High Level Design Document
Mobile Security Solution in Cloud
Team 31

Tarun Gupta
201403002

Sharvil Katariya
201301129

Chinmay Patel
201405627

Kavya Nerella
201301121

November 2015

Contents

1 Abstract

To design an end-to-end solution for mobile security on cloud that facilitates:

- Storing images (taken automatically through mobiles for every one minute) in a scalable storage.
- Storing Meta info about the image such as:
  - Image Timestamp – to track the time at which the image is captured by the device.
  - Mobile Location – to track the user's location.
  - User name – to track the user's info.
- Fetching the images via a web page based on user query customized using the image Meta info.

2 Introduction

Fueled by the widespread adoption of mobile devices and the explosion of mobile applications, mobile device security has become a critical issue. This mobile security solution would help the user to enhance the security of a location by capturing the photograph every minute. Therefore, in this project, we plan to design a cloud framework that allows mobile phones to post images on a regular basis and also retrieve the images to the end user with minimal delay. We will also provide a web interface for the easy filtering of the images.

3 Scope

The android application can be used through any android device that supports camera, take a photo every minute upload it on cloud. Web based album can be used to view the captured pictures. User can use the search option to search the stored pictures by time and device.

4 Overview

4.1 Intended Audience

This application can be used by security personals for any public or private surveillance systems.

4.2 User Benefits

- Financial : Low cost security solution (Costs for video surveillance is reduced)
- Storage : Storage requirements of storing a video is more than the storage requirement of storing an image. Thus this solution will help save a lot of storage space, hence money saving.

4.3 Proposed Framework

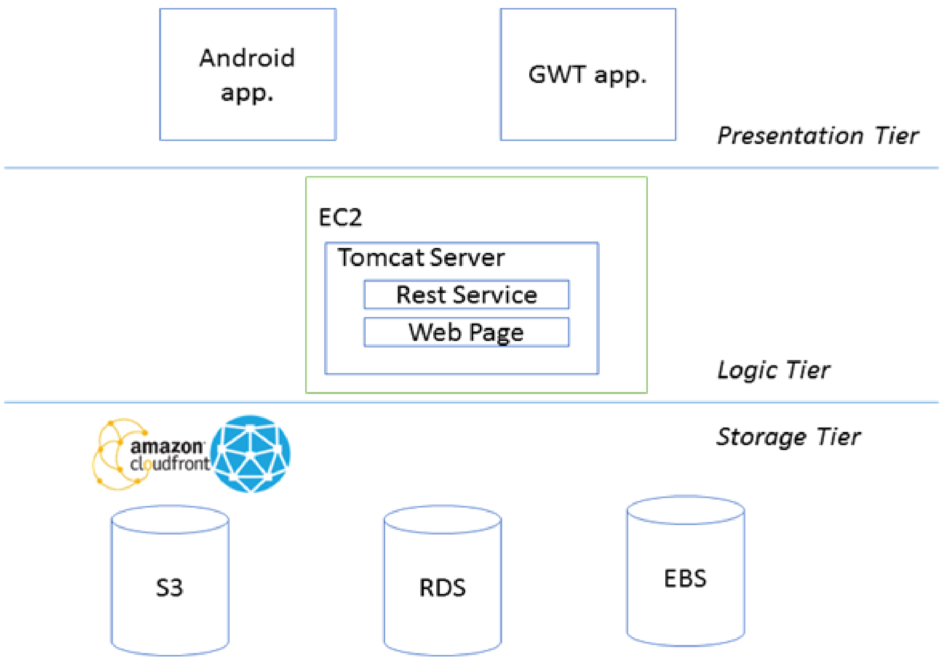

Figure 1: High level Design

5 Detailed Design

5.1 Cloud Services

We will be using services provided by Amazon i.e. AWS (Amazon Web Services), explained below:

5.1.1 Amazon Simple Storage Service (Amazon S3) [1]

Amazon S3 is a web service that enables us to store data in the cloud. It provides developers with secure, durable, highly-scalable object storage. We will be using S3 REST API's to upload, download and filter images. S3 will store pictures in a bucket with device name followed by session identifier.

- Bucket URL :
  `http://mobilepics018.s3.amazonaws.com/`
- Bucket URL with CloudFront :
  `http://d1o4b71ila6msg.cloudfront.net/`
- Access : Public

5.1.2 Amazon Relational Database Service (Amazon RDS) [2]

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. We will be using RDS API's to interact with the database. RDS will store device info, session info, etc. which are required by the REST APIs. We have used RDS services under free tier as per the following configuration:

- Engine : MySQL 5.6.23
- Storage : 20 GB
- Availability Zone : us-west-2b (Oregon)
- Configuration : 1 VCPU, 1GB.
- Endpoint :
  `cloudinstance.c2db2p6ssgi7.us-west-2.rds.amazonaws.com:3306`
- Database Name : cloudinstance

5.1.3 Amazon Elastic Compute Cloud (Amazon EC2) [3]

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. We will be hosting our web server on a virtual machine running in cloud. We have used EC2 services under free tier as per the following configuration:

- Public DNS -
  `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com`
- Availability zone – SouthEast (Singapore)

- Elastic Block Storage size – 30 GB
- Amazon Machine Image – Amazon Linux AMI (HVM) 3.14 Kernel
- Tomcat Server is running and is responsible for handling requests from web application and Android application. `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080`

5.2 Logic Tier

5.2.1 Use Case Diagram

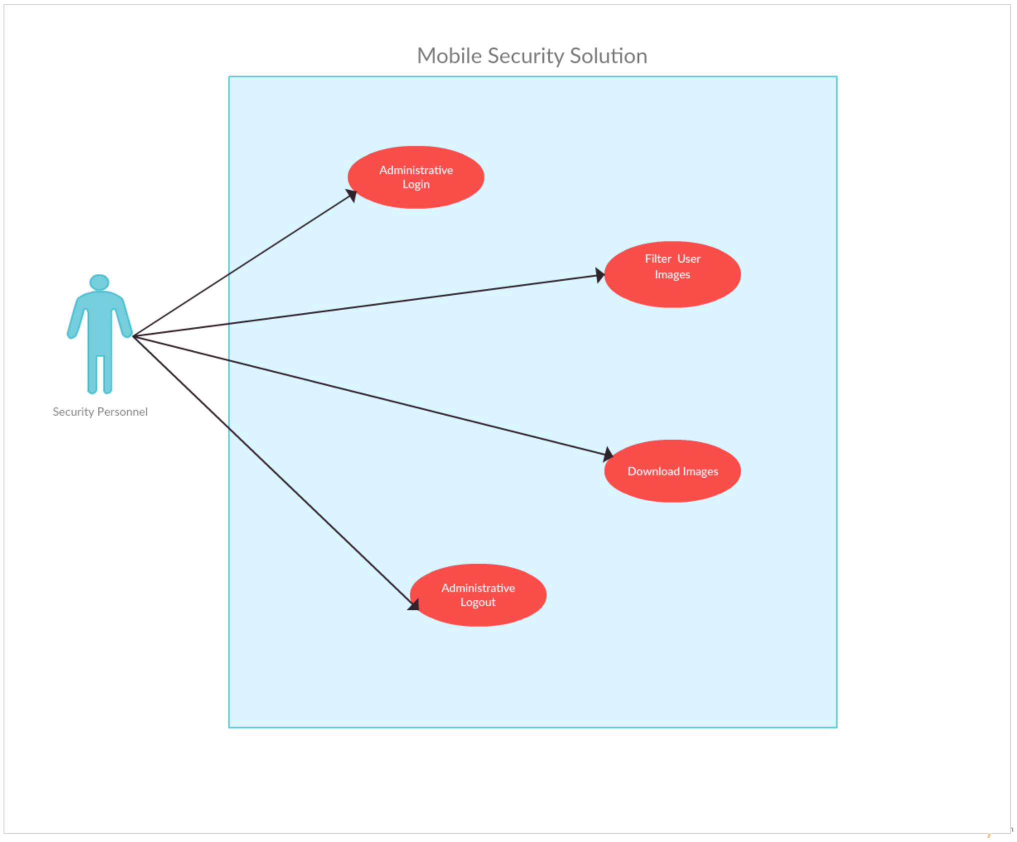

Figure 2: Use Case Diagram

5.2.2 Application Flow

- Android Application: This application captures and uploads photos periodically to cloud. This application will be developed completely in Java, using Android SDK. The application automatically starts capturing pictures, and uploading them periodically to cloud. It makes use od REST calls to upload images to S3.
- Web Page: This is a front end web application, used by security administrator to view images stored in cloud, in gallery form. This service

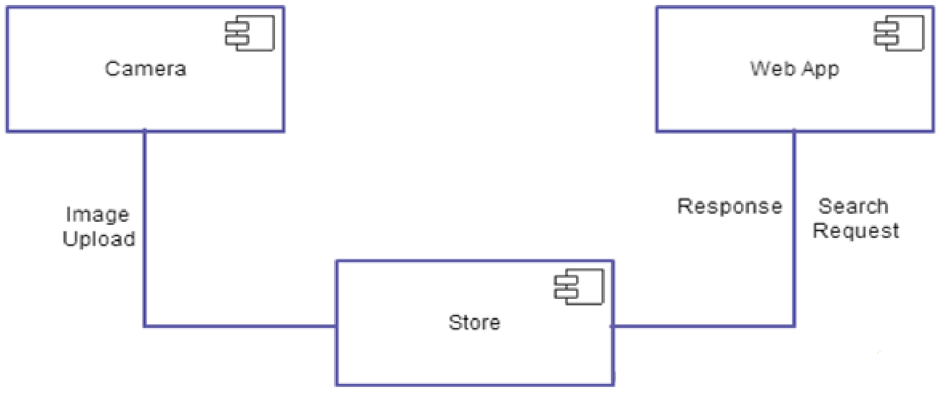

Figure 3: Component Diagram

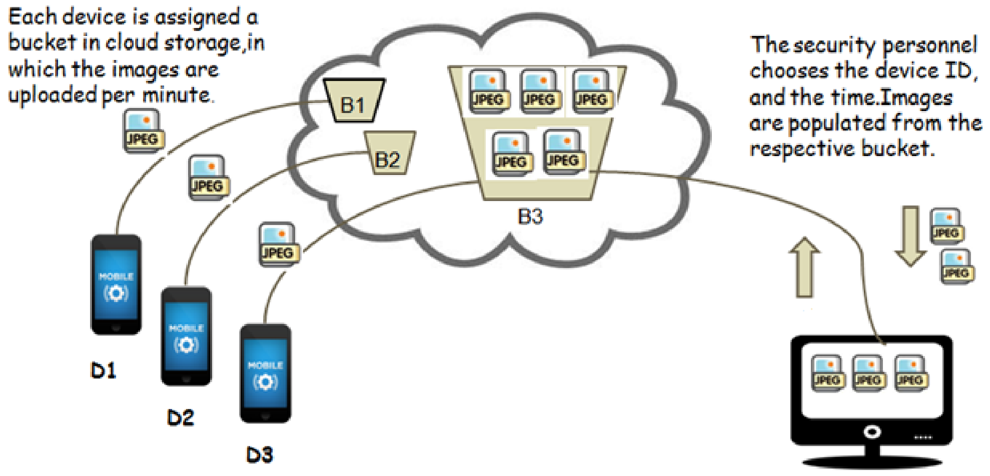
Each device is assigned a bucket in cloud storage,in which the images are uploaded per minute.
B1
B2
B3
JPEG
The security personnel chooses the device ID, and the time.Images are populated from the respective bucket.
MOBILE
D1
D2
D3

Figure 4: Flow Diagram

will be running in the EC2 instance. This also makes use of REST calls to get images stored in cloud.

- Rest Back End Service : RestService is a REST based back end service. This service accepts request from both android application and web page.

5.3 Database Design

- **Device Info** For every device (User), we maintain a table named User in the database which has the following fields:

  ID, NAME, PASSWORD, CLICK DELAY, UPLOAD INTERVAL

  CLICK DELAY, specified in seconds is an adjustable attribute which defines the time interval between two successive clicks by the camera.

  UPLOAD INTERVAL, specified in seconds is also a configurable attribute which defines a session of time so that images captured in that session are to be uploaded altogether at the end of that session.

ID	**NAME**	**PASSWORD**	**CLICK DELAY**	**UPLOAD INTERVAL**
himalaya	Himalaya	cloud	3	120

- **Session Info** For every session we maintain a table named Session in the database which has the following fields:

  ID, USERID, START TIME, END TIME, LOCATION

ID	**USERID**	**START TIME**	**END TIME**	**LOCATION**
7e65ec99-c952-4768-a88c-b551e4ed6837	himalaya	2015-11-22 00:00:00	2015-11-22 00:40:00	IIIT-Hyderabad

5.4 REST Endpoints

5.4.1 GET

- `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/get/users`

```
{
"userList": ["himalaya"],
"message": "Success",
"code": "200"
}
```

- `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/get/locations`

```
{
"locationList": ["IIIT-Hyderabad"],
"message": "Success",
"code": "200"
}
```

5.4.2 POST

- `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/post/loginInfo`
  Payload :

  ```
  {
  "userid":"himalaya",
  "password":"cloud"
  }
  ```

  Response:

  ```
  {
  "name": "Himalaya",
  "userid": "himalaya",
  "clickDelayInSeconds": "3",
  "uploadIntervalInSeconds": "120",
  "password": "cloud",
  "message": "Login Success.",
  "code": "200"
  }
  ```

- `http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/post/signUpInfo`
  Payload :

  ```
  {
  "name":"nivas",
  "id":"admin",
  "clickDelayInSeconds":"30",
  "uploadIntervalInSeconds":"50",
  "password": "6781"
  }
  ```

  Response:

  ```
  {
  "message": "Signup success.",
  "code": "200"
  }
  ```

- http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/post/createSessionInfo
  Payload :

  ```
  {
  "userId":"himalaya",
  "startTime":"2015-11-22 00:00:00"
  }
  ```

  Response:

  ```
  {
  "sessionId": "7e65ec99-c952-4768-a88c-b551e4ed6837",
  "message": "Session created successfully.",
  "code": "200"
  }
  ```

- http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/post/imageInfo
  Payload :

  ```
  {
  "userId":"himalaya",
  "sessionId":"7e65ec99-c952-4768-a88c-b551e4ed6837",
  "snapedAt":"2015-11-22 00:00:03",
  "location":"IIIT-Hyderabad",
  "data":"sadiuaeiaujsd=="
  }
  ```

  Response:

  ```
  {
  "message": "Image Upload success.",
  "code": "200"
  }
  ```

- http://ec2-52-74-222-81.ap-southeast-1.compute.amazonaws.com:8080/RestService/post/searchInfo
  Payload :

  ```
  {
  "dateStart":"2015-11-22 00:00:00",
  "dateEnd":"2015-11-22 00:40:00",
  "locationList":["IIIT-Hyderabad"]
  }
  ```

Response:

```
{
"userList": [
    {
    "count": 1,
    "userid": "himalaya",
    "imgSet": [
    {
        "count": 1,
        "day": "2015-11-22",
        "imgs": [
        {
            "location": "IIIT-Hyderabad",
            "snapedAt": "2015-11-22 0:0:0",
            "imageId": "24ec2bfb-8091-42cb-a4b1-f7e7b9580226",
            "resourcePath":
            "himalaya/7e65ec99-c952-4768-a88c-b551e4ed6837/24ec2bfb-8091-42cb-a4b1-f
        }
        ]
        }
    ]
    }
],
"message": "Successfully fetched records.",
"code": "200"
}
```

6 Design Choices

6.1 S3 Services

- Cross Region Replication : CRR is an Amazon S3 feature that automatically replicates data across AWS regions. With CRR, every object uploaded to an S3 bucket is automatically replicated to a destination bucket in a different AWS region that we can choose. We can use CRR to provide lower-latency data access in different geographic regions.

- Accessing Images quickly : Each image is stored as a Key, Value pair with FileName as – $imageId\_sessionId\_timestamp.jpg$ which provides a user with explicit information about the exact time (timestamp) at which the image was taken. The file structure in S3 follows a hierarchy. So the path to image is $userId/sessionId/imagename$.

- CloudFront CDN : We used Amazon CloudFront a CDN (Content Delivery Network) service to serve the images faster to the user. CloudFront integrates with other Amazon Web Services products to give an easy way to distribute content to end users with low latency, high data transfer speeds. CloudFront is a web service that speeds up distribution of static and dynamic web content, for example, .html, .css, .php, and image files, to end users. CloudFront delivers content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so content is delivered with the best possible performance. If the content is already in edge location with the lowest latency, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

6.2 RDS Services

- RDS provides supports **ACID**, i.e, Atomicity, Consistency, Isolation, and Durability.
- **Querying**: Querying data in NoSQL Databases like DynamoDB is very limited, especially to query for non-indexed data. Also, complex querying can be easily done in RDS.
- **Backup**: RDS has a slick backup when compared to the tedious backup procedure for NoSQL Databases like DynamoDB
- **Speed**: RDS has better response time when compared with NoSQL Databases like DynamoDB.
- **Latency**: On table creation, RDS allows us to use the table with negligible latency. Also, latency for read/write is better in the case of RDS.

7 Problems Faced

- We are unable to connect to the Amazon EC2 node as port 22 is blocked inside our college LAN network. We solved this problem using ssh tunneling.
- Sending images from mobile device using asynchronous background Task instead of making the post call for the image data on the UI Thread.

- We had to define appropriate security groups for different AWS services used such that our application can GET and POST data to these services.

8 Tentative Timeline

- Phase 1 (6 Nov - 13 Nov) : Set up REST APIs and Android Application and enable upload of images to cloud.
- Phase 2 (14 Nov - 28 Nov) : Enable querying of images through a web app.

References

[1] Amazon S3,
`http://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html`

[2] Amazon RDS,
`http://docs.aws.amazon.com/AmazonRDS/latest/APIReference/Welcome.html`

[3] Amazon EC2,
`https://aws.amazon.com/ec2/`

[4] Creately,
`http://creately.com/`