**Mobile Security Solution**

Team No – 9

(Mentor – Vishrut Mehta)

Deepak Jiwal

Dilwar Hossain

Jitendra Yasaswi

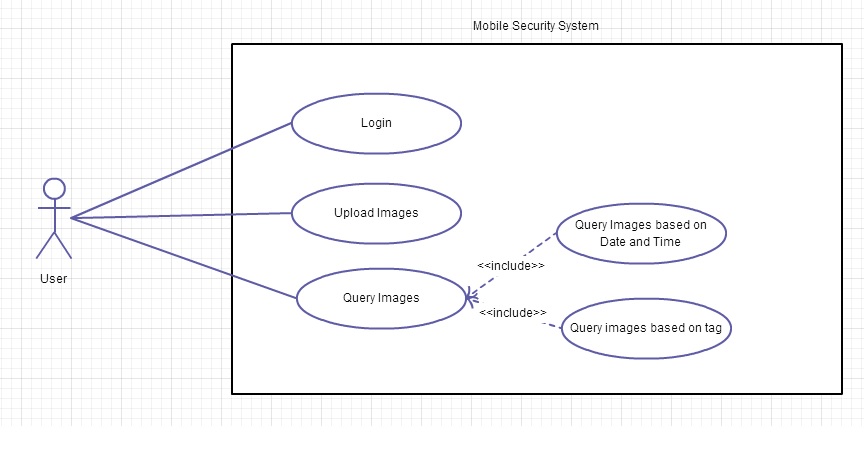
Ganesh

**Problem Statement:**

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.

**Use Case Diagram:**



**Design Database:**

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

ID, NAME, PASSWORD, CLICK DELAY IN SECONDS, UPLOAD INTERVAL IN SECONDS

CLICK DELAY IN SECONDS is an adjustable attribute which defines the time interval between two successive clicks by the camera.

UPLOAD INTERVAL is also an adjustable 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.

**USER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **NAME** | **PASSWORD** | **CLICK DELAY IN SECONDS** | **UPLOAD INTERVAL IN SECONDS** |
| himalaya | Himalaya | 1234 | 60 | 1800 |

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

ID, USERID, START TIME, END TIME and LOCATION.

**SESSION**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **USER ID** | **START TIME** | **END TIME** | **LOCATION** |
| d07d7163-ebf6-404c-b3e7-643e3c003813 | Himalaya | 2014-09-23 08:00:00 | 2014-09-23 08:29:59 | East-IIITH |

The attributes START TIME and END TIME defines the start and end time of the session respectively.

**Store Meta Info:**

For storing the Meta information (user and session tables) we used Amazon **Relational Database Service** (Amazon **RDS**) which is a web service that makes it easy to set up, operate, and scale a relational database in the cloud.

* Engine – **MySQL** 5.6.19a
* Storage – 5GB
* Availability zone – US West (Oregon)
* Configuration - 1 VCPU, 1GB.

**Store Image:**

For storing the images in cloud we used Amazon **S3** (**Simple Storage Service)**which is an online file storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces.

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**

* Key – userId/sessionId/fileName.jpg
* Value – Base64 encoded image.
* Bucket name – mobilesecurity
* Publicly accessible – Yes
* FileName – imageId\_sessionId\_timestamp.jpg

**EC2 - Spin up virtual server:**

Tomcat server is deployed in the EC2 instance, which host the REST Back end service and web page.

* Public DNS - ec2-54-201-117-236.us-west-2.compute.amazonaws.com
* Availability zone – US West (Oregon)
* EBS size – 8 GB
* Amazon Machine Image – Amazon Linux AMI (HVM) 3.14 Kernel

**Serve images faster:**

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.

Steps:

* Storing the original versions of our images in one Amazon Simple Storage Service (Amazon S3) bucket.
* Then we distribute download content (here images).
* Make the images accessible to everyone.
* Use the CloudFront domain name in URLs for images (for example, http://d111111abcdef8.cloudfront.net/image.jpg) instead of our own domain name (for example,http://www.example.com/image.jpg).

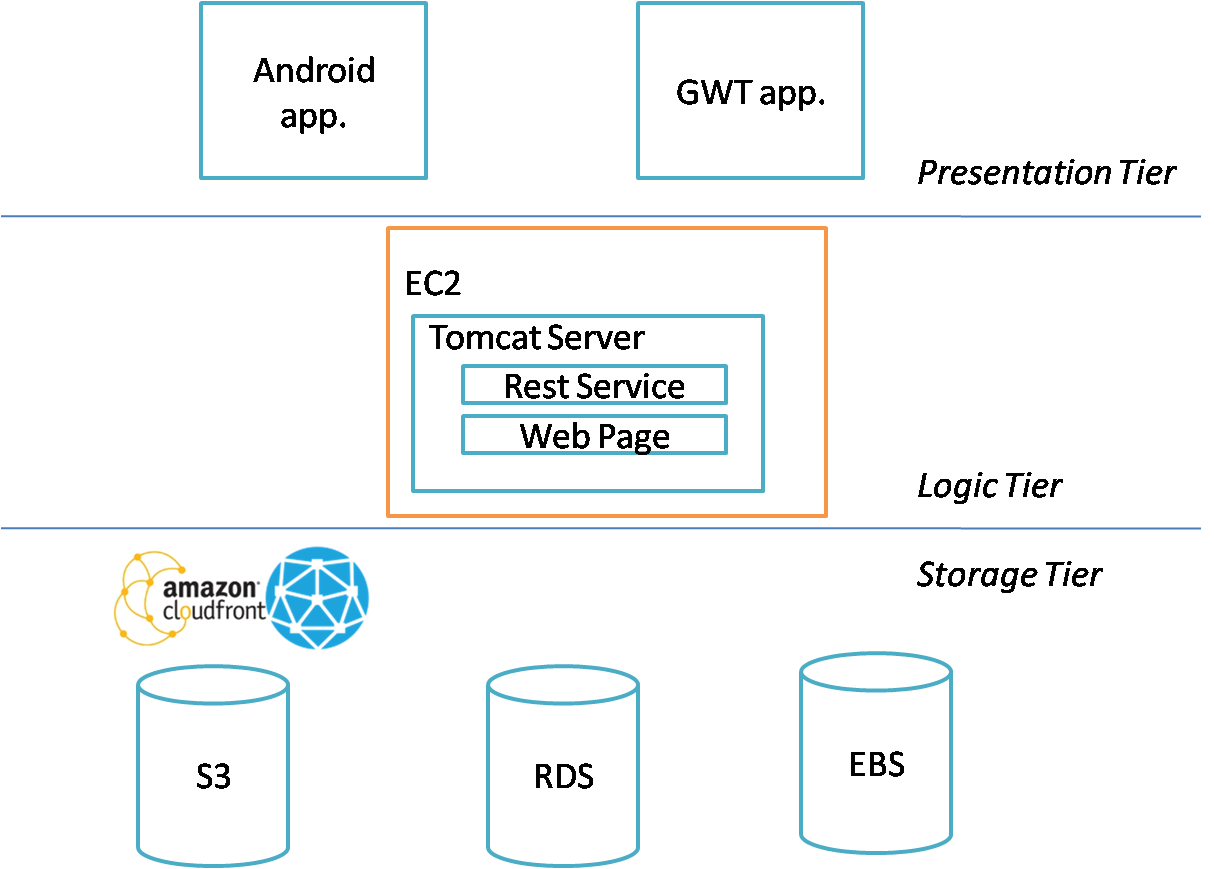
Below shown are the URLs to images before and after using the CloudFront. (Notice the changes in the URL)

* Before - https://**s3-us-west-2.amazonaws.com**/mobilesecurity/admin/...
* After - https://**d3iq5ru0cfmp88.cloudfront.net**/mobilesecurity/admin/...

**REST Calls:**

|  |  |  |
| --- | --- | --- |
| Endpoint | Payload | Sample response |
| POST  Content-Type: application/json  [http:// ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/post/loginInfo](http://localhost:8080/RestService/post/loginInfo) | {  "userid":"library",  "password":"678"  } | {      "name": "library",      "userid": "library",      "clickDelayInSeconds": "20",      "uploadIntervalInSeconds": "20",      "password": "678",      "message": "Login Success.",      "code": "200"  } |
| POST  Content-Type: application/json  [http:// ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/post/signUpInfo](http://localhost:8080/RestService/post/signUpInfo) | {"name":"nivas","id":"admin","clickDelayInSeconds":"30","uploadIntervalInSeconds":"50","password": "6781"} | {      "message": "Signup success.",      "code": "200"  } |
| <http://ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/post/createSessionInfo>  Required before initiating a upload | {"userId":"admin","startTime":"2014-12-12 00:00:00"} | {      "sessionId": "7e65ec99-c952-4768-a88c-b551e4ed6837",      "message": "Session created successfully.",      "code": "200"  } |
| POST  Content-Type: application/json  [http:// ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/post/imageInfo](http://localhost:8080/RestService/post/imageInfo)  Uploads only one image, with data in Base64 format.  Must have created a session before. | {"userId":"admin","sessionId":"7e65ec99-c952-4768-a88c-b551e4ed6837","snapedAt":"2014-12-12 00:00:00","location":"centre-iiith","data":"dddddd"} | {      "message": "Image Upload success.",      "code": "200"  } |
| POST  Content-Type: application/json  <http://ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/post/searchInfo> | {    "dateStart":"2014-11-20 00:00:00",  "dateEnd":"2014-12-21 00:55:00",    "locationList":["centre-iiith"]  } | {      "userList": [          {              "count": 1,              "userid": "admin",              "imgSet": [                  {                      "count": 1,                      "day": "2014-11-12",                      "imgs": [                          {                              "location": "centre-iiith",                              "snapedAt": "2014-11-12 0:0:0",                              "imageId": "24ec2bfb-8091-42cb-a4b1-f7e7b9580226",                              "resourcePath": "admin/7e65ec99-c952-4768-a88c-b551e4ed6837/24ec2bfb-8091-42cb-a4b1-f7e7b9580226.jpg"                          }                      ]                  }              ]          }      ],      "message": "Successfully fetched records.",      "code": "200"  } |
| GET  http:// ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/get/users  Used in web page. | - | {      "userList": [          "admin",          "entrance",          "kadamba nivas",          "library",          "obh north mess",          "sh1"      ],      "message": "Success",      "code": "200"  } |
| GET  [http:// ec2-54-201-111-236.us-west-2.compute.amazonaws.com:8080/RestService/get/locations](http://localhost:8080/RestService/get/locations)  Used in web page. | - | {      "locationList": [          "center-iiith",          "north-iiith",          "west-iiith",          "south-iiith",          "centre-iiith"      ],      "message": "Success",      "code": "200"  } |

**Technology stack:**



Other technologies Used:

* Android SDK – To build android application
* Jersey – To build REST backend.
* Google Web Toolkit – To build the web application. GWT is a compiler to convert Java code to Javascript, improving the development time.
* Maven – Build tool.
* Eclipse
* Putty – Remote login to the EC2 instance for executing commands.
* WinSCP – Remote login to the EC2 instance for transferring files.

**Problems faced:**

* **Connecting to EC2, RDS instances.** (Firewall issues) – We were unable to connect to the EC2 and RDS nodes as those ports were in blocked state. We solved by using a open network (airtel data card).
* **Defining security groups for services over ports**. – We defined inbound and outbound rules for each resource in the cloud.
* **Remote connection to EC2.** - We solved by using tools such as Putty (to remotely execute commands on a EC2 instance) and WinSCP (to transfer the service and web app. wars)
* **Database storage optimization.** – We improved the database storage by storing every session information, and using it to fetch all image specific details from S3. Thus we successfully avoided having one entry for each image in the database.
* **Parsing JSON Payload** – We solved by using Jersey which seamlessly support exposing our data in a variety of media types, hiding the low-level details of the client-server communication.