LocAdoc
System Architecture Design Document (SADD)

Version 1.0

Prepared by: Abhi Jay Krishnan
Kim Hyeocheol
Rivaldo Erawan
Durrah Afshan

Table of Contents

1. Introduction

This document describes the architecture of LocAdoc system.
It describes:

- A general description of the system
- The logical architecture of software, the layers and top-level components
- The physical architecture of the hardware on which runs the software
- The justification of technical choices made
- The traceability between the architecture and the system requirements.

2. Software Architecture

2.1 Architecture overview

The system works in mobile environment. The user may carry his mobile device along with him where ever he may wish to go. Hence the system is developed taking this into consideration. The user may be anyone who wish to use secure his or her files.
The architecture takes these functionalities into consideration.

- User authentication
- Signup
- Password recovery
- Importing new files
- User account management
- File backup
- Local file storage
- Deleting files
- File recovery and data (all files) recovery
- Location based file locking
- Grouping of files in same area.
- User account management

The application will use a SQLite database internally and will require continues internet access and GPS connectivity.
The system makes use of the external interface that is the AWS services such as Cognito, DynamoDB and S3.

2.2 Architecture Design Patter

The application will be implemented using *Model View Presenter* (MVP) architectural pattern.

2.2.1 Why MVP pattern?

MVP pattern developed from MVC (Model View Controller) architectural pattern which has been gaining importance over time.
In MVC we have a problem arising from the fact that Android activities are closely coupled to both interface and data access mechanisms. This leads to following issues : -

- **Testability** - The controller is tied so tightly to the Android APIs that it is difficult to unit test.
- **Modularity & Flexibility** - The controllers are tightly coupled to the views. It might as well be an extension of the view. If we change the view, we have to go back and change the controller.
- **Maintenance** - Over time, particularly in applications with anemic models, more and more code starts getting transferred into the controllers, making them bloated and brittle.

MVP pattern solve this by combining the activity with the View in MVC (XML UI definition) and creating a new layer called the precentor to update the view.

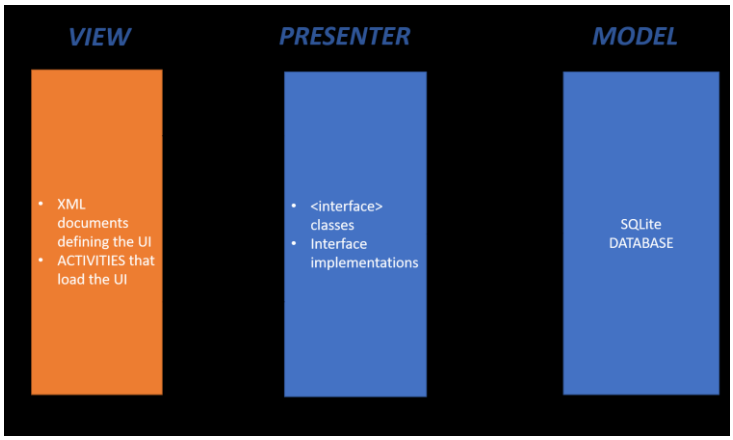

Figure 1 : MVP design

2.3 Logical architecture overview

The application logical architecture is developed based on the MVP pattern described above.

2.3.1 Layer diagram

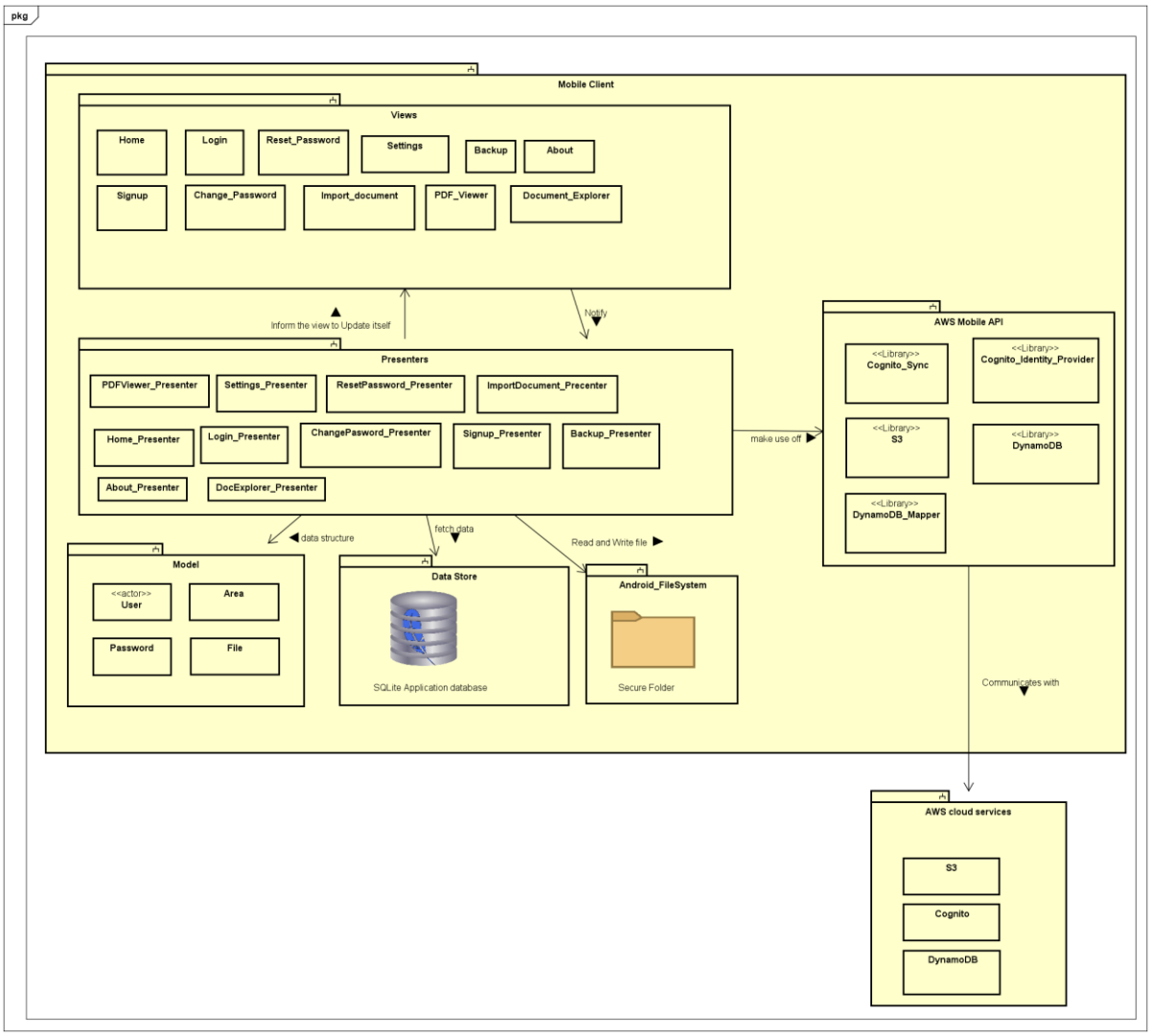

Figure 2: System Architecture

2.3.1.1 View

This layer contains activity classes and xml files that define the structure of the android application user interface. Each view will also have an interface to link with the Presenter. The interface will define the different sections that can be modified in the UI.
There are 11 separate activates the layout of these document will be defined in the user interface design document.

2.3.1.2 Presenter

This layer defines the classes which hold the core business logic. The Presenters form a link between the UI and the model. The interface notifies the Presenter based on the user activity and the Presenter will do necessary logical computation and instruct the user interface to update itself. Each activity will have a corresponding Presenter, but the Presenter may resort to other helper classes to do some additional computations.
When there is a need to retrieve data from the database the Presenter will make connection with the SQLite database and view and same when it comes to updating database. Presenter also read and write to files in android files system. Presenter will also make use of AWS Mobile SDK to access AWS services.

2.3.1.3 Model

This layer forms a data structure layer which does all the necessary backend operations related to the business logic.

2.3.1.4 AWS Mobile API

This is the library created by AWS and forms a interface between the application and the AWS cloud services. The main sub libraries required for this application will be Cognito related libraries (User authentication) [1], S3 related libraries (Data storage) [2] and DinamoDB (Database) [3]. AWS IAM (Identity and Access Management) provide a secure access protocol to interact with amazon services [4].

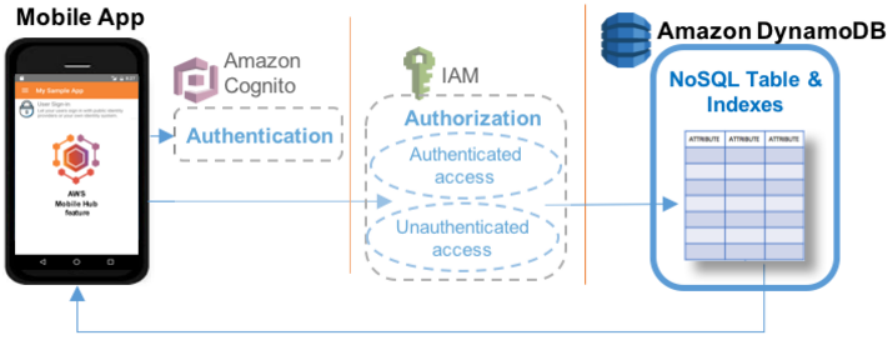

Figure 3: AWS DynamoDB Access

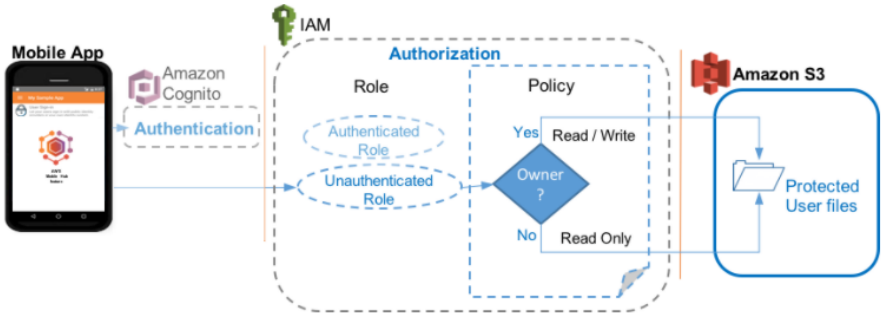

Figure 4: AWS S3 Access

2.3.2 Logical design qualities

The system intends to achieve following qualities through the above implementation
Design principles of the application.

2.3.2.1 Modularity

Separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.

2.3.2.2 High Cohesion

Each module has functions and elements that are strongly related, only to fulfil one purpose or task.

2.3.2.3 Low Coupling

Modules are loosely coupled and independent so that a change in one module do not affect the other modules.

2.3.2.4 Standardization

Implementation will conform to standard that has been established and agreed by different parties, this is crucial for things like security.

2.4 Physical Architecture

2.4.1 Physical Architecture Overview

The architecture adopted by the application will be a new and sophisticated *cloud-client architecture*. This architecture works by making use of cloud computing resources to manage user's data and credentials as well as authenticating user. The proposed application will be using AWS (Amazon Web Services) as its cloud service provider and AWS Mobile SDK [4] for the development of the app. The main advantage this architecture has is that we don't need to manage the resources in the cloud including the security of it.

2.4.2 Deployment Diagram

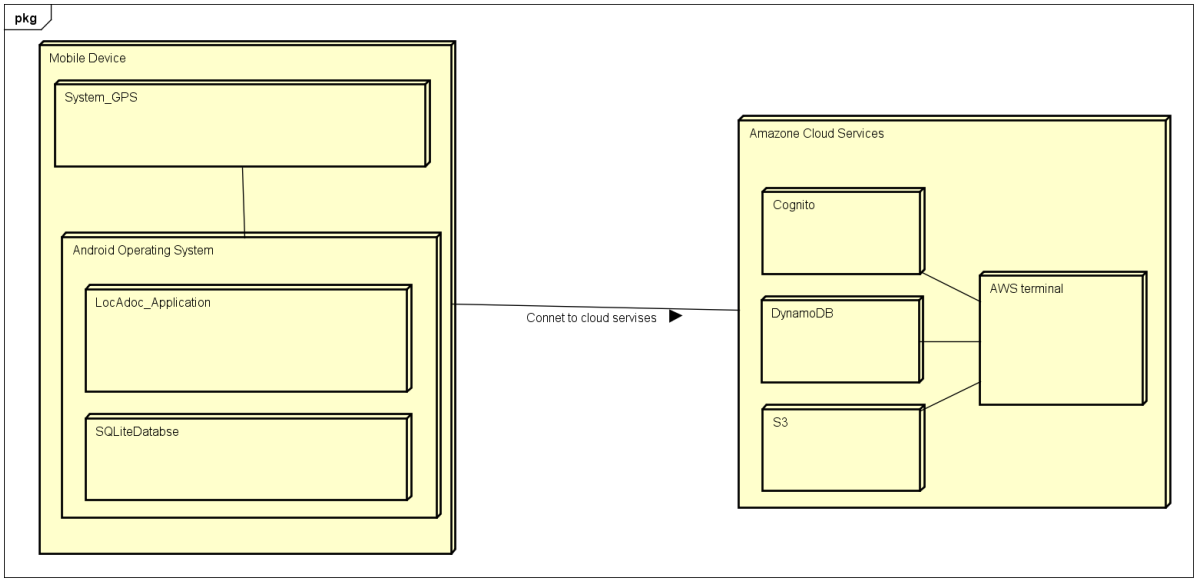

Figure 1: System deployment diagram

The above diagram shows the physical architecture of the system. The execution environment of the system is a mobile devise running android operating system. The application requires access to the phones GPS device to get locational details of the user. The application will also have local SQLite database.
Amazon Cloud Service will be the external interface providing centralized user authentication, database and cloud storage services.

3. System Architecture Qualities

The proposed application will have the following quality:

3.1.1 Extensibility

The proposed application can add additional functionality without changing or damaging much of the current system. New data types can be added if it is supported by the android.

3.1.2 Maintainability

Following the design principles of high cohesion and low coupling, small modifications will not be a problem. Changing one module will not affect other modules significantly.

3.1.3 Performance

The response time will be in acceptable manner even with the huge amount of data that are processed. Efficient encryption algorithm is used as well as other processing algorithm.

3.1.4 Usability

Adapting KISS (Keep It Simple Stupid) principle in designing interfaces will give user easier times in learning and figuring out the proposed application. It lets user to take less time to perform a certain task.

3.1.5 Compatibility

The proposed application will be able to run in various type of android devices as well as different version of android.

3.1.6 Security

Data are kept safe by encryption and login is required to have access. Security measures like protection against SQL injection or encryption algorithm will follow standard.

4. Appendix A: Glossary

User Interface (UI): The front end of the system through which the user interacts with the system functionalities.

AWS (Amazon web services): Is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis.

Cognito: Amazon Cognito is an Amazon Web Services (AWS) product that controls user authentication and access for mobile applications on internet-connected devices.

DynamoDB: Amazon DynamoDB is a fully managed proprietary NoSQL database services that is offered by Amazon.com as part of the Amazon Web Services portfolio.

IAM (Identity and Access Management): AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

S3 (Simple Storage Service): A cloud storage service provided by amazon web services.

5. References

[1] Amazone, "User Sign-in," Amazone Web Services , [Online]. Available: https://docs.aws.amazon.com/mobile-hub/latest/developerguide/user-sign-in.html.

[2] Amazone, "User Data Storage," Amazone Web Services, [Online]. Available: http://docs.aws.amazon.com/mobile-hub/latest/developerguide/user-data-storage.html.

[3] Amazon, "NoSQL Database," Amazon Web Services, [Online]. Available: http://docs.aws.amazon.com/mobile-hub/latest/developerguide/nosqldb.html.

[4] Amazon, "What Is IAM?," Amazon Web Service, [Online]. Available: http://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html.

[5] Amazon, "AWS Mobile SDK," Amazon, [Online]. Available: https://aws.amazon.com/mobile/sdk/.