PHOTOCACHE

SOFTWARE DESIGN OVERVIEW DOCUMENT

Berkay AKYAZI
2035558

Ömercan GÖKKKAYA
2035954

Veli Özgür YILDIZ
2036366

Ali BAYRAMÇAVUŞ
2035723

1 Introduction

This document is prepared by following the IEEE STD 1016-2009. This document contains content and organization for software design details and devolopment way of PhotoCache project. The details and contents are represented by using UML context, class, component and deployment diagrams.

1.1 Product Description

Our phone are not merely cameras that we record our valuable moments, they are also albums through which we can share these moments with friends. However, Phones come with extremely limited storage options, and most people often opt to pay hundreds of dollars to get some more storage. Nevertheless, the phone storage fills up pretty fast, and we will provide a solution to this problems.

This project aims to develop a "gallery" application running on the phone and a server running on a desktop that would essentially use of the phone as a "cache" for all the photos that are stored on the users desktop "server". The system will provide four modes:

• When the phone gets a wi-fi connection to the server, it automatically uploads all the new photos to the server. No photos are deleted in this process.

• When the phone storage gets low, the photos that are back-upped are automatically deleted to create space for the new shots automatically. Thumbnails for the deleted photos are kept on the phone.

• The gallery app on the phone keeps track of the access statistics for the photos, and uses this information to decide on which photos be deleted. The app will use the on-board storage on the phone as a cache for the photos stored on the server. Frequently accessed photos will never be deleted from the phone, while photos that are not accessed will eventually be deleted (reduced to a thumbnail) to create space for new photos.

• The gallery app will allow the pinning of photos to prevent their deletion as well as provide options for delete from phone and delete from everywhere and delete from server.

• If the server is available online on the internet, the phone should be able to fetch the full-resolution version from the server to the phones storage.

2 References

This document is created by using references:

• **IEEE International Standards Document:** IEEE Std 1016-2009 IEEE Standard for Information TechnologySystems DesignSoftware Design Descriptions

3 Glossary

IEEE : Institute of Electrical and Electronics Engineers

UML: Unified Modeling Language

DBMS : Database Management System

Content management : a set of processes and technologies that supports the collection, managing, and publishing of information in any form or medium.

Front-end web development : the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly.

Subclass : Child class inheriting attributes and methods from super-class.

Super-class : Parrent Class

HTTP : (Hypertext Transfer Protocol) functions as a requestresponse protocol in the clientserver computing model .

TCP/IP : Transmission Control Protocol (TCP) ve Internet Protocol(IP) are used for secure data transfer.

UDP : User Datagram Protocol is used for data transfer.

Hole-Punching : Hole punching is a technique in computer networking for establishing a direct connection between two parties in which one or both are behind firewalls or behind routers that use network address translation (NAT).

JSON : (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generat

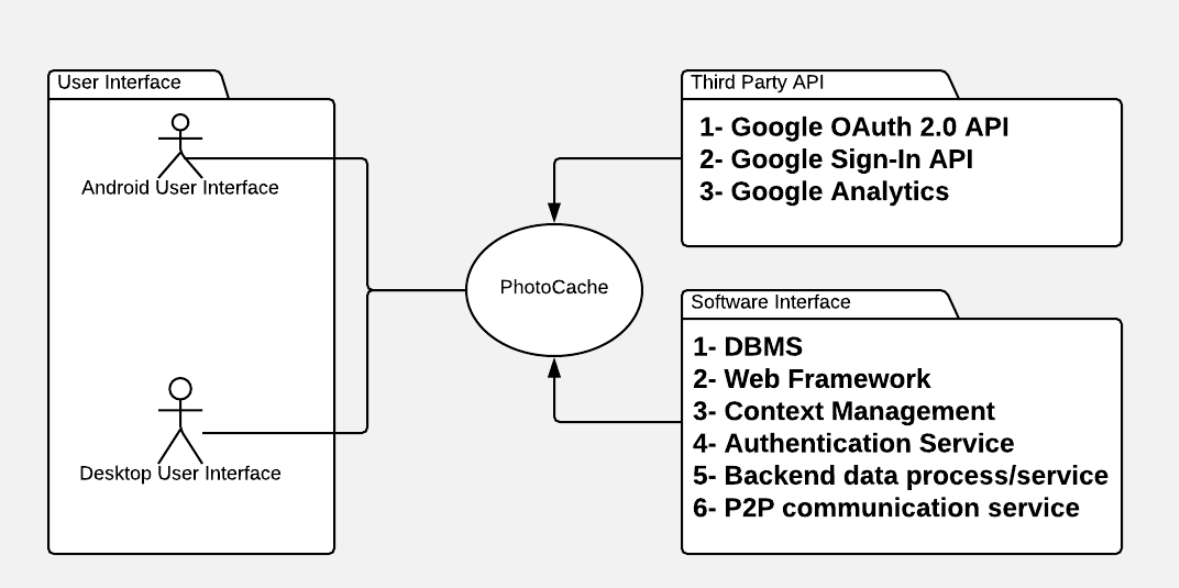
Authentication : Authentication is act of confirming the truth of user or data.

MySQL Workbench : MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more.

4 Architectural Views

4.1 Context View

Figure 1:Context Diagram of the Project



4.1.1 User Interface

We have two User Interfaces. One of them is on the Android Device and the other one on the Desktop Device.

4.1.2 Third Party API

We uses Google OAuth 2.0 API for the Desktop Authentication system and Google Sign-In API for the Android Application LogIn System. Google Analytics is integrated with the central server in order to track the server requests.

4.1.3 Software Interface

• We have used MySQL DBMS for the Central Server and SQLlite DBMS for the Android and Desktop Clients.

• We have used glassfish server service as the Java Web Framework. Glassfish server handles all the requests and responses.

• All context management operations and backend data processes are handled with JAVA implementations.

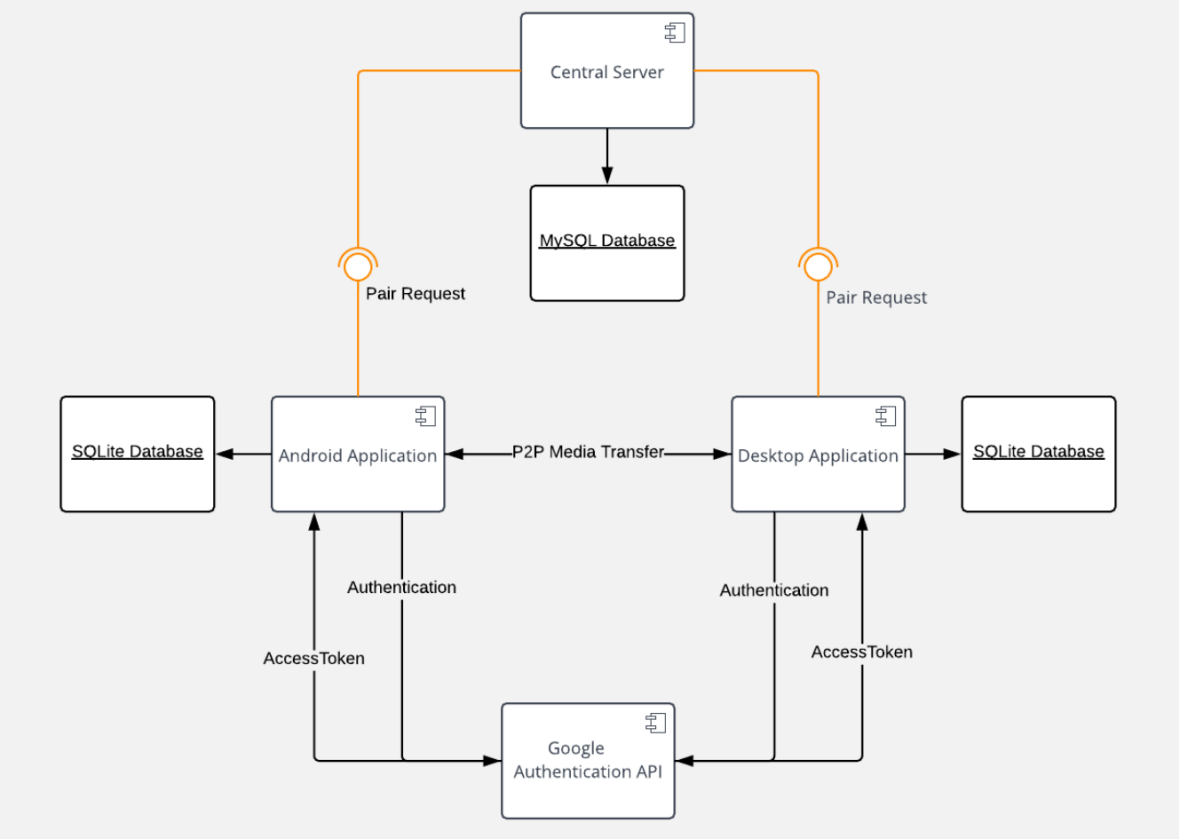
• Google Authentication service is used as Sign-in/Sign-up systems.

• We have used UDP-Hole Punching and uPnp for creating P2P communication between Android and Desktop devices.

4.2 Composition View

The composition viewpoint of the PhotoCache system is showed by UML Component and Deployment diagrams given below.

Figure 7:Component Diagram



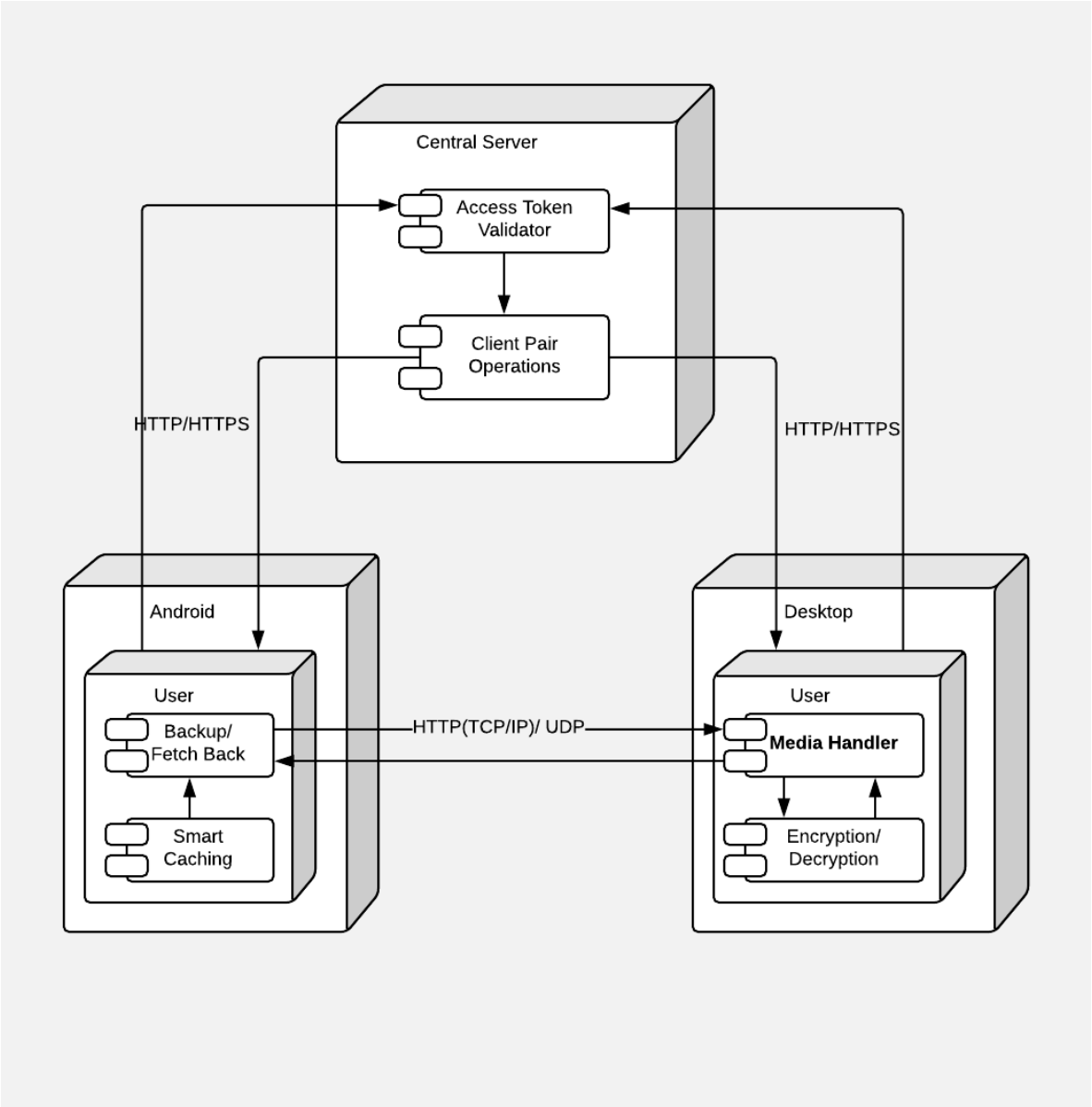
Central Server: Central Server is only used for storing user informations and pairing the user's Mobile and Desktop devices. Central server stores user's id, name, e-mail, devices etc. in the MySQL database. Also, it stores port, local and remote hosts of the user when user's devices become online and passes these information to the other online devices of the user in order to enable them to create smooth pair-and-forget backup process.

Google Authentication API: Google Authentication system is used for authenticating the users and identifying them. Both Android and Desktop devices take Access Token from Google Authentication API after authentication process.

Android Application: Android application is the source of every data in the system. The application authenticates by Google API and notifies the server about its information (username, e-mail, device type etc.) It will automatically backups the photos and videos into Desktop Device when it has wi-fi connection. It also keeps track of the free phone storage and access statistics for the photos, an also examines the user behavior in order to decide which photo should be replaced with its thumbnail. Furthermore, android application enables users to fetch back their old photos from Desktop device by using its thumbnail. Also, android application stores backed-up media in the SQLite Database in its local storage. This database is always synchronized with the Desktop database.

Desktop Application: Desktop Application consist of two system. One of them is the background service which creates a P2P communication between Android application and Desktop Server. And the other one is the Graphical User Interface that enables users to access their backed-up photos and videos. Desktop app. also uses Google Authentication system and notifies the central server after the authentication process. After it creates a P2P connection with the Android Application, it begins to listen the commands, given by Android Application. For example, some commands are backup media or recover media. Also, desktop application stores backed-up media in the SQLite database.

Figure 8:Deployment Diagram



Central Server only corresponds to authentication validation and pairing. There is no media which is transmitted over the Central Server. We have glassfish server that works on the central server. It handles HTTP/ HTTPS requests.

Android corresponds to keep tracking the phone. We have called this operation *Smart Caching*. Android app decides which media is stored or deleted.

Desktop corresponds to store the backed-up media. All the media and the databases are encrypted for extra security.

Backup/Recover Operations are implemented by using UDP hole punching when the devices are connected to the different wi-fi networks and client-server communica-

tion when they are connected to the same network (Desktop device acts as a server and android device acts as a client in this case).

Design Rationales:

• HTTPS ensures secure connection between Central Server and Android/Desktop client. This protocol guarantees safety of the exchanged data, protection of the user's privacy and authentication of the website.

• MySQL ensures security of user data because MySQL is globally renowned for being the most secure and reliable database management system and it is also used in popular web apps like WordPress, Facebook and Twitter.

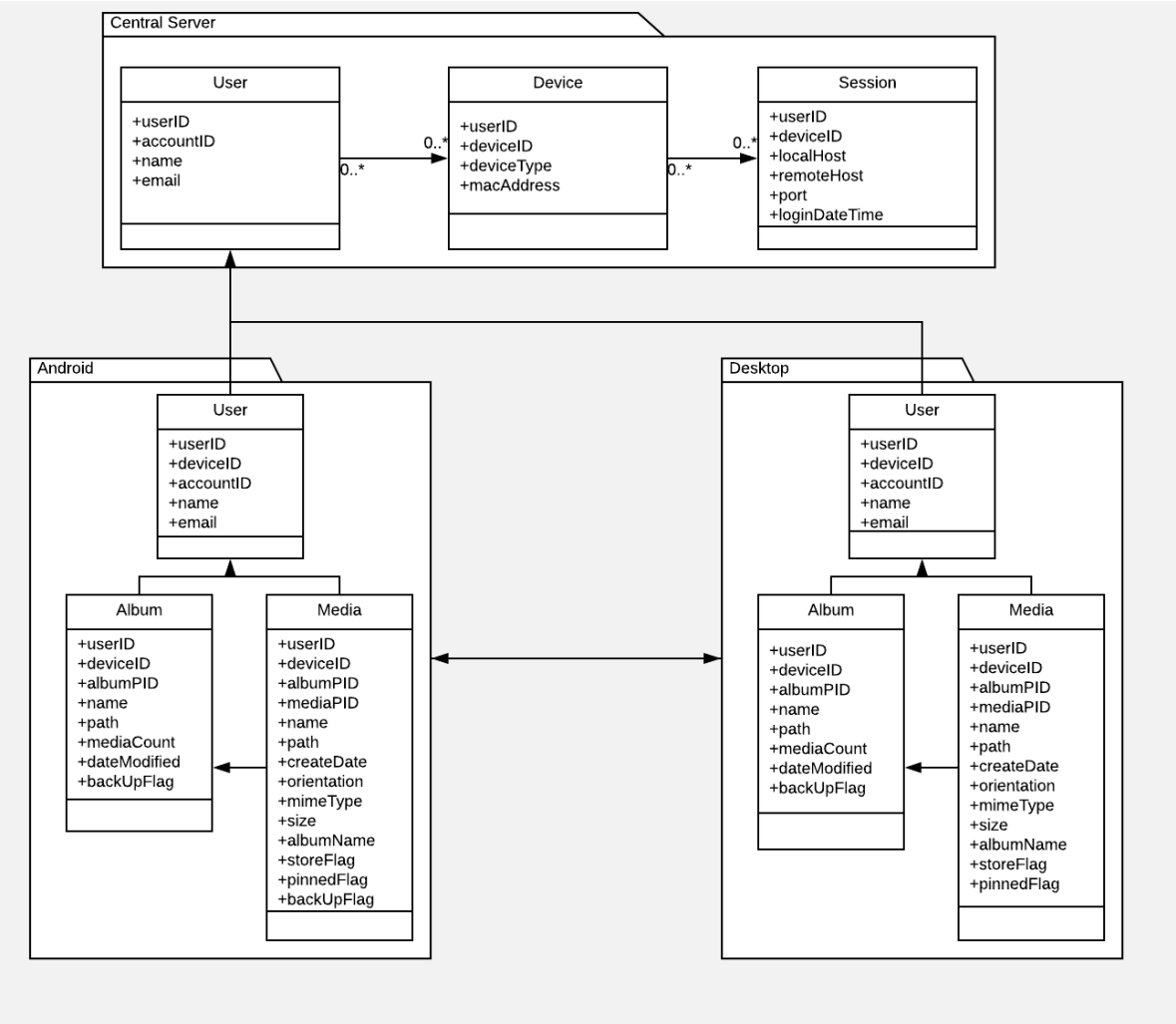
• MYSQL has high-speed transactional processing system so it meets our system functionality constraints.

• MySQL also reduces the cost of the project because it is open-source.

• We have implemented secure and reliable data transfer over UDP which does not provides any features such as security or reliable data transfer but provides faster and simpler solution.

4.3 Information View

Figure 9:Class Diagram



4.3.1 Central Server

- **User Table**

It stores userID, accountID which Google Authentication API provides, name and email address.

- **Device Table**

Since user can have multiple devices we need to store all of them separately. It stores userID, deviceID, deviceType (Desktop, Android), macAddress. MacAddress is used as unique identifier for devices.

- **Session Table**

It stores which user is online and waits for its pair. The attributes are userID, devi-

ceID, localHost (IP), remoteHost (IP), port, loginDateTime (time stamp at which user become online). After user's devices are paired, the entries of the devices are removed from the Session Database.

4.3.2 Android Client

- **User Table**

It stores userID, deviceID, accountID, name, email. We need this table since there could be multiple user which currently uses the device.

- **Album Table**

It stores all the albums in the android device. It stores userID, deviceID, albumPID, name, path, mediaCount, dateModified, backupFlag(used to indicate whether the album is backed-up or not.)

- **Media Table**

It stores the media info if the user changes the media's flags. The flags are storeFlag which shows the media is still stored in the phone or not, pinnedFlag which shows whether user is pinned the media or not and backUpFlag shows is the media backed up or not. Also Media Table stores, IDs, media name, path, size, creation date, orientation, mime type(jpg, png, mp4 etc.), albumName.

4.3.3 Desktop Client

- **User Table**

It stores userID, deviceID, accountID, name, email. We need this table since there could be multiple user which currently uses the device.

- **Album Table**

It stores all the albums in the android device. It stores userID, deviceID, albumPID, name, path, mediaCount, dateModified, backupFlag(used to indicate whether the album is backed-up or not.)

- **Media Table**

It stores backed-up media info. Media Table stores, IDs, media name, path, size, creation date, orientation, mime type(jpg, png, mp4 etc.), albumName, storeFlag, pinnedFlag.

4.4 Alternative Design Options

• We may use Google Drive or another cloud service as a intermediate layer for transferring media between devices. But, we want to give full privacy to the users. However, these services forces the users to give up their privacy by storing them in the cloud.

• We may develop Desktop Application with C++. It will be faster but it could be much more hard to implement, and problematic to compile on the different operating systems.