $salt = $hash["salt"]; // salt
    $result = mysql_query("INSERT INTO users(unique_id, name, email, encrypted_password, salt, created_at) VALUES('$uuid', '$name', '$email', '$encrypte
    // check for successful store
    if ($result) {
        // get user details
        $uid = mysql_insert_id(); // last inserted id
        $result = mysql_query("SELECT * FROM users WHERE uid = $uid");
        // return user details
        return mysql_fetch_array($result);
    } else {
        return false;
    }
}
```

The following code snippet shows the server getting the user info by comparing the stored hashes.

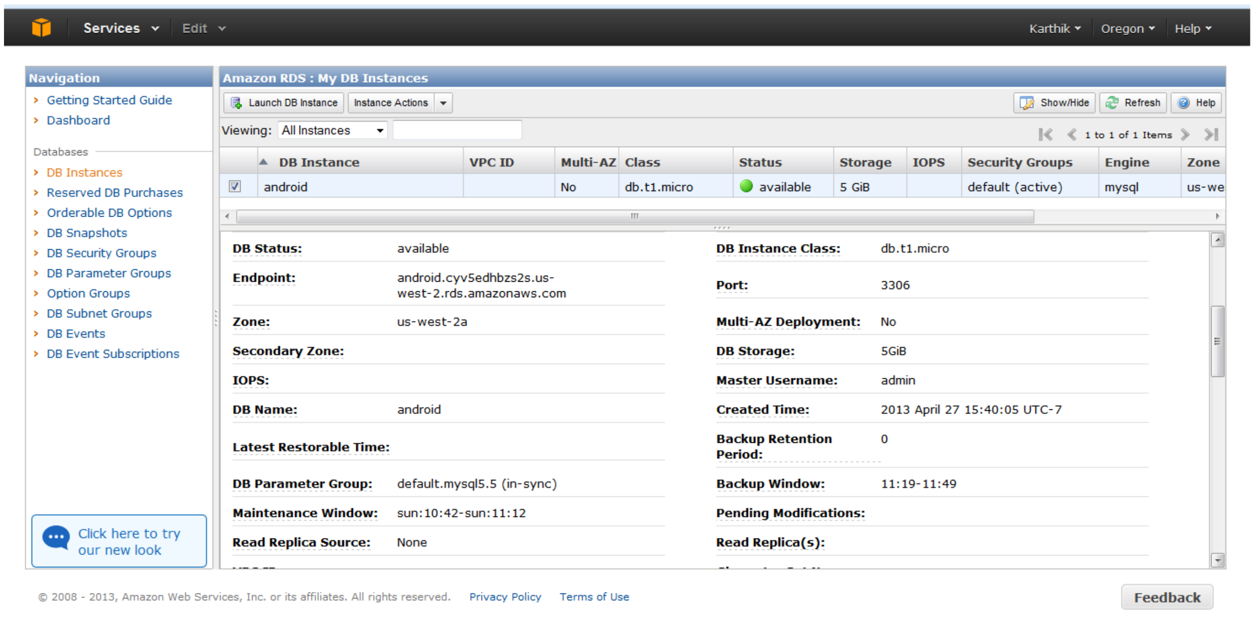


```php
public function getUserByEmailAndPassword($email, $password) {
    $result = mysql_query("SELECT * FROM users WHERE email = '$email'") or die(mysql_error());
    // check for result
    $no_of_rows = mysql_num_rows($result);
    if ($no_of_rows > 0) {
        $result = mysql_fetch_array($result);
        $salt = $result['salt'];
        $encrypted_password = $result['encrypted_password'];
        $hash = $this->checkhashSSHA($salt, $password);
        // check for password equality
        if ($encrypted_password == $hash) {
            // user authentication details are correct
            return $result;
        }
    } else {
        // user not found
        return false;
    }
}
```

3.3 Database Configuration and Queries:

For this project, I have created an instance of the Amazon Relational Database and have used for remote storage. This database stores the user information in an encrypted format. The server side scripts control and maintain this database.

Figure 7 Relational Database System

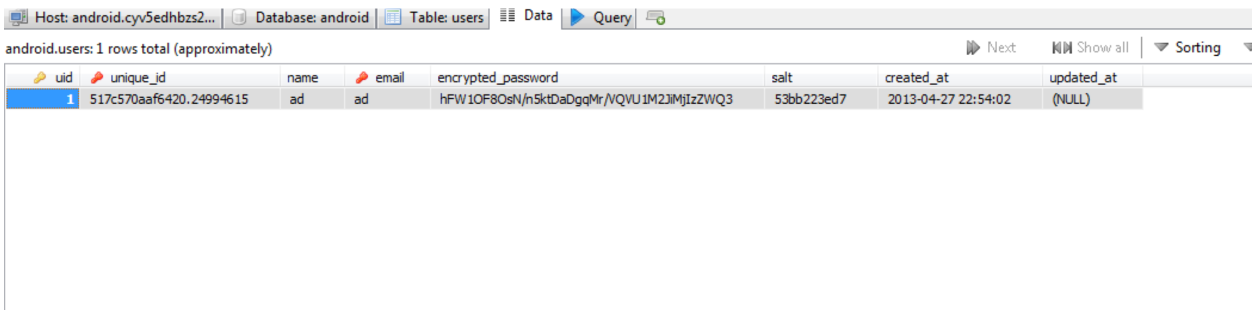


	DB Instance	VPC ID	Multi-AZ	Class	Status	Storage	IOPS	Security Groups	Engine	Zone
✓	android		No	db.t1.micro	● available	5 GiB		default (active)	mysql	us-we

DB Status:	available	**DB Instance Class:**	db.t1.micro
Endpoint:	android.cyv5edhbzs2s.us-west-2.rds.amazonaws.com	**Port:**	3306
Zone:	us-west-2a	**Multi-AZ Deployment:**	No
Secondary Zone:		**DB Storage:**	5GiB
IOPS:		**Master Username:**	admin
DB Name:	android	**Created Time:**	2013 April 27 15:40:05 UTC-7
Latest Restorable Time:		**Backup Retention Period:**	0
DB Parameter Group:	default.mysql5.5 (in-sync)	**Backup Window:**	11:19-11:49
Maintenance Window:	sun:10:42-sun:11:12	**Pending Modifications:**	
Read Replica Source:	None	**Read Replica(s):**	

© 2008 - 2013, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Figure 8 Encrypted User Information



android.users: 1 rows total (approximately)

uid	unique_id	name	email	encrypted_password	salt	created_at	updated_at
1	517c570aaf6420.24994615	ad	ad	hFW1OF8OsN/n5ktDaDgqMr/VQVU1M2JiMjIzZWQ3	53bb223ed7	2013-04-27 22:54:02	(NULL)

3.4 Android Client Development:

The client for this product is an android based device running on android 2.2. When the user opens the application, he/she is presented with the welcome screen. This screen presents the user with two options, Login and Registration. Depending on which option is selected, an intent is called to switch the display to that screen and to load the resources of the specified class files. By using this approach, I ensure that the application does not load all the classes at run time, and only uses them when called, minimizing the memory consumption.

If the registration option is selected, then the application switches the view to the login screen. To create a new account, the user has to enter his username, email address and a password. These values are then encoded using JSON and send to the server. These parameters are stored as a name value pair on the client. The client uses SQLite as the database handler.

The server makes use of MySQL to communicate with the cloud database. The android client uses SQLite as the database handler. Both MySQL and SQLite cannot communicate directly with each other, for this reason, the responses are encoded in JSON and parsed at the client and server to enable the flow of information.

If the user clicks on login, then the user passes his email and password as parameters. The client uses a http post method, to send the information to the server. When receiving information from the server, the client opens a http client object and gets the entity associated with the JSON url. Both login and registration have their own specific url which gives sends the output out in JSON parsed format.

When the user successfully creates an account and logs in, he/she is presented with the control panel. The control panel lists all the functionality the user can access. This includes the image capture application, the image upload and display application and the push notification module.

3.5 Image Capture and Secure:

Figure 9 Image Capture Prototype

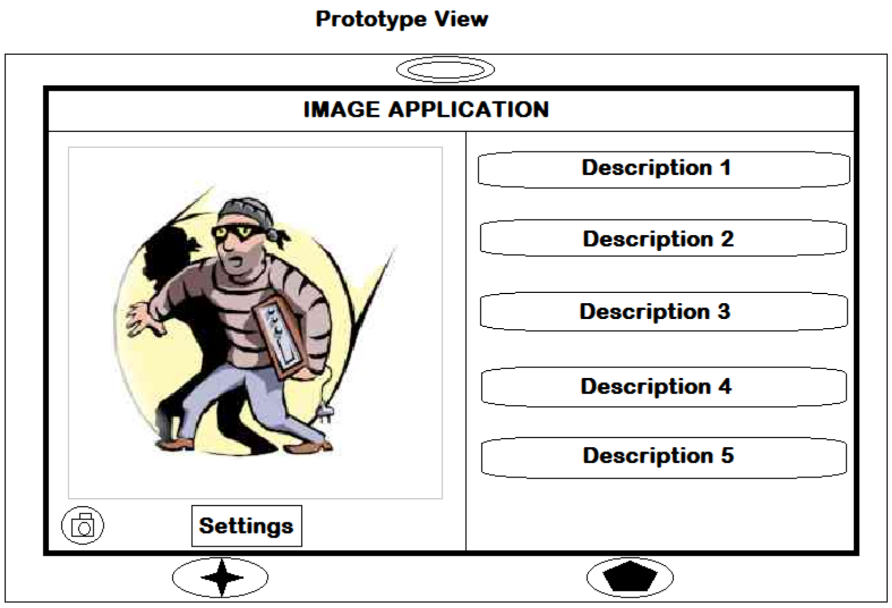


Image capture and secure is a module, which uses a trigger based mechanism to capture and store images.

This module makes use of a custom surface view object, to render and display images. When the images are captured, they are saved in the sd card or the internal memory of the device, by using a mediastore object. When the user clicks on the settings button, the user is given an option to upload or retrieve the images. When the upload option is selected, the images are encrypted and removed from the SD card, when the retrieve option is selected, the decryption algorithm, performs the decryption routine, and displays the images back to the user.

For encrypting the images, the Data Encryption Standard is used. The algorithm copies all the files from the mediastorage object, deletes from the SD card and performs the encryption algorithm. This way, when the user clicks on the encrypted images, he/she will be unable to view the images as they are encrypted.

The UI elements used in this module are as follows.

SurfaceView: This is the area where the selected image will be displayed.

DisplayArea: This part of the screen lists the 5 recent images along with their description.

Settings: This button is present on the bottom of the screen. The settings screen has two buttons Backup and Retrieve. Backup has two options dealing with the encryption and backup of images. Retrieve also has two options dealing with decryption and retrieval of images.

Event Trigger: This button is present on the bottom of the screen. When this button is pressed, the application uses the panel camera to take a picture.

Figure 10 Image Capture Activity Diagram

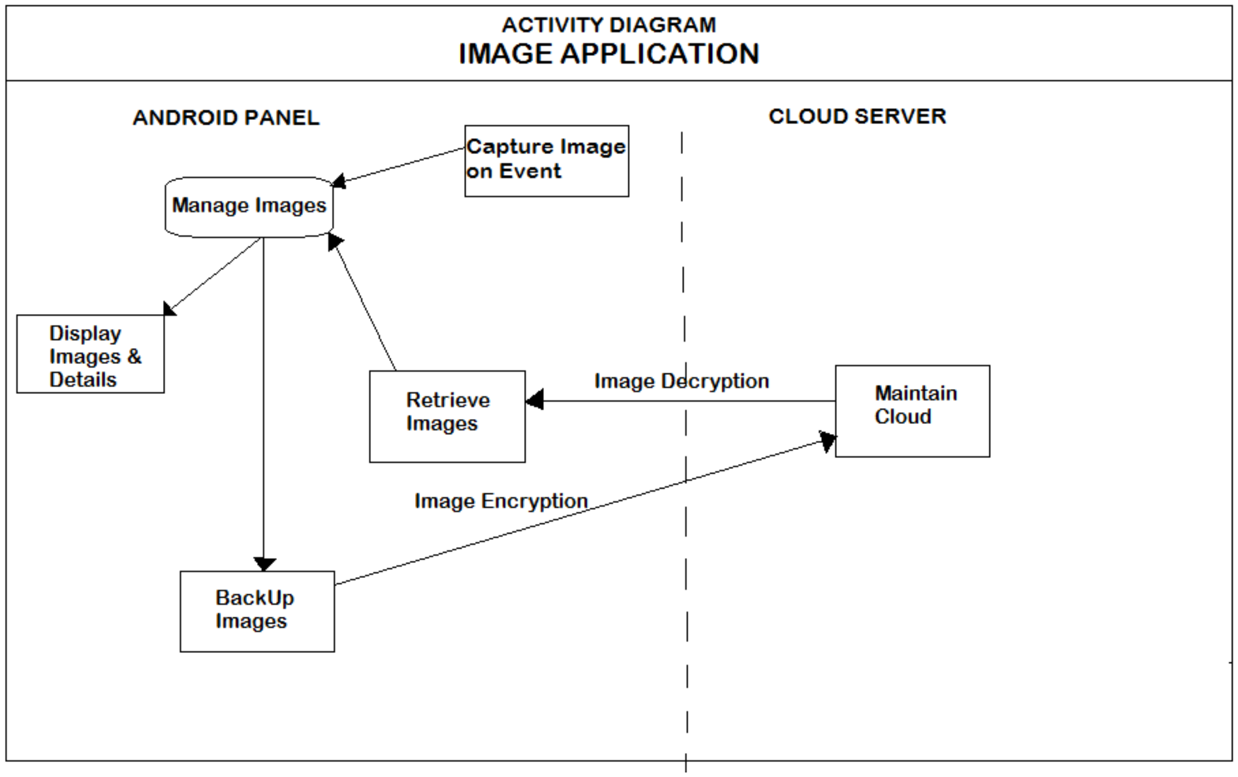
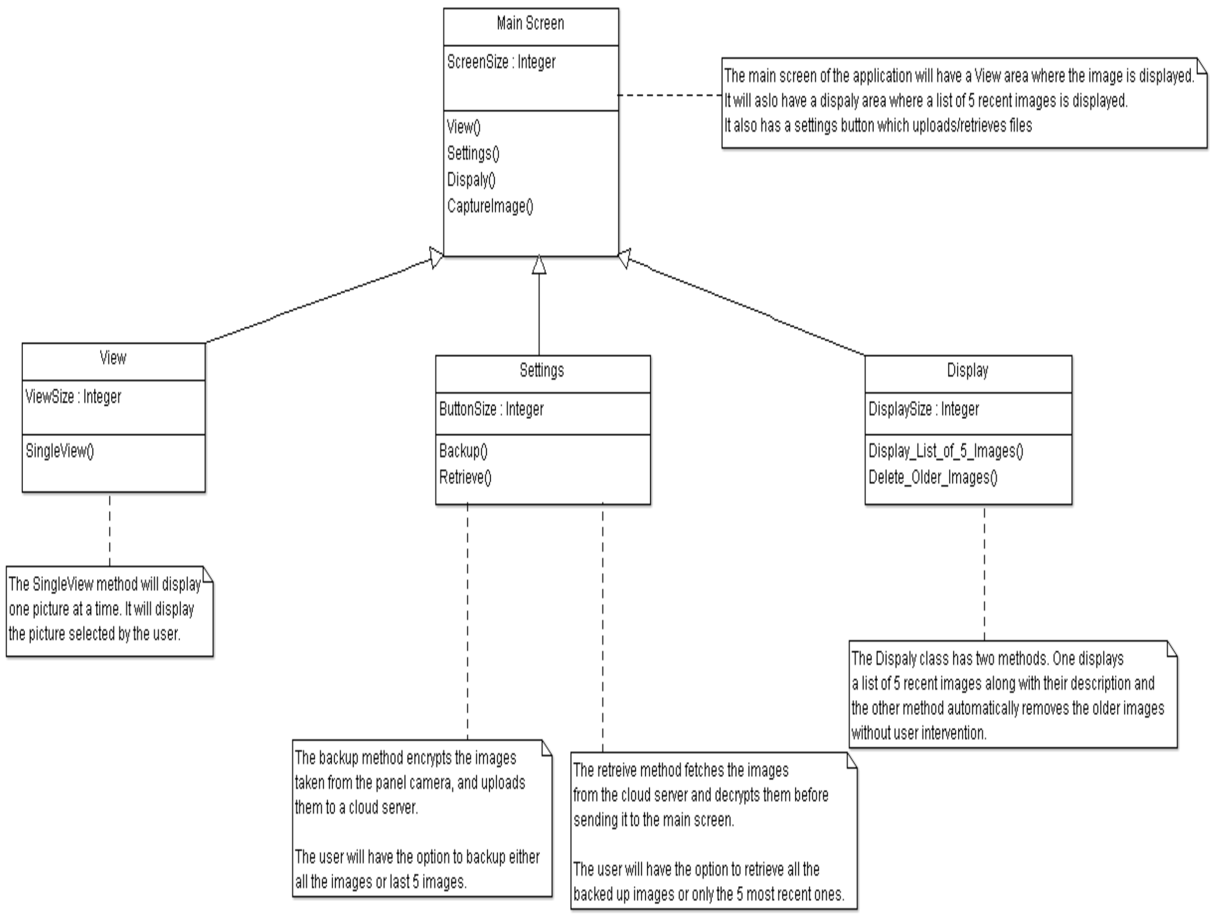
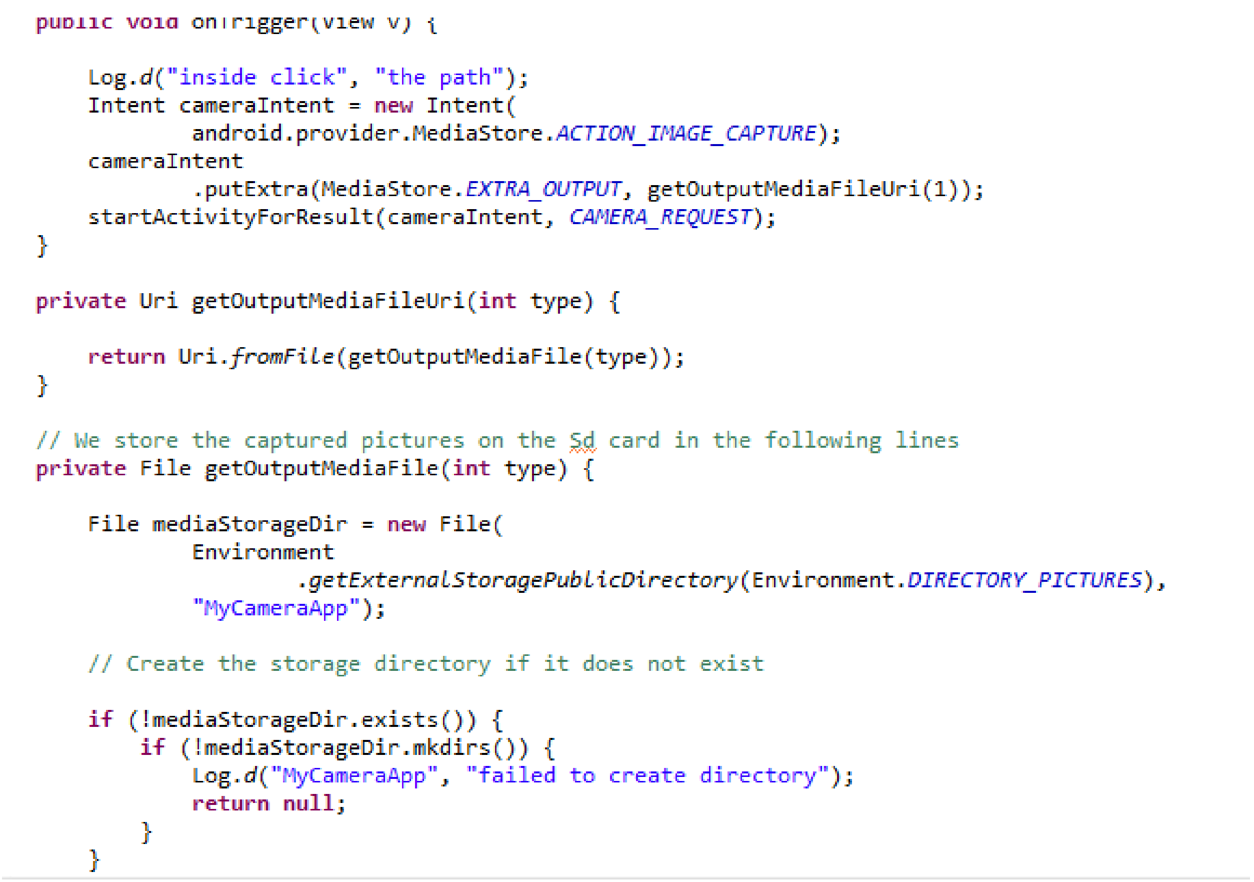


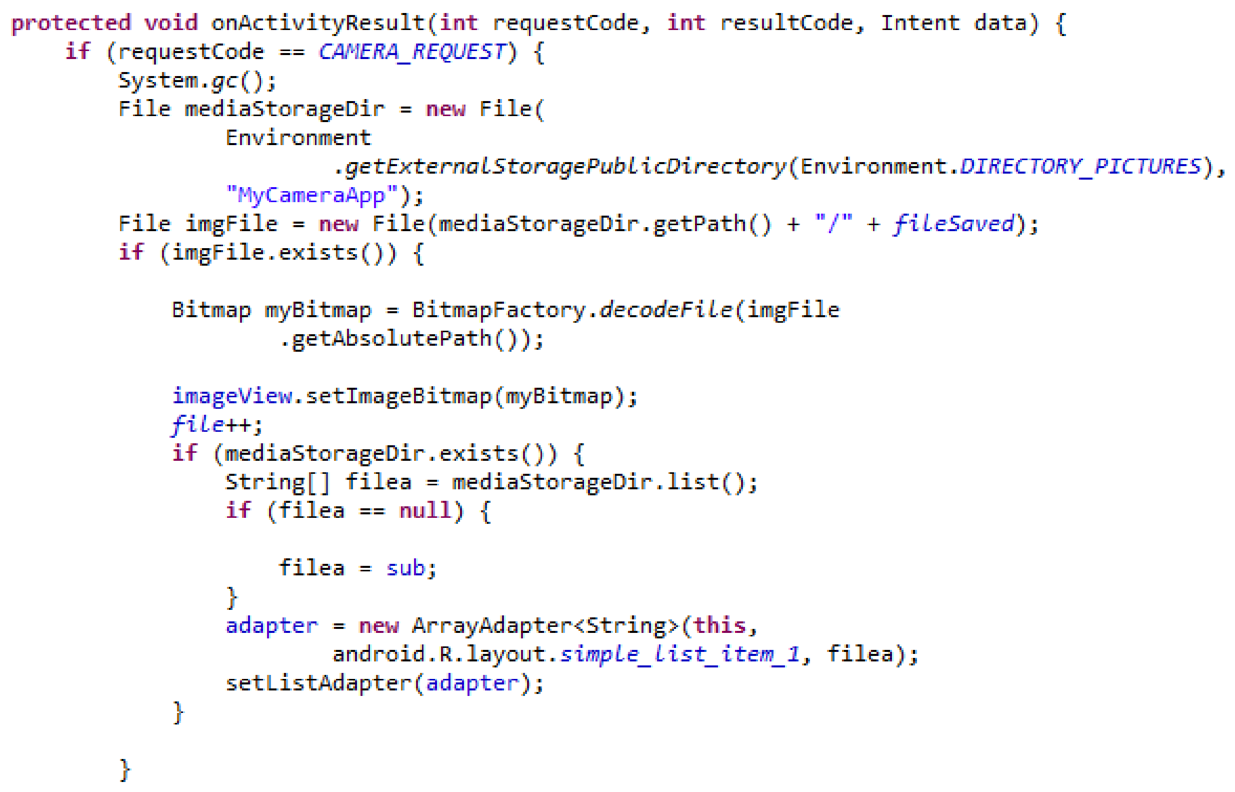
Figure 11 Image Capture Class Diagram


Main Screen
ScreenSize : Integer
View()
Settings()
Dispaly()
CaptureImage()
The main screen of the application will have a View area where the image is displayed.
It will aslo have a dispaly area where a list of 5 recent images is displayed.
It also has a settings button which uploads/retrieves files
View
ViewSize : Integer
SingleView()
Settings
ButtonSize : Integer
Backup()
Retrieve()
Display
DisplaySize : Integer
Display_List_of_5_Images()
Delete_Older_Images()
The SingleView method will display one picture at a time. It will display the picture selected by the user.
The backup method encrypts the images taken from the panel camera, and uploads them to a cloud server.
The user will have the option to backup either all the images or last 5 images.
The retreive method fetches the images from the cloud server and decrypts them before sending it to the main screen.
The user will have the option to retrieve all the backed up images or only the 5 most recent ones.
The Dispaly class has two methods. One displays a list of 5 recent images along with their description and the other method automatically removes the older images without user intervention.

As seen from the above UML diagrams, when the user clicks on the trigger button, the application invokes a surfaceview object, and takes a picture. Once the image is taken, the listview on the right side of the screen is populated with the image and the time stamp. If the user clicks on the image, a bitmap of the image object is created, and is drawn on the imageview. The following snippets, give an idea of how this is carried out.



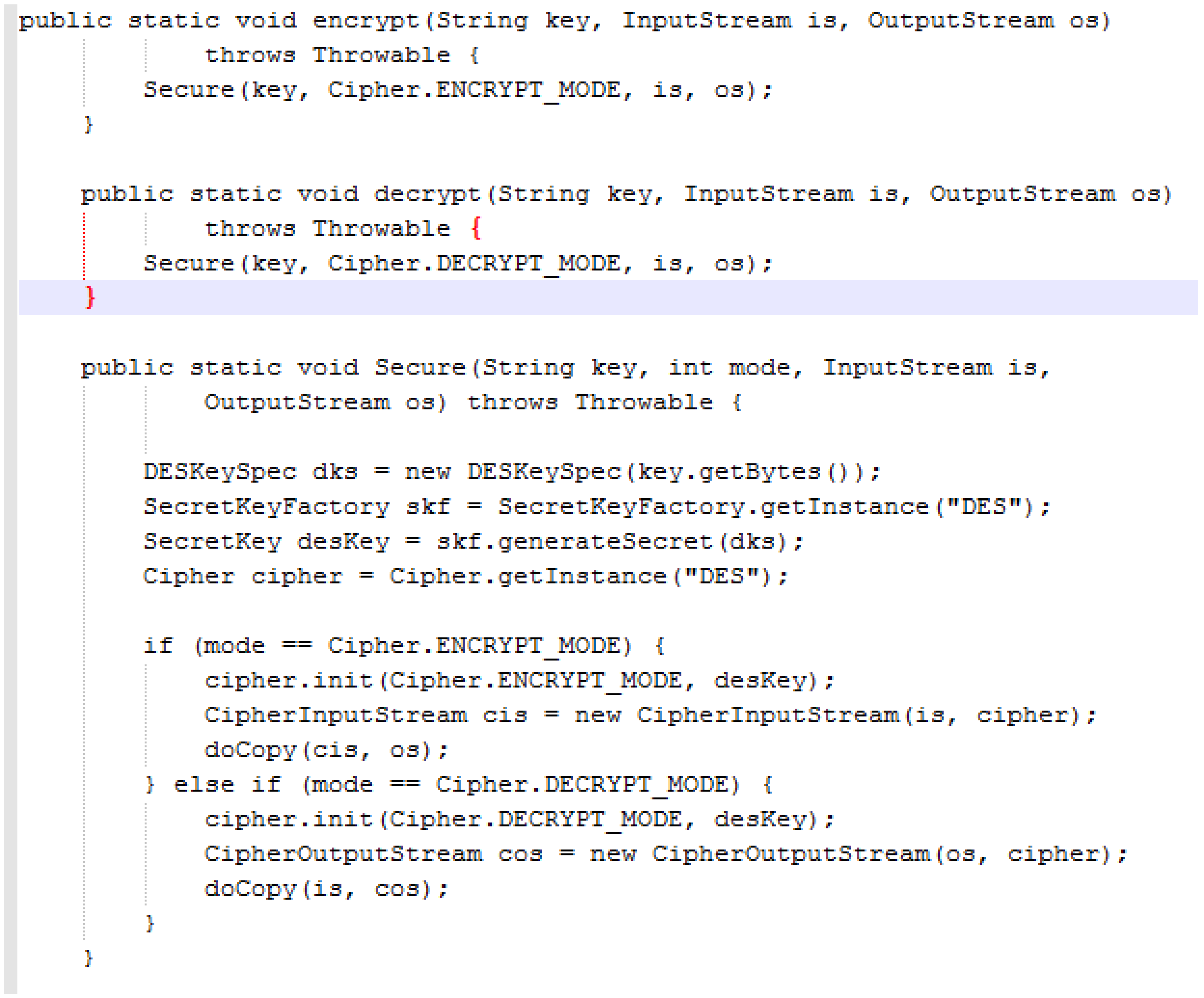
```
public void onTrigger(View v) {

    Log.d("inside click", "the path");
    Intent cameraIntent = new Intent(
            android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent
            .putExtra(MediaStore.EXTRA_OUTPUT, getOutputMediaFileUri(1));
    startActivityForResult(cameraIntent, CAMERA_REQUEST);
}

private Uri getOutputMediaFileUri(int type) {

    return Uri.fromFile(getOutputMediaFile(type));
}

// We store the captured pictures on the Sd card in the following lines
private File getOutputMediaFile(int type) {

    File mediaStorageDir = new File(
            Environment
                    .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
            "MyCameraApp");

    // Create the storage directory if it does not exist

    if (!mediaStorageDir.exists()) {
        if (!mediaStorageDir.mkdirs()) {
            Log.d("MyCameraApp", "failed to create directory");
            return null;
        }
    }
```



```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == CAMERA_REQUEST) {
        System.gc();
        File mediaStorageDir = new File(
                Environment
                        .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
                "MyCameraApp");
        File imgFile = new File(mediaStorageDir.getPath() + "/" + fileSaved);
        if (imgFile.exists()) {

            Bitmap myBitmap = BitmapFactory.decodeFile(imgFile
                    .getAbsolutePath());

            imageView.setImageBitmap(myBitmap);
            file++;
            if (mediaStorageDir.exists()) {
                String[] filea = mediaStorageDir.list();
                if (filea == null) {

                    filea = sub;
                }
                adapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1, filea);
                setListAdapter(adapter);
            }

        }
```

Another functionality offered by this module, is the image encryption and decryption. If the user presses the settings button, or swipes the screen, he/she is given an option to encrypt and hide or decrypt and display the images. When the user swipes the screen, the gesture detection method is called. This method listens to any movement the user makes on the screen and if the specific pattern is detected (in this case swiping the screen) it presents encryption and decryption options to the user. The code snippet for the functions is as follows

```java
String[] encrypt_dest = new String[50];
File files[] = mediaStorageDir1.listFiles();
for (int i = 0; i < 20; i++) {
    encrypt_dest[i] = "ISG_" + i + ".jpg";
}
String[] filem = mediaStorageDir.list();
int[] a = new int[50];
int ip = 0;
if (filem != null && filem.length >= 1) {
    int x = Integer.parseInt(filem[0].substring(4,
            filem[0].indexOf('.')));
    ;
    for (int i = 0; i < filem.length; i++) {
        int index = filem[i].indexOf('.');
        a[i] = Integer.parseInt(filem[i].substring(4, index));
        if (a[i] > x) {
            x = a[i];
        }
    }
    ip = x + 1;
} else {
    ip = 1;
}
for (File f : files) {
    FileInputStream fis2 = new FileInputStream(f);
    FileOutputStream fos2 = new FileOutputStream(mediaStorageDir
            + "/" + encrypt_dest[ip++]);
    decrypt(key, fis2, fos2);
}
```



```
public static void encrypt(String key, InputStream is, OutputStream os)
            throws Throwable {
        Secure(key, Cipher.ENCRYPT_MODE, is, os);
    }

    public static void decrypt(String key, InputStream is, OutputStream os)
            throws Throwable {
        Secure(key, Cipher.DECRYPT_MODE, is, os);
    }

    public static void Secure(String key, int mode, InputStream is,
            OutputStream os) throws Throwable {

        DESKeySpec dks = new DESKeySpec(key.getBytes());
        SecretKeyFactory skf = SecretKeyFactory.getInstance("DES");
        SecretKey desKey = skf.generateSecret(dks);
        Cipher cipher = Cipher.getInstance("DES");

        if (mode == Cipher.ENCRYPT_MODE) {
            cipher.init(Cipher.ENCRYPT_MODE, desKey);
            CipherInputStream cis = new CipherInputStream(is, cipher);
            doCopy(cis, os);
        } else if (mode == Cipher.DECRYPT_MODE) {
            cipher.init(Cipher.DECRYPT_MODE, desKey);
            CipherOutputStream cos = new CipherOutputStream(os, cipher);
            doCopy(is, cos);
        }
    }
```

3.6 Image Upload and Display:

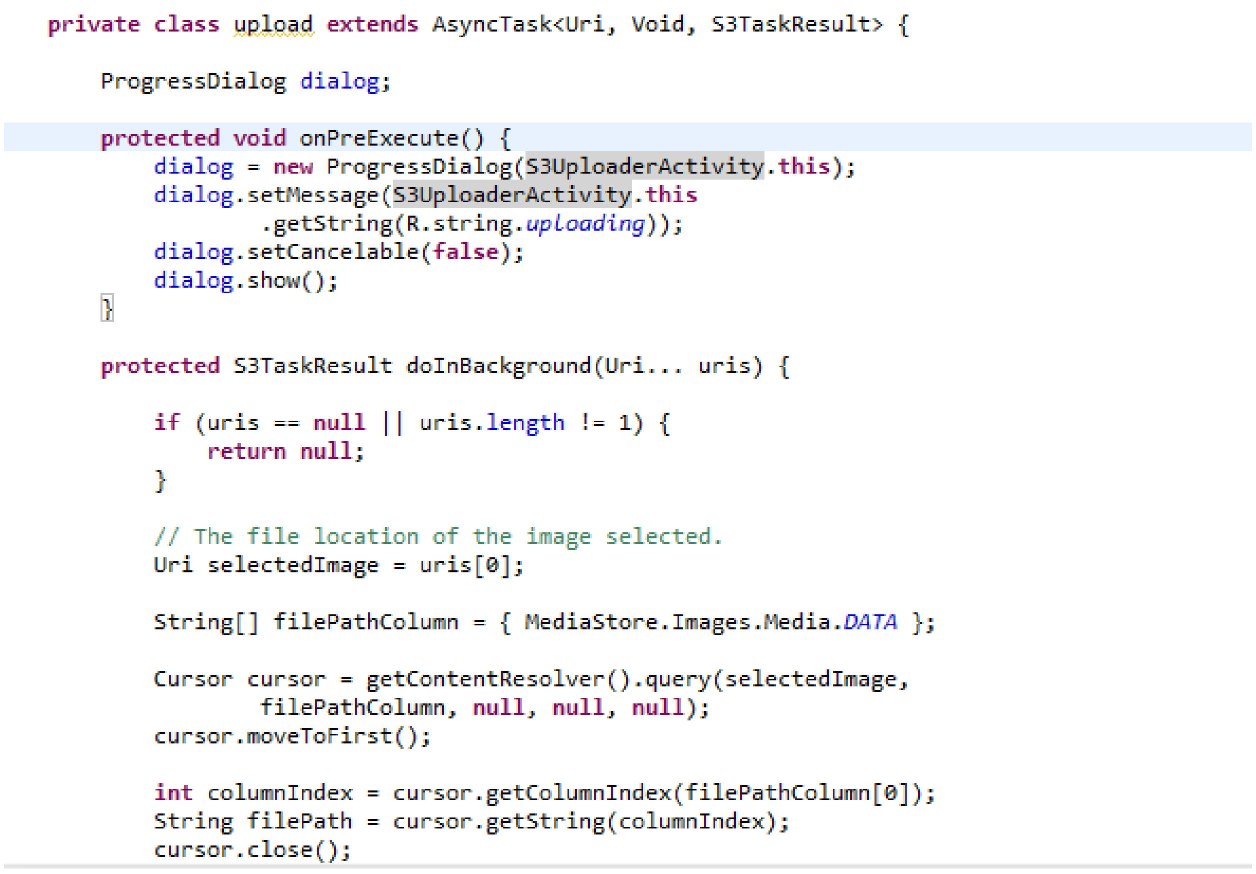
Image upload and display is a module which uses Amazon Simple Storage Service to upload pictures online and also allow the users, to display it on their mobile devices. This module builds on the image capture module. After taking the pictures , the user can either send them to the server and get a public url which can be used to display the image, or upload the encrypted image which is then stored on the bucket.

First the application creates a client object and passes the credentials file and properties as parameters. When this object connects to Amazon S3, the client authenticates itself, by passing the above information. I have written an image picker activity to select the images. The image picture activity upon selection of the image, calls the onActivityResult method. This method returns an uri object, and calls the upload method.

By using an ImagePicker activity, I optimize the code because the callback action is automatically called, and I do not have to manually copy the output of the bitmap object and process it. Since this is a native android object, the memory utilization is also minimal.

The upload method uses an algorithm, which uses an asynchronous upload mechanism, to upload the files to the server. This method takes the uri object as the input and uploads it to the server, generating a url object. Each url object, can be accessed using a pre-signed link. Pre-signing the url allows the users to access the object without specifying the credentials. The user can then click this link to display the picture online using a browser window. This link can also be shared with other users. This generated link has an expiration time. Once the timer expires, the link will no longer work.

The following code snippet shows the upload method, which is an asynchronous task, getting the image as the bitmap uri and a cursor object to select the image and point the file path.



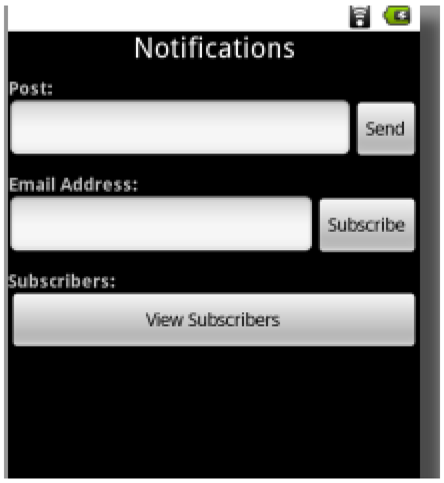
```java
private class upload extends AsyncTask<Uri, Void, S3TaskResult> {

    ProgressDialog dialog;

    protected void onPreExecute() {
        dialog = new ProgressDialog(S3UploaderActivity.this);
        dialog.setMessage(S3UploaderActivity.this
                .getString(R.string.uploading));
        dialog.setCancelable(false);
        dialog.show();
    }

    protected S3TaskResult doInBackground(Uri... uris) {

        if (uris == null || uris.length != 1) {
            return null;
        }

        // The file location of the image selected.
        Uri selectedImage = uris[0];

        String[] filePathColumn = { MediaStore.Images.Media.DATA };

        Cursor cursor = getContentResolver().query(selectedImage,
                filePathColumn, null, null, null);
        cursor.moveToFirst();

        int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
        String filePath = cursor.getString(columnIndex);
        cursor.close();
```

3.7 Push Notifications:

This module's primary use will be to poll users about any information related to the security system and allows the developer to send messages to the users.

When the user successfully logs in, the user information is parsed and sent to the server. The server then displays the control panel to the user. The user clicks on Administrative controls, and this opens the screen for the notification system.

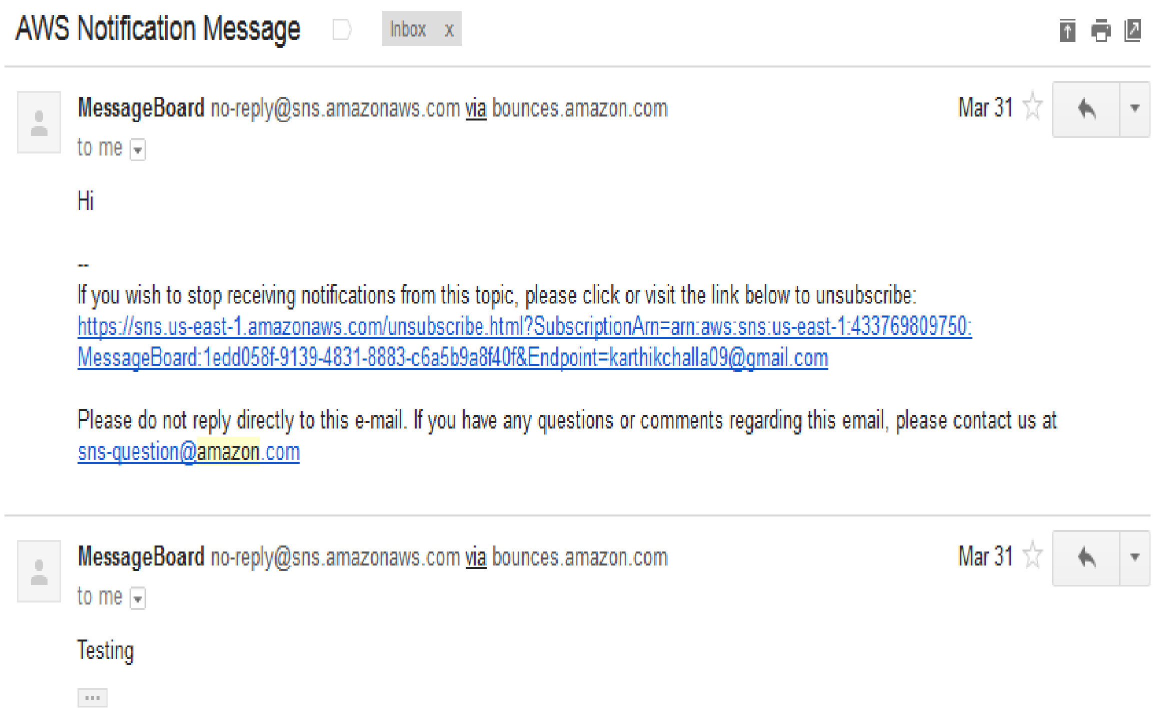
Figure 12 Push Notifications



As seen in the above picture, first the administrator adds users by entering their email address and clicking on subscribe button. This sends a verification email to the users which they click on to accept the subscription. Once the user subscribes to a topic, the administrator can send them notifications by posting a message and clicking on send. Clicking on View Subscribers, displays the list of subscribed users.

I have implemented this module, by using the API's provided by Amazon Simple Notification System. First I create an instance of the Amazon SNS client, and the SQS server by passing the ACCESS_ID and the secret key as the parameters. To post a message, the topicARN and the message contents are sent as parameters to the publish method. Each message sent by the developer, is added to the queue. This queue makes use of an array list, to process the messages and to send them to the amazon server for posting. queueARM methods are used to set up queue policies and to retrieve. To send messages successfully, a queue policy needs to be defined. I have used a hashmap to store the subscriber names and the topicARN.

Figure 13 Notification Example


AWS Notification Message

Inbox x

MessageBoard no-reply@sns.amazonaws.com via bounces.amazon.com Mar 31

to me

Hi

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:433769809750:MessageBoard:1edd058f-9139-4831-8883-c6a5b9a8f40f&Endpoint=karthikchalla09@gmail.com

Please do not reply directly to this e-mail. If you have any questions or comments regarding this email, please contact us at sns-question@amazon.com

MessageBoard no-reply@sns.amazonaws.com via bounces.amazon.com Mar 31

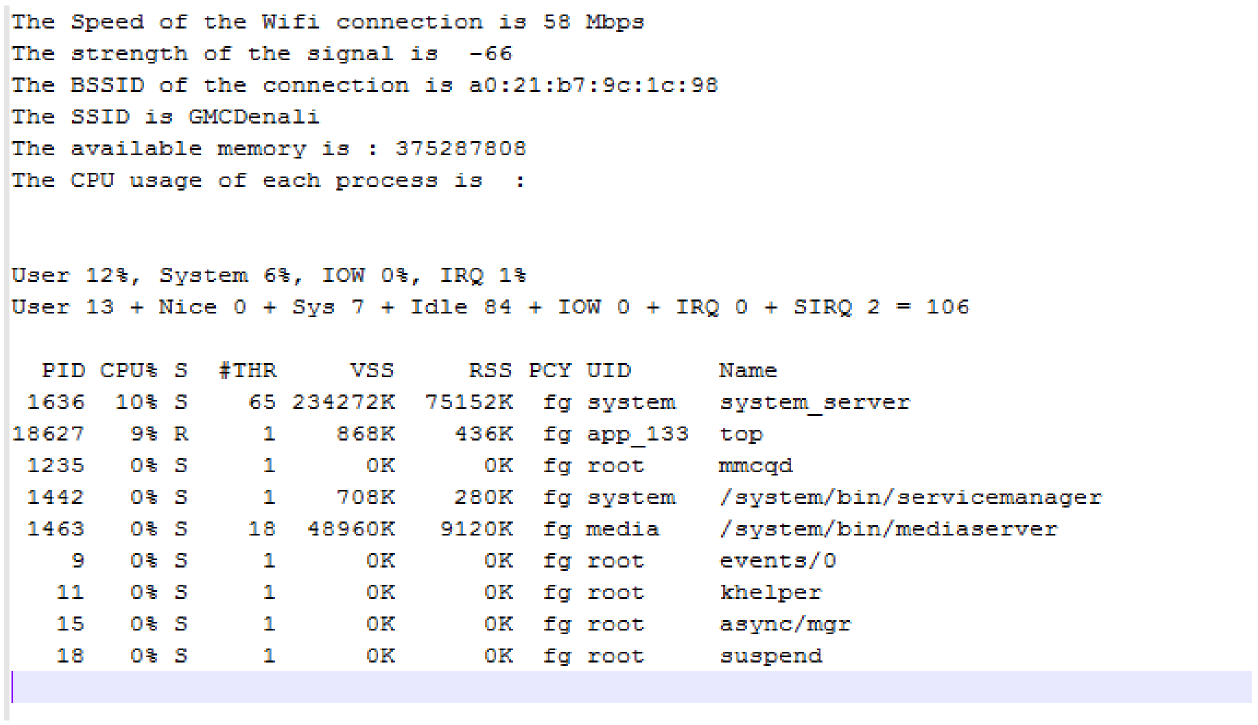
to me

Testing

...

3.8 Diagnostic Monitoring:

This module's primary purpose is to monitor and log all the information about the system's state. This module monitors all the information related to wifi connectivity including the signal strength, RSSID, SSID, speed and ip address. It also displays a list of all the active processes along with their cpu usage and memory usage. It also displays the available system memory. Finally it logs all this information and stores it in the cloud server.



```
The Speed of the Wifi connection is 58 Mbps
The strength of the signal is  -66
The BSSID of the connection is a0:21:b7:9c:1c:98
The SSID is GMCDenali
The available memory is : 375287808
The CPU usage of each process is  :


User 12%, System 6%, IOW 0%, IRQ 1%
User 13 + Nice 0 + Sys 7 + Idle 84 + IOW 0 + IRQ 0 + SIRQ 2 = 106

  PID CPU% S  #THR     VSS     RSS PCY UID      Name
 1636  10% S    65 234272K  75152K  fg system   system_server
18627   9% R     1    868K    436K  fg app_133  top
 1235   0% S     1      0K      0K  fg root     mmcqd
 1442   0% S     1    708K    280K  fg system   /system/bin/servicemanager
 1463   0% S    18  48960K   9120K  fg media    /system/bin/mediaserver
    9   0% S     1      0K      0K  fg root     events/0
   11   0% S     1      0K      0K  fg root     khelper
   15   0% S     1      0K      0K  fg root     async/mgr
   18   0% S     1      0K      0K  fg root     suspend
```

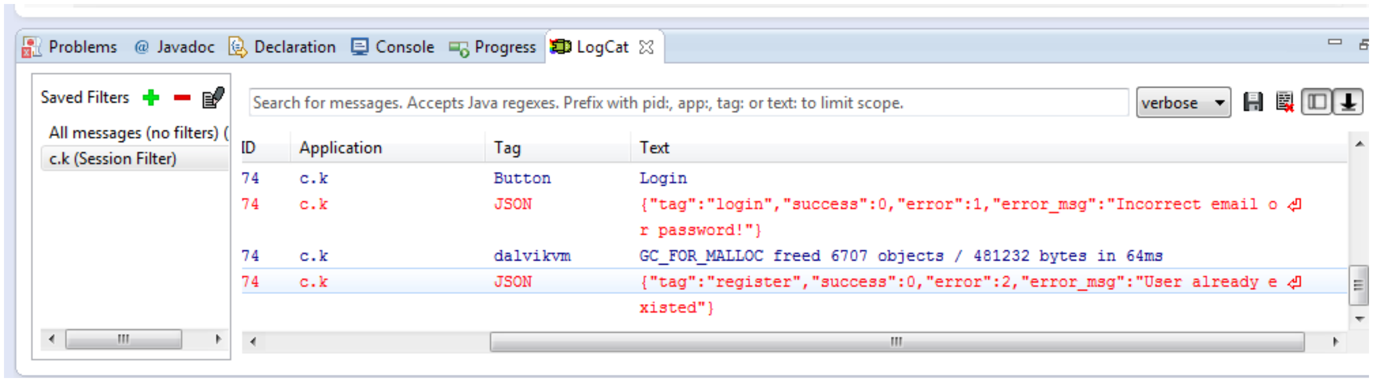
4. Evaluations and Testing:

Since this is a complex application with of modules running and operating across multiple domains, stability and robustness was a primary concern. Since the communication is between the client and server, the communication between the client and server and the message passing between them is of major importance.

The JSON responses generated in response to events, have message codes corresponding to if the operation being performed was a success or failure. These responses help in diagnosing any issues a user might run into.

Response Code Error: This code is returned if the user tries to create an account with already existing credentials. As seen in the below message, register has a response of 2 meaning that the registration failed, followed by the message details.

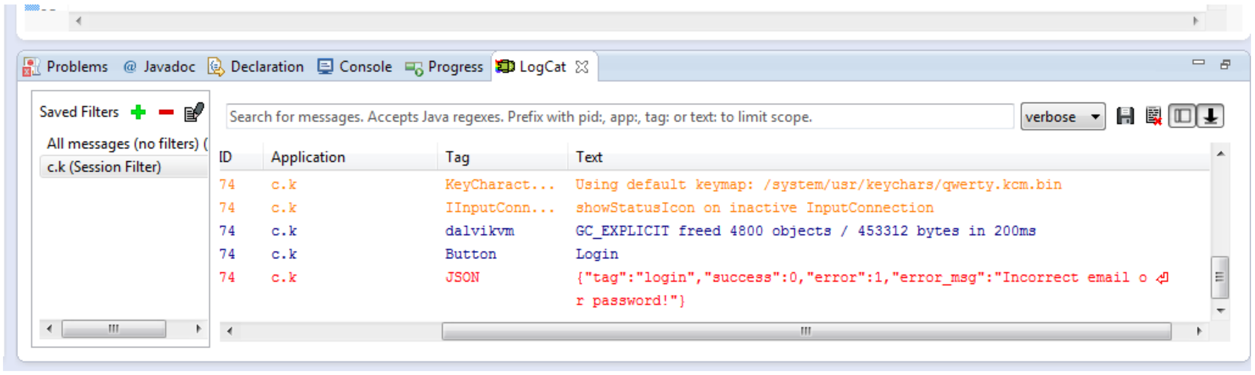
Figure 14 Testing : Error Response Code - Register


Problems
Javadoc
Declaration
Console
Progress
LogCat
Saved Filters
All messages (no filters) (
c.k (Session Filter)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

```
ID   Application   Tag        Text
74   c.k           Button     Login
74   c.k           JSON       {"tag":"login","success":0,"error":1,"error_msg":"Incorrect email o
                              r password!"}
74   c.k           dalvikvm   GC_FOR_MALLOC freed 6707 objects / 481232 bytes in 64ms
74   c.k           JSON       {"tag":"register","success":0,"error":2,"error_msg":"User already e
                              xisted"}
```

Response Code Error: This code is returned if the user tries to login with incorrect user name and email details. As seen in the below message, login has a response of 2 meaning that the login failed, followed by the message details.

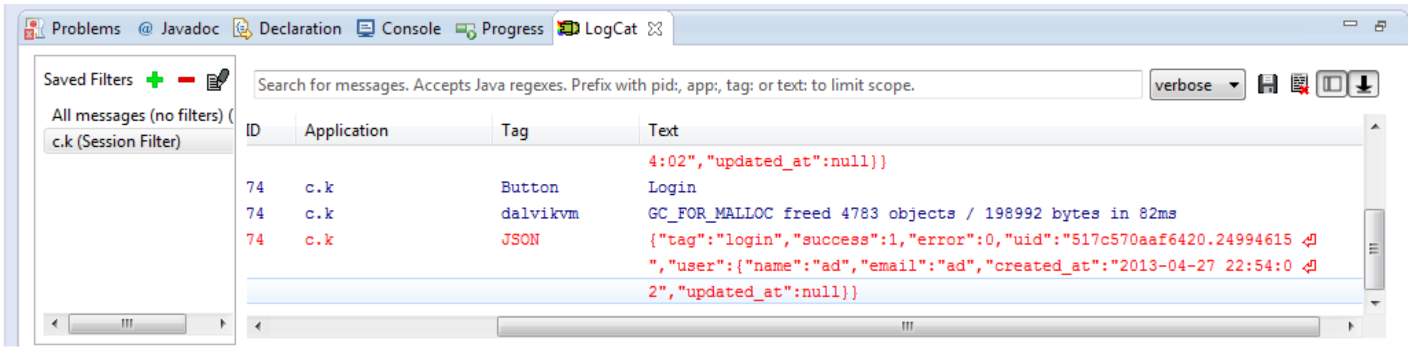
Figure 15 Testing Error Response Code - Login


Problems
Javadoc
Declaration
Console
Progress
LogCat
Saved Filters
All messages (no filters)
c.k (Session Filter)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

```
ID  Application  Tag            Text
74  c.k          KeyCharact...  Using default keymap: /system/usr/keychars/qwerty.kcm.bin
74  c.k          IInputConn...  showStatusIcon on inactive InputConnection
74  c.k          dalvikvm       GC_EXPLICIT freed 4800 objects / 453312 bytes in 200ms
74  c.k          Button         Login
74  c.k          JSON           {"tag":"login","success":0,"error":1,"error_msg":"Incorrect email o
                                r password!"}
```

Response Code – Success: This code is returned if the user login was successful. Along with the success tag, the message also returns the time stamp at which the account was logged in.

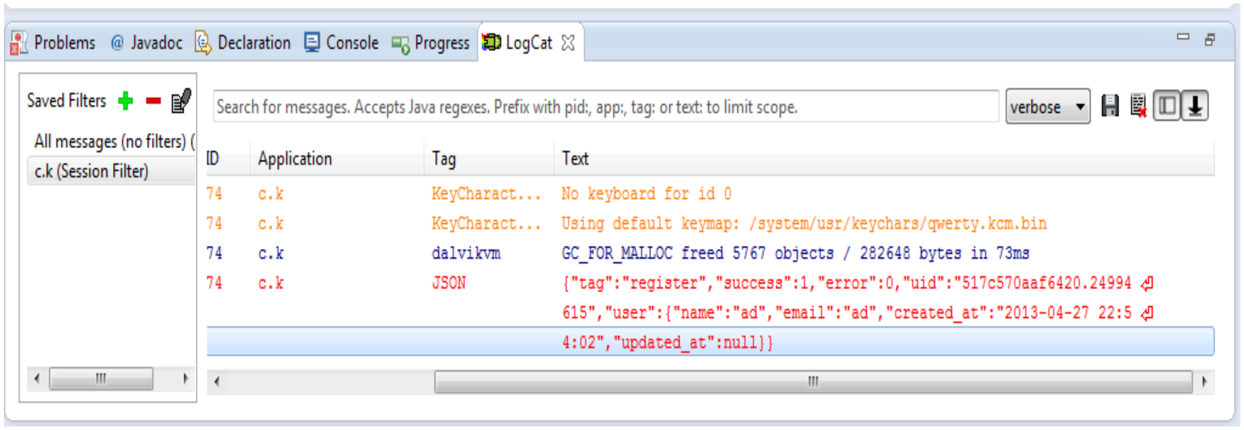
Figure 16 Testing Success Response Code - Login


Problems
Javadoc
Declaration
Console
Progress
LogCat
Saved Filters
All messages (no filters)
c.k (Session Filter)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

```
ID  Application  Tag       Text
                           4:02","updated_at":null}}
74  c.k          Button    Login
74  c.k          dalvikvm  GC_FOR_MALLOC freed 4783 objects / 198992 bytes in 82ms
74  c.k          JSON      {"tag":"login","success":1,"error":0,"uid":"517c570aaf6420.24994615
                           ","user":{"name":"ad","email":"ad","created_at":"2013-04-27 22:54:0
                           2","updated_at":null}}
```

Response Code – Success: This code is returned if the user registration was successful. Along with the success tag, the message also returns the time stamp at which the account was created.

Figure 17 Testing Success Response Code - Register


Problems
Javadoc
Declaration
Console
Progress
LogCat
Saved Filters
All messages (no filters) (
c.k (Session Filter)
Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.
verbose

```
ID  Application  Tag            Text
74  c.k          KeyCharact...  No keyboard for id 0
74  c.k          KeyCharact...  Using default keymap: /system/usr/keychars/qwerty.kcm.bin
74  c.k          dalvikvm       GC_FOR_MALLOC freed 5767 objects / 282648 bytes in 73ms
74  c.k          JSON           {"tag":"register","success":1,"error":0,"uid":"517c570aaf6420.24994
                                615","user":{"name":"ad","email":"ad","created_at":"2013-04-27 22:5
                                4:02","updated_at":null}}
```

To verify the client side functionality and general application behavior in android, I have written several JUnit test cases to validate the behavior.

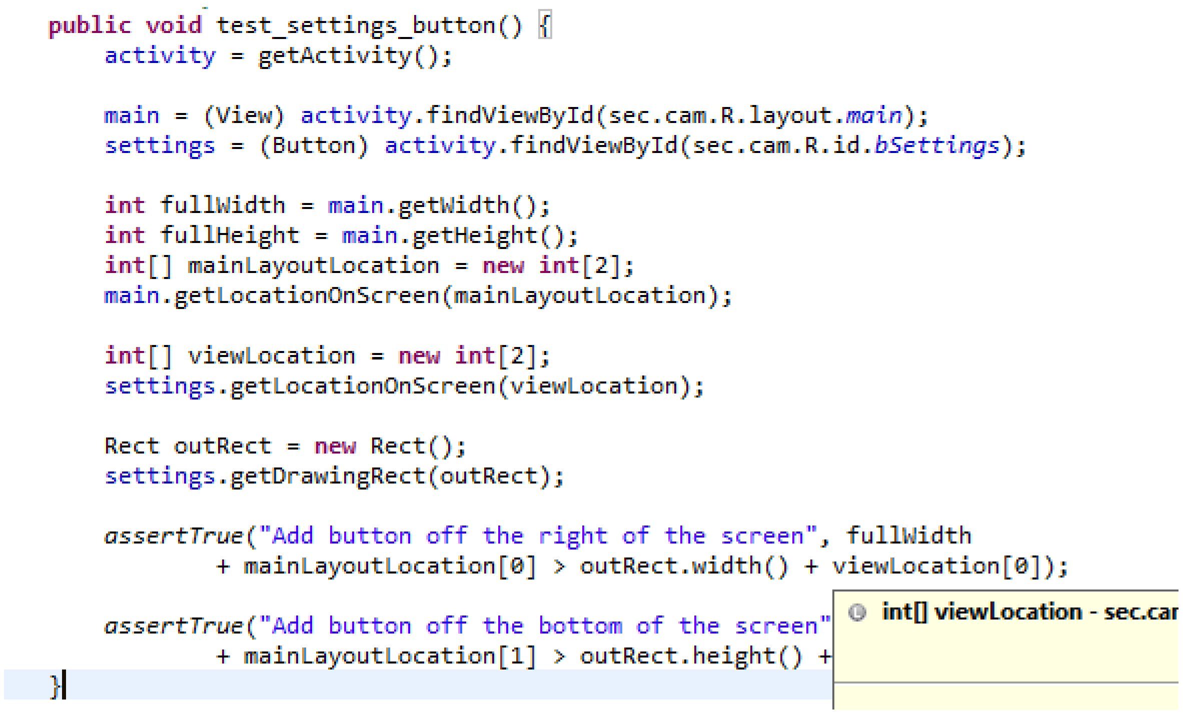
The following test case is used for verifying if any of the created views are empty or not. If they are empty, then the test fails implying that the methods are incorrect.
The pseudo code for the test cases is as follows:

```
Testconditions () {

        assertNull(view description);
        assertNotNull(button info);
        assertNotNull(test condition);

    }
```

The following test case is used to verify if the orientation of the application

```
TestOrientation {

        Activity _get_orientationinfo()

        Assertequals(activity.getconfig().getorient())

}
```

The following is a code sample of the junit test case testing the placement of the settings button.



```
public void test_settings_button() {
    activity = getActivity();

    main = (View) activity.findViewById(sec.cam.R.layout.main);
    settings = (Button) activity.findViewById(sec.cam.R.id.bSettings);

    int fullWidth = main.getWidth();
    int fullHeight = main.getHeight();
    int[] mainLayoutLocation = new int[2];
    main.getLocationOnScreen(mainLayoutLocation);

    int[] viewLocation = new int[2];
    settings.getLocationOnScreen(viewLocation);

    Rect outRect = new Rect();
    settings.getDrawingRect(outRect);

    assertTrue("Add button off the right of the screen", fullWidth
            + mainLayoutLocation[0] > outRect.width() + viewLocation[0]);

    assertTrue("Add button off the bottom of the screen"
            + mainLayoutLocation[1] > outRect.height() +
}
```

5. **Conclusion:**

Cloud computing has been making significant inroads in the domain of application development. The fact that companies can rely on huge amounts of storage and computational ability without worrying about scalability and availability of resources has brought a lot of visibility to the cloud platform.

This project has allowed me developed a framework working with the android platform and the cloud platform to leverage the benefits of both to provide a seamless integrated solution, addressing the limitations of both the platforms. For my project, I have combined the portability and optimization of mobile devices with the computational resources and storage capabilities of cloud servers bringing together a solution which can easily be expanded to provide multiple functionalities depending on the domain.

Working on this project, has allowed me to work with and learn multiple development environments like Android, which is one of the fastest growing operating systems, PHP for the server side development and Amazon Web Services specifically Amazon S3, RDS, EC2 and SNS to provide seamless integration between the client and server along with JUnit framework for testing.

Working on this project, was very challenging as I had to handle multiple design issues. One of the first design issues that I encountered was that the server side database was MySQL and the android client was only able to handle SQLite queries. To facilitate communication and message passing between the two, I have come up with JSON encoding and TAG responses. These messages are interpreted at both the client and server side, and the responses are transferred between the two.

Another significant challenge was to seamlessly integrate the multiple Amazon Web Services with the android client. Because the project involved communications between different services, and the client was an android application, memory usage and application stability was a primary concern. Because images and the mobile camera are used significantly, I have optimized my code efficiently to handle memory issues by detecting when the application loses focus and closing the background instances, loading classes on demand rather than at application launch, restricting access to the device by using permissions and not keeping sessions active for long.

Because of a multitude of services working in parallel, testing the project to diagnose and fix issues required me to build a comprehensive test methodology. For the server side testing, I have written JSON encoded responses to get the state and diagnose any server side issues, and for the android client, I have written multiple JUnit tests along with logs to test the client side of the application.

Because the project involved cloud computing, privacy of the end user and security of the user data was important, hence I worked on restricting access to the cloud servers, hardening the servers and client side security. To do this, I have encrypted the user data and to avoid sending the data in the clear, I have used a secure hash algorithm to generate hash computations of the user credentials; I then send the hash values to the server, where it is compared with the hash value of the saved credential. If there is a match, then the user is allowed access. This way I ensure the integrity of the credentials without compromising the security.

The future work for this project will be to integrate the application with the Qolsys Home Security System, and work on developing additional features to further improve the product.

6. References:

1. National Institute of Standards and Technology – Cloud Computing [1]

   http://www.nist.gov/itl/cloud/

2. Cloud Computing Bible – Barrie Sosinsky [2]

   http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470903562.html

3. Infrastructure As a Service – Margaret Rouse [3]

   http://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS

4. Amazon Elastic Compute Cloud (EC2) – Amazon [4]

   http://aws.amazon.com/ec2/

5. A View of Cloud Computing – U.C.BerklEY
   http://dl.acm.org/citation.cfm?id=1721672

6. Cloud Computing: Distributed Internet Computing for IT and Scientific Research - Dikaiakos, M.D.
http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5233607&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5233607

7. Amazon Web Services – Amazon

http://aws.amazon.com/

8. Google Android Development – Google

http://developer.android.com/index.html