eRhODIS™

Architectural Specification
Overall System and Architecture
Description

RhODIS® Android Development : eRhODIS™ Application

Version 1.3

This document contains the architectural specification of the eRhODIS™ application and a detailed description of the architecture and system.

Change History

Date	Version	Description	Updated By
05 August	1.0	Created Document	Gideon
05 August	1.0	Added main sections as discussed.	Gideon
28 October	1.1	Added Amazon EC2 services.	Gideon
18 November	1.2	Added Combined Architectural diagram.	Gideon
7 February	1.3	Added a diagram to display the life-cycle of an android application and how we use it in our sync process.	Gideon

**Changes will be displayed in red throughout the document.*

Table of Contents

Introduction

RhODIS® (Rhino DNA Index System) is a project that was initiated by the Veterinary Genetics Laboratory of the University of Pretoria in order to help with the plight of the rhinos. The Veterinary Genetics Laboratory is collecting DNA samples of rhinos across the country to create a database using the unique DNA profile of individual rhinos.
The goal is for all rhinos to be on the system. This will deter poachers and assist in forensic prosecutions.

RhODIS® was first used in a rhino poaching case in 2010 and resulted in a Vietnamese citizen being sentenced to 10 years imprisonment for having rhinoceros horns from poached rhinos in his baggage when he was apprehended at OR Thambo International Airport. South African National Parks (SANParks) have partnered with RhODIS since 2010 and in association with the Forensics Science Laboratory of the South African Police Services have played a key role in the development and implementation of the RhODIS Kit for sample collection.

The South African Department of Environmental Affairs introduced amendments to the norms and standards for sample collection and identification of live and poached rhinos under the National Environmental Management: Biodiversity Act 10 of 2004 which requires that samples are collected from all poached rhinos and other rhinos that are immobilized or die using RhODIS® kits which then have to be submitted to the Veterinary Genetics Laboratory for inclusion on the RhODIS® database. A number of other bodies including the South African National Parks Honorary Rangers, the World Wildlife Fund, corporates and individuals have donated funds to support the development and implementation of RhODIS®. eRhODIS™ has been developed as an adjunct for RhODIS® to aid in the collection of samples and information relevant to the RhODIS® project and Samsung is the exclusive technology partner associated with this development.

Purpose

The purpose of this document is to provide a comprehensive overview of the eRhODIS™ application that the Veterinary Genetics Laboratory of the University of Pretoria together with RhODIS® will be developing.

This document will give a detailed and comprehensive description of the architecture of the system and will include the overall architecture and the architectural features, its subsystem views, policies and its data requirements as well as constraints under which it should operate.

The intended audience of this document would be any person interested in the development life-cycle of the eRhODIS™ android application. This document will serve as a guideline for intended android application functionality.

Document Conventions

Conventions used in this document:

Use-Case notation using the Unified Modelling Language (UML).
Crow's foot notation is used to document entity relationship diagrams.

Project Scope

The main idea of the application is that it ensures that all information needed to ensure that DNA samples are collected from poached and live rhinos are collected in a standard way and that all the necessary information to provide details of the chain of custody for these samples are automatically collected and uploaded to a secure database for future use should the need arise. All information, including GPS coordinates, photos, sample information is uploaded and stored on the cloud server.

It also ensures that key required data is always collected and uploaded. Use of the application also enhances data accuracy and does away with the need to manually enter any of the data after receipt of the samples in the Laboratory.

The application also uses the S-Pen to capture the authorised person’s signature which therefore provides further integrity for the chain of custody features incorporated into the application.

References

Amazon Web Services http://aws.amazon.com/what-is-cloud-computing/?navclick=true
Use-case notion in Unified modelling language (UML).
Model View Controller, http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.
N-tier, http://en.wikipedia.org/wiki/Multitier_architecture.
Client-server, http://en.wikipedia.org/wiki/Client%E2%80%93server_model.
Android, http://www.android.com/.
Android architecture, http://developer.android.com/index.html.

Related Documents

eRhODIS™ Requirements Specification.
Other documents still needs to be referenced.

System Description

The eRhODIS™ application will serve as a utility tool for end-users to collect samples and relevant data in the field in the event of a rhino poaching or related incident, where after the data will be stored and uploaded to a secure cloud server.

The application ensures that all information needed to ensure that DNA samples are collected from poached and live rhinos are collected in a standard way and that all the necessary information to provide details of the chain of custody for these samples are automatically collected and uploaded to a secure database for future use should the need arise. All information, including GPS coordinates, photos, sample information is uploaded and stored on the cloud server.

It also ensures that key required data is always collected and uploaded. Use of the application also enhances data accuracy and does away with the need to manually enter any of the data after receipt of the samples in the Laboratory.

The application also uses the S-Pen to capture the authorised person's signature which therefore provides further integrity for the chain of custody features incorporated into the application.

User account information are automatically set to a demo account upon the first installation of the application on the device. The specific user's account details are provided and preconfigured by RhODIS® officials and administrators before the device is used by the end user. These account details will also be used by the user to log into the erhodis backed to view their submissions.

The application provides online and off-line capabilities for areas where users have no internet connectivity i.e. cell phone reception or wi-fi coverage.

Overall Architecture

Users will only have access to all of the features discussed in the requirement specification via the android application and the web interface. The application will communicate with a cloud-server-system that provides a DBMS and a platform that allows information to be display to the users.

We have so far been using Amazon Web Services as a platform for the above mentioned requirements. AWS offers a platform with all the necessary tools for us to specify the type of connection or service needed and to provide us with an infrastructure to manage these services.

Architectural Patterns

When looking at the requirements specification, we note that many of the requirements are met on the android device itself. The built-in GPS takes care of acquiring the correct latitude and longitude coordinates. The built in camera takes care of the camera functionality and interface used to capture images throughout the application. The S-Pen handles signature capturing. The application further parses all the data into a SQLITE database that sits on the device self.

This leaves a few non-functional and functional requirements for the server or cloud infrastructure to cater for, namely, user registration, email and pdf generation and further data visualisation methods and displaying the actual data.

To meet all the requirements specified, a three-tier client server and MVC (Model View Control) design is required.

Motivation

- MVC caters for extensible views and provides easily reusable code.
- A Three-Tier design reduces load and abstracts database interaction from the client.
- A Client Server architecture is needed for security.

Three-Tier Pattern

How it works

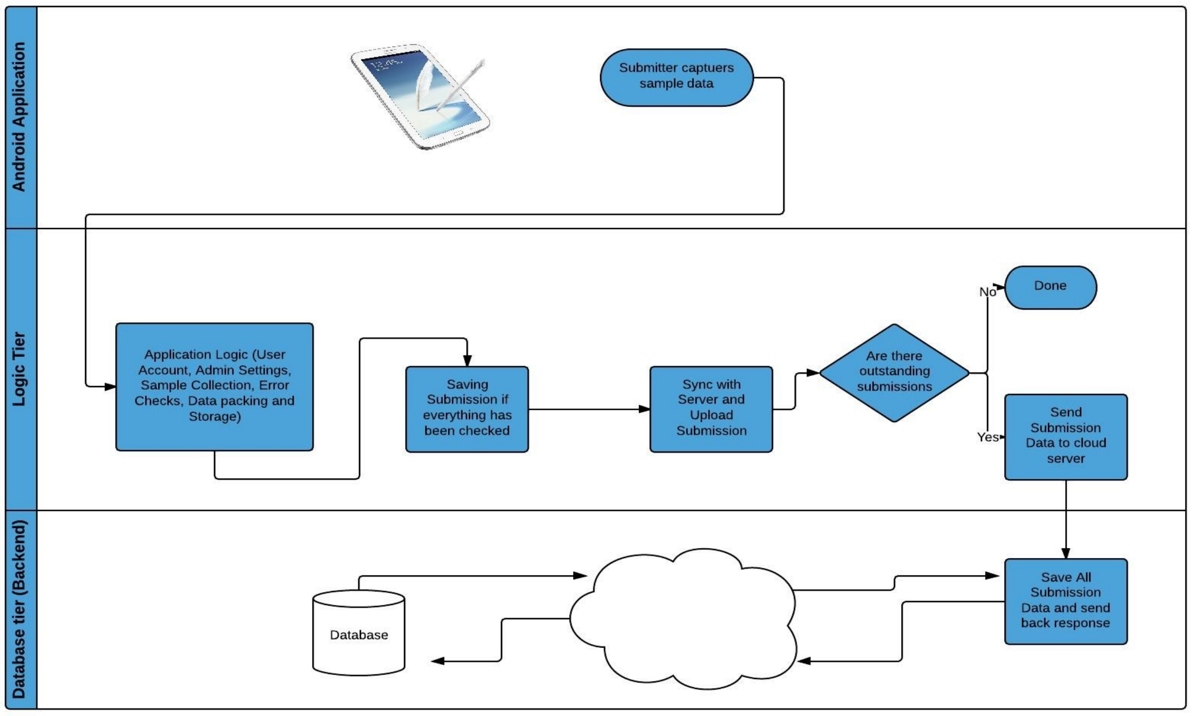

Android Application
Logic Tier
Database tier (Backend)
Submitter captuers sample data
Application Logic (User Account, Admin Settings, Sample Collection, Error Checks, Data packing and Storage)
Saving Submission if everything has been checked
Sync with Server and Upload Submission
Are there outstanding submissions
No
Done
Yes
Send Submission Data to cloud server
Save All Submission Data and send back response
Database

Figure 1 – Explaining the Three-Tier architectural pattern.

Android Application

The android application is essentially our presentation tier. The application background uses features available on the device and in the android framework to performs tasks and to speak to our logic tier in the cloud.

Logic Tier

The Logic Tier handles all account validation, sample collections, data and photo storage, error checking and syncing functionality. All data/submissions that needs to reach the Database Tier will be validated here and packaged/serialised before being sent to the Database Tier. Any requests for data or services from the server must first go through the logic tier as well.

Database tier

The DBMS of the database tier deals strictly with database management, issues of concurrency and data control are abstracted from the logical tier to keep a modular design. Any requests to the database tier must come via the logic tier.

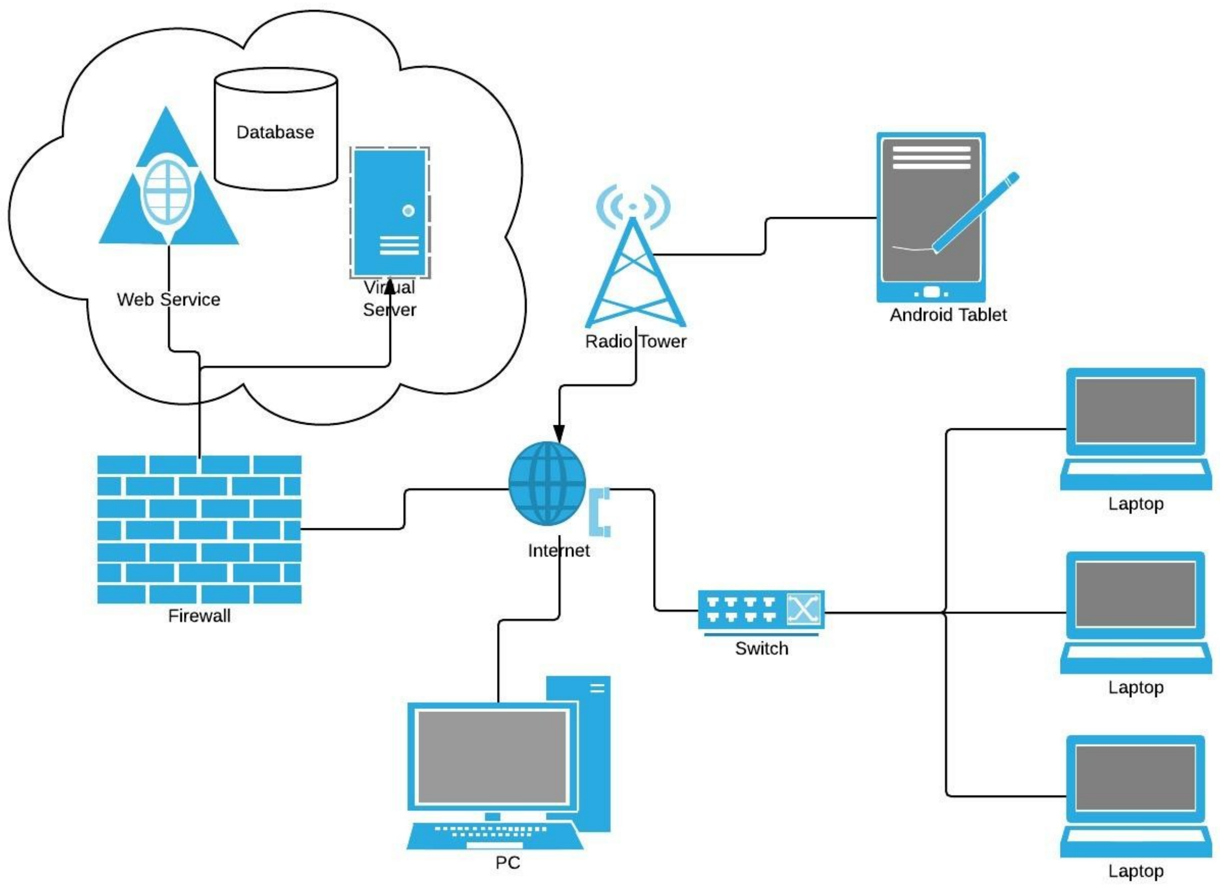
Motivation

We picked this pattern as it allows us to move the database and logic tier as soon as the traffic load gets too high for our current cloud package or to make the process easier if we decide to ever migrate to a different host.

Furthermore, each section of the project can be abstracted from one another meaning that we can make changes to one sub-system without it affecting the other sub-systems.

Client Server Pattern

How it works



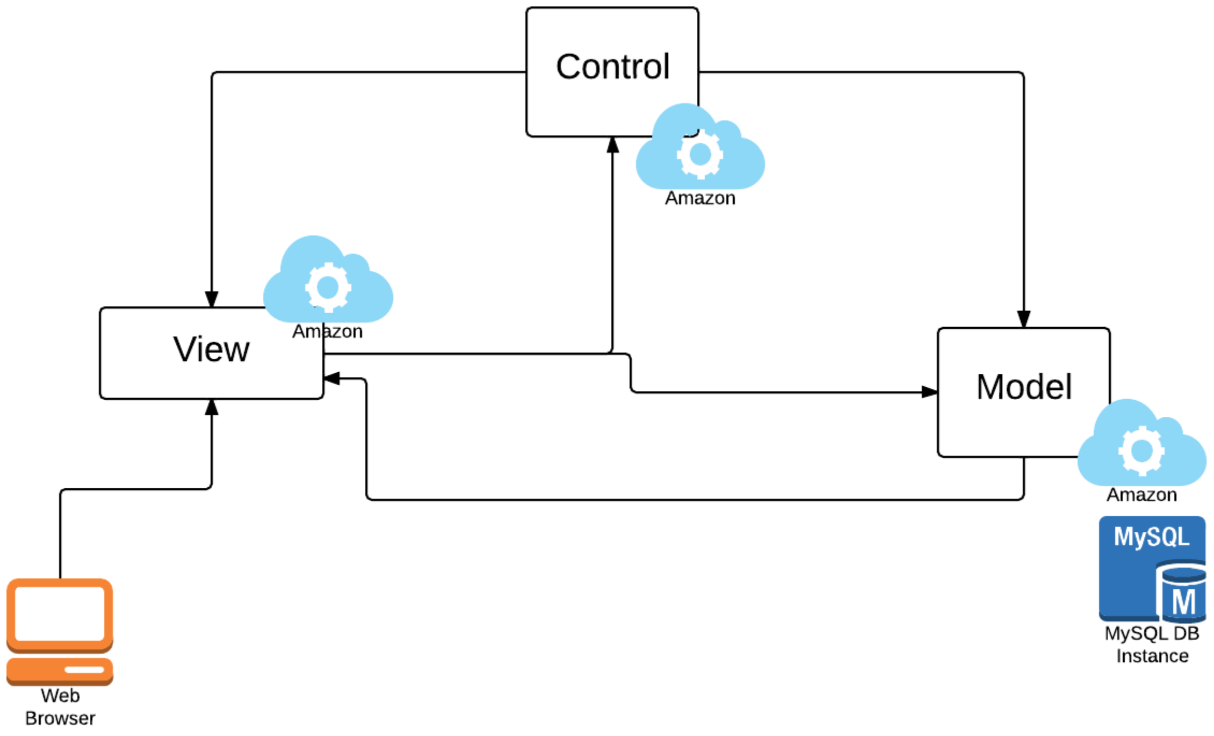
Motivation

The client-server model has been used because it is unsafe and difficult to have peer-to-peer interaction between android devices unless using the NFC protocol, which is not practical for our purposes. We are saving sensitive information in terms of the animal's details, owner information, sample information and so on, and is therefore essential that we make every effort to ensure that the information the user intends to save is only visible to people who are authorised to view this type of information.

The client server pattern allows us to introduce an additional level of security. The only means of one account to view the information submitted, is to login to the secure web server. Furthermore, we can limit the users view to only see his own submissions on the server.

Model View Controller

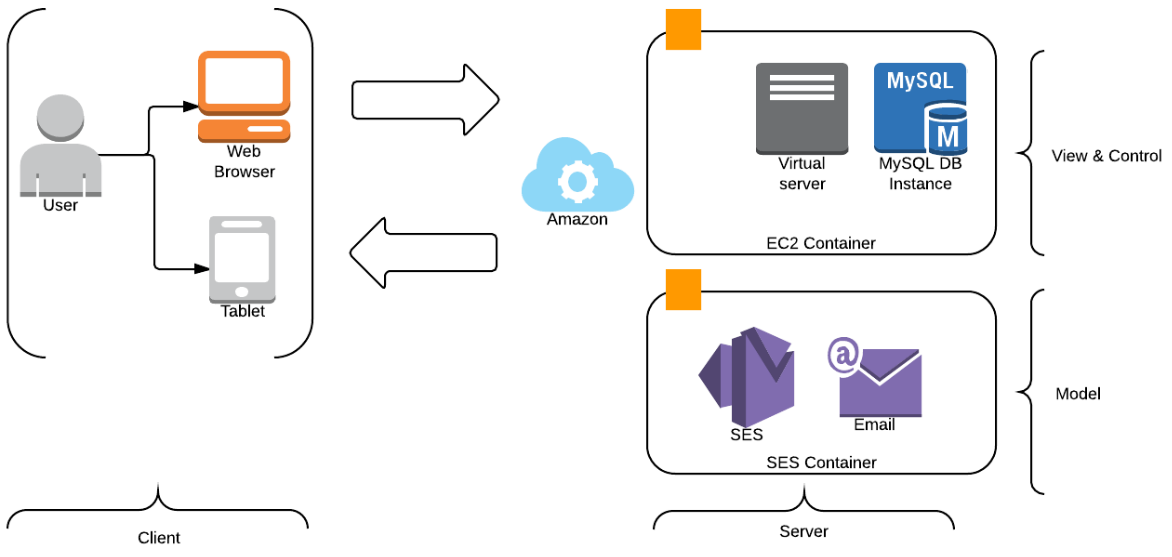
How it works



Motivation

This architecture and functionality thereof is automatically incorporated into all android applications because of the way that the android architecture is designed.

Combined Architectural Patterns

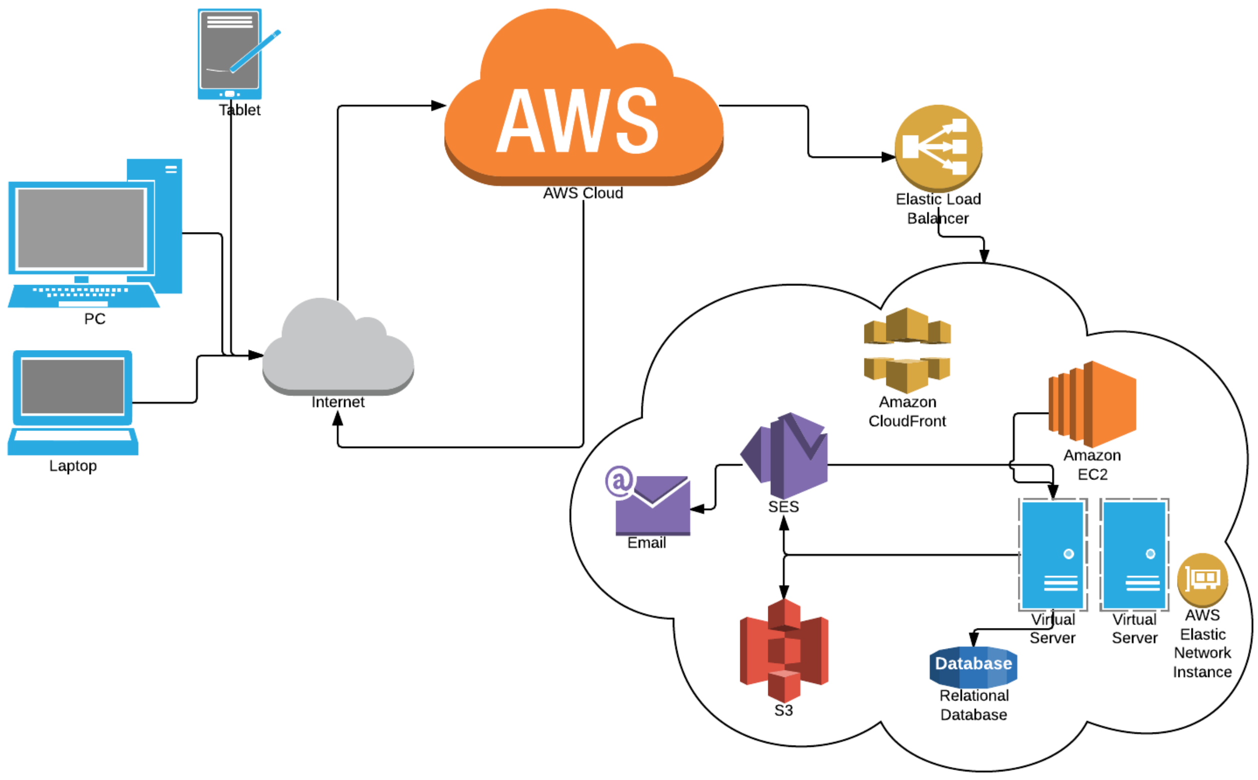


Details of Subsystem

Overview

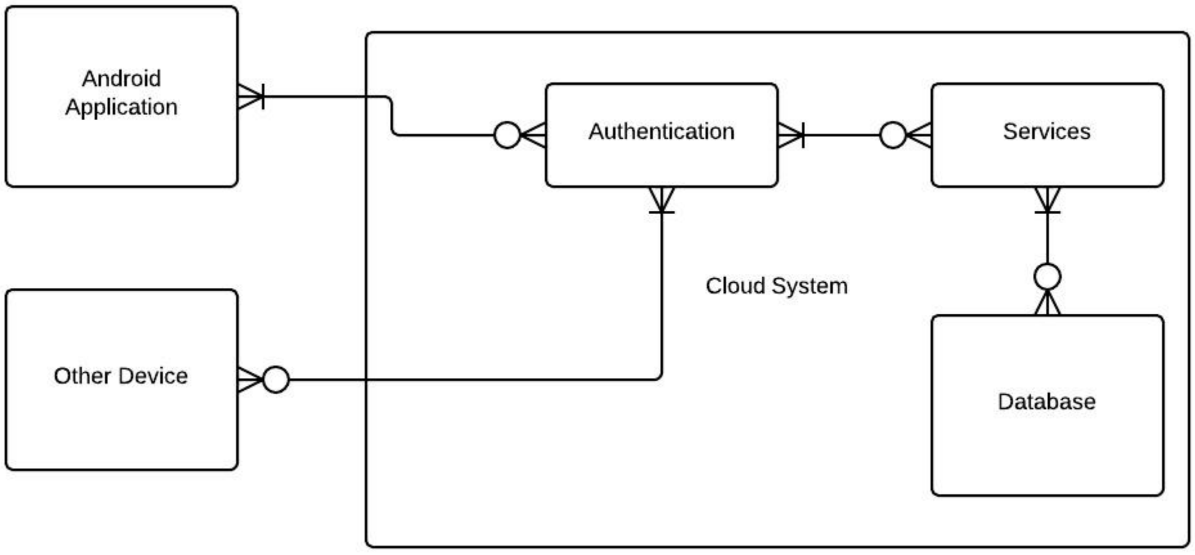
The logical tier and DBMS will be hosted by a cloud service provider. *We have since been using Amazon Web Services as our cloud computing infrastructure. Managing our own servers locally can later become a hassle as we need more infrastructure and need to maintain our servers personally. Thus, we wish to rather offload such a responsibility to the cloud service provider (Amazon) so that we can focus more on the quality of our application and the services it needs to perform.

Below is a basic idea of how such an infrastructure looks like:



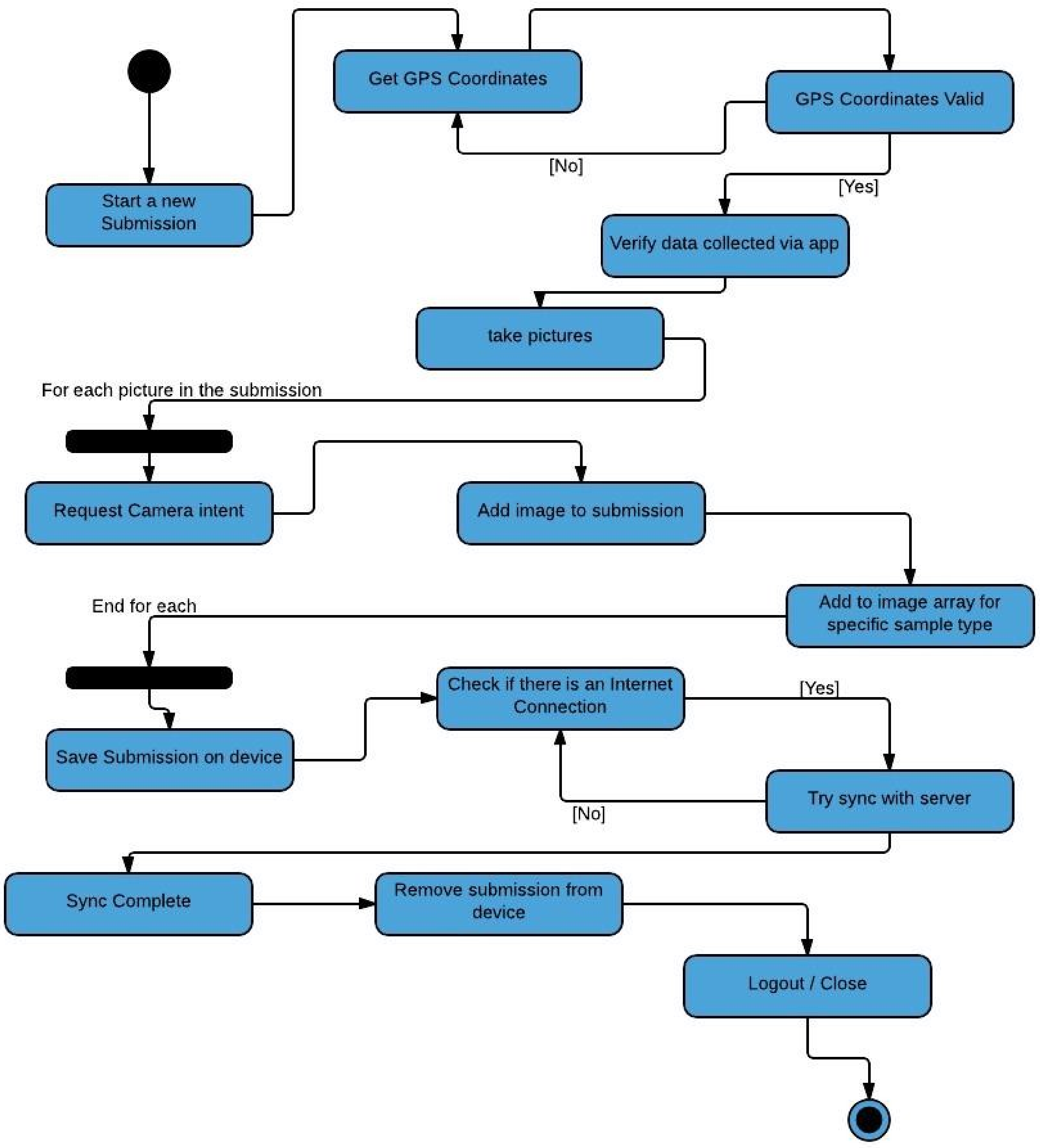
Development View

An illustration of the components which make up the system, demonstrated by a component diagram



Process View

A brief illustration of the submissions process from the application.

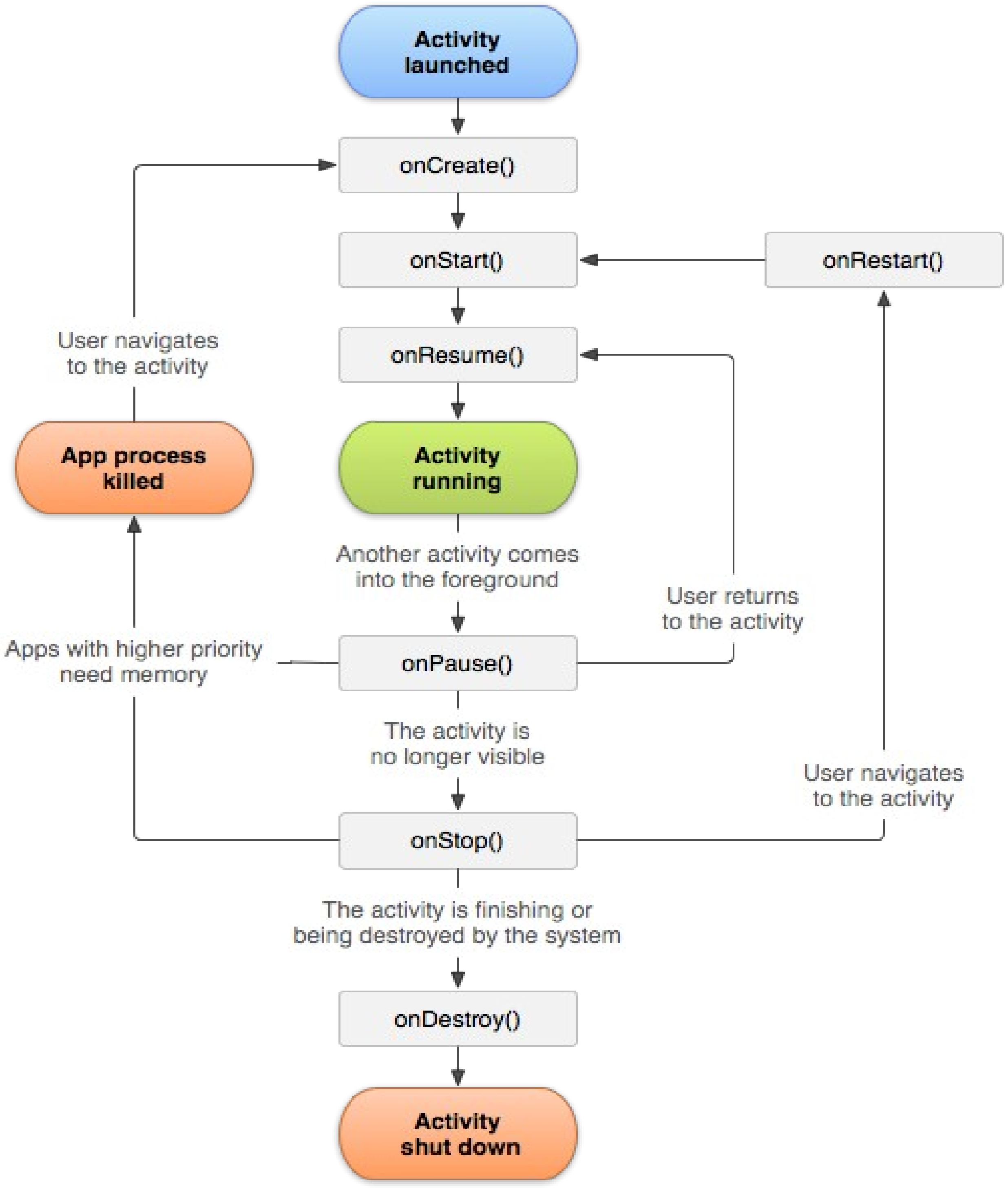


Start a new Submission
Get GPS Coordinates
GPS Coordinates Valid
[No]
[Yes]
Verify data collected via app
take pictures
For each picture in the submission
Request Camera intent
Add image to submission
Add to image array for specific sample type
End for each
Save Submission on device
Check if there is an Internet Connection
[Yes]
Try sync with server
[No]
Sync Complete
Remove submission from device
Logout / Close

Android Life cycle View

A brief illustration of the android life-cycle, relevant mostly because of our background syncing requirements and implementation. As stated in the Requirements Specification, our application makes use of a background syncing service to ensure that submissions made by users are synced at all times (when a network becomes available) and that it runs seamlessly in the background without interference or interaction with the user.

This diagram shows how we are able to implement this functionality in the background.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Policies

Coding Standards

Android Interface
Typography: Improves consistency, neatness and readability.

Naming conventions
Constants indicated with capital letters, all other variables and method names to begin with a lower-case letter and use camel case.
Class names are nouns and begin with upper case letters.
Function names are verbs.

Layout
Tabs are to be used to indent code.
Use white space to separate functions.
Spaces between operands and variables.
Use braces to define the scope of functions, methods , loops and conditional statements.

Clarity
Use comments to provide a brief description of the intention of a segment of code.
English is to be used when writing comments.

Re-use
Segments of code that are frequently used are to be turned into methods.

Effectiveness
The system must be memory efficient as we are developing on a mobile device with limited memory.

Debugging and Testing
Usage of try-catch statement must be used where methods or interfaces are likely to cause problems.

Code must be checked for bugs that need to be resolved.
If an error occurs during runtime the user must be provided with a detailed error message.

Technologies for Implementation

Android Interface must follow object orientated policies.

Restrictions

The application is restricted to android devices only and currently restricted to be specifically used on a Samsung Note 8 device only. The application has been tested mostly on Android version 4.2.1 (Jelly bean) which makes 4.2.1 the preferred android version.

Availability

Once an administrator has configured the users account in the admin settings, users are able to start creating sample submissions, with or without an internet connection.

Platform Support

The application is only available on Android. At the moment, we are only support Android 4.0 (Ice cream sandwich) and higher. The current version of development is 4.1 / 4.2 Jellybean.

Glossary

Term	Meaning
Android	A mobile operating system based on a modified version of the Linux kernel.
AWS	Amazon Web Services. (A cloud computing service provider and infrastructure.)
Cloud Server	A Server that is maintained within a cloud computing infrastructure.
Cloud Computing	Computing concepts that involve a large number of computers connected through a real-time communication networks such as the Internet.
Database	A database is an organized collection of data.
DBMS	Database Management System
eRhODIS™	Electronic RhODIS®
GPS	Global Positioning System
Ice cream sandwich	An android operating system release. 4.0
Jellybean	An android operating system release. 4.1
MVS	Model View Control
RhODIS®	Rhino DNA Indexing System
PDF	Portable Document Format
Server	A computer system that runs one or multiple servers to provide services over a network and deliver content on demand.
Web Server	A Server providing web services such as retrieving web pages and/or data from a database.
UML	Unified Modelling Language