CLOUD DROP

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of

the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Preetham V Dhondaley

FALL

2016

CLOUD DROP

A Project

by

Preetham V Dhondaley

Approved by:

_____, Committee Chair

Dr. Jinsong Ouyang

_____, Second Reader

Dr. Jun Dai

_____

Date

Student: Preetham V Dhondaley

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

_____, Graduate Coordinator _____

Dr. Jinsong Ouyang Date

Department of Computer Science

Abstract

of

CLOUD DROP

by

Preetham V Dhondaley

CloudDrop proposes to provide a quality system to support cross-platform file transfer mechanism between Android and IOS devices with cloud storage. The devices include iPhones, iPads, Android phones, and Android tablets. The current status of this application and the approaches used to solve some of the critical problems and features are presented in this report.

_____, Committee Chair

Dr. Jinsong Ouyang

_____

Date

ACKNOWLEDGEMENTS

I am using this opportunity to express my gratitude to everyone who supported me throughout this project.

I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work.

I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

I express my thanks to Dr. Jinsong Ouyang and Dr. Jun Dai for their support and guidance throughout this project.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figures Page

LIST OF TABLES

Tables Page

1. INTRODUCTION

There has been a significant increase in the number of mobile devices in our household, and we continue to deal with an increasing number of problems due to platform compatibility. One problem is on how to transfer files between between different platforms such as iOS and Android devices. Although there are options to transfer files from an Android device, Apple has always been against the whole concept of file sharing between devices. They believe in sharing between applications though.

One of the main reasons for not incorporating the sharing feature was because it tends to compromise the performance of the system in terms of resource management and system security. Apple follows the concept of resource ownership where a resource should be owned by an application, unlike android where the user can download the files directly into the device and manage the files on their own.

We continue to explore new techniques to overcome the compatibility problems between these two platforms. In this project, we are developing a solution to one of the major cross-platform file sharing problems.

2. BACKGROUND AND MOTIVATION

There are a lot of applications available in the Android market which supports file sharing, but there are very few applications which support cross-platform sharing.

CloudDrop is a solution for sharing files between multiple platforms, and it provides an additional features where you can group all your home and office devices, and with a single touch, you can transfer all your selected files to the devices in the group. A group can have devices with different platforms, and a device can belong to multiple groups. CloudDrop also provides a cloud storage where the users can upload all the important documents on the cloud, and download it whenever needed.All the shared files can be stored on the cloud by default, unless the user decided not to.

Users can also organize their files by creating folders and naming them accordingly.

The user is expected to install CloudDrop on all their devices by downloading it from the respective Application Stores. The user should use the same login credentials on all the devices in order to use this this application. Once the application has been installed and logged in, the user can start sharing files by selecting the files that needs to be shared, and

selecting CloudDrop on the share option which displays a list of destination devices/ groups available for sharing.

CloudDrop will be available for iPhones, iPads, Android Phones, and Android Tablets. It can be downloaded from the Apple's app store and Google's play store.

Due to Apple's restrictions on music files, e-books, and all the other purchased media, they cannot be shared on CloudDrop. Although these type of files can be shared from the IOS if they are provided by the third party applications. An audio player has been specially designed for the IOS version to play downloaded music.

Since Apple has a strict policy on background services, large files may not be downloaded in the background. But CloudDrop has been well designed to handle such situations and the user will have a subtle experience.

3. DEVELOPMENT ENVIRONMENT

Table 1. Development Environment

Development Operating System	MacOS 10.10-10.11
Android IDE	Android Studio
IOS IDE	Xcode
Server	Apache
Database	mysql
Android Client Programming Language	Java
IOS Client Programming Language	Objective-C
Server Side Language	PHP
Test Server	Localhost
Production Server	Hostgator Shared Web Hosting

Project Start Date: May 2015

Expected Completion Date: November 2016

4. SYSTEM DESCRIPTION

The following figure shows depicts a very high level description of CloudDrop's system architecture.

The system is divided into 4 major modules.

- Source Device: Device initiating the file transfer to the destination device.
- Cloud Device (CloudDrop Server): Resides temporary files that needs to be pushed to the target devices and user's cloud files.
- Destination Devices: Could be a cloud or any mobile device or devices receiving the incoming file.
- Communication Channel: The communication approach used to transfer the files between the devices and CloudDrop server.

The following sections will describe each of these module's architecture in detail.

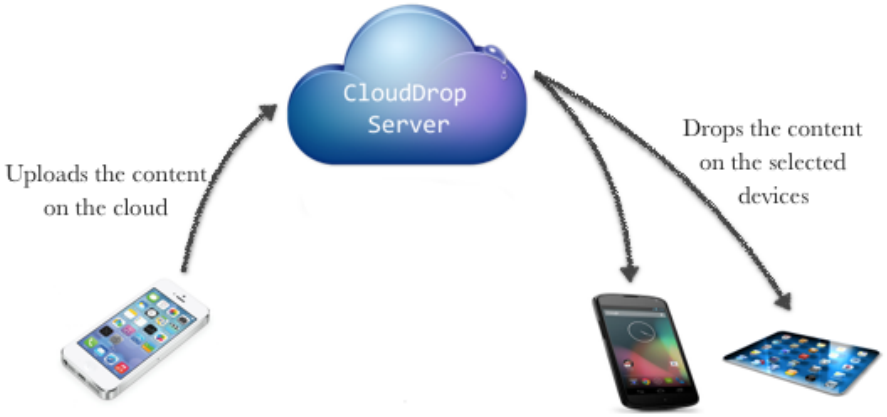

Figure 1. System Architecture

5. SYSTEM ARCHITECTURE

5.1 CloudDrop Client

CloudDrop is currently compatible with both IOS and Android platforms. But since there are many similar applications on the Android store, CloudDrop was primarily built to support IOS users who also own Android devices. Hence we have invested most of our time and resources to build a great application for the IOS users.

CloudDrop comes with a music player with hands-free support exclusively for the IOS version as Apple doesn't support downloading and playing music without iTunes.

The IOS users can also stream music directly from the cloud without buffering the entire audio file.

5.2 **Client Architecture**

As shown in the below figure, the client architecture of both IOS and Android is divided into 5 major modules. Each of these modules are coordinated by the "System Coordinator" module. The lines represent a relationship between those modules.

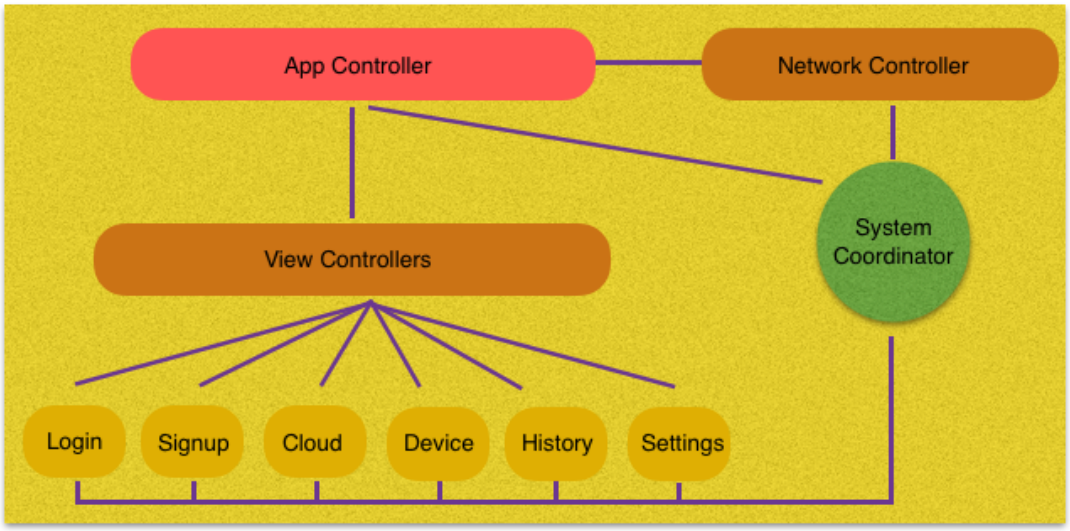

Figure 2. Client Architecture

5.3 App Controller

Application controller is the primary controller which handles all the low level events such as enabling and disabling applications, backing up data, handling push notification, instantiating View Controllers, Network Controllers, and System Coordinator.

The app controller is also responsible for playing sound when notification is received.

Application controller is the first class that is instantiated when the application is action.

It's referred as AppDelegate on IOS and MainActivity on the Android version.

5.4 Network Controller

Is a singleton class which broadcasts network related events to the application. Some of the events are

- Device's network: Notifies the application when the device's internet connection status is changed.
- Server's reachability: Notifies the application when the server's availability has changed.

Although these modules may not significantly be visible to the user, it assures best experience to the users and help troubleshoot problems when the application is experiencing unexpected events.

5.5 System Coordinator

Is a singleton class shared across all the other classes on CloudDrop. This class is instantiated by the App Controller. The system coordinator resides all the data needed by the application during the runtime. The data includes server's ip address, authentication tokens, app defaults, and setters and getters for other classes to access these information.

Its named as "config" on both IOS and Android version.

System Coordinator is also responsible for fetching the device credentials from the server. This credentials are used to request service and authenticate the device to the server. Since these credentials are very critical as its a primary source of representing a devices, they are are stored in the "KeyChain" app os IOS and "Shared Preferences" on Android. Both these application are internally managed by Apple and Google.

The System Coordinator also provides a mechanism to retrieve configuration keys through KeyChain and Shared Preference by abstracting the implementation layer. On IOS, the coordinator takes care of generating the music playlist for the hands-free control.

5.6 View Controllers

View controllers manages its views. There are six view controllers on CloudDrop.

- History View Controller
- Devices View Controller
- Grouped Devices View Controller
- Cloud View Controller
- Settings View Controller
- Login/Signup View Controller

5.7 History View Controller

Displays all the files previously downloaded and also receives requests to download a files from the server. The user can view the downloaded files using the default application set for handling that specific file type.

CloudDrop supports variety of file types such as music files, video files, compressed files, document files, image files, and pdf files. On IOS, the user will prompted with the list of applications that can be used to open the file type. And on Android, the user will be prompted with the list of applications unless the user has not set any default application to handle the file type.

Since apple doesn't support playing online music on its music application, a music player has been specially built for the IOS version to fill in this gap.

5.8 Devices View Controller

Displays all the devices associated with this user account. Currently CloudDrop only supports IOS devices (iPhone and iPad) and Android devices (phones and tablets). This can be extended to support Windows and MacOS machines.

User can unassociated any device from the device list and all the files and credentials will automatically be erased from the device once unassociated. The account will be automatically logged out once the device has been unassociated.

5.9 Grouped Devices View Controller

Displays all the device groups. User can group devices together with a comfortable name. This feature is specially useful when the user wants to share files with a group of devices with a single click. For an example lets say that the user has an iPhone, iPad, and an Android phone for home use and two iPads and an iPhone for work. User can group them together and share files from any device to the target group. The files will automatically be downloaded to the target devices.

Users can achieve this feature by selecting share option on any file and selecting CloudDrop from the list of applications.

5.10 Cloud View Controller

Files shared across devices are automatically stored on the cloud unless “Auto Backup” is disabled in the settings. Users can browse and manage all the cloud files. Folders can be create to organize the files. Folders in reality doesn't actually exist in server; they are only simulated.

The following figure describes how folders and files are managed on the CloudDrop server.

Each and every cloud file has an unique id with user_id as the owner of the file. Parent_folder is just another id whose reference key is a primary key of cloud_file. Which means parent_folder points to the same table in the database unless is_root is set to “no”.

File_id is set if the file_type is not “folder”. File_id points to the primary key in the “files” table.

Name	Type
id	int(15)
user_id	int(15)
parent_folder	int(15)
file_type	enum('file', 'folder')
file_id	int(15)
folder_name	tinytext
is_root	enum('yes', 'no')
date_created	timestamp

Figure 3. Cloud_Files

5.11 Settings View Controller

Users have access to logout, delete account, change password, and also activate/ deactivate cloud services through the settings view. Once the account is deleted, the files are automatically erased. But once the user logs off, they are asked to retain the files or to delete it. On IOS, the data can be viewed outside the application by using a third party application but not on IOS.

5.12 Login/Signup View Controller

Is the first view to be presented to the user when the application is opened for the first time. It displays two Text boxes to enter User ID and Password. But it doesn't exist technically.

CloudDrop has been designed to provide a good quality security to the users which compromising on the usability.

Users entering confidential information like password on an untrusted device may jeopardize their account as the information might get leaked if the phone is compromised. To prevent this, restricting client from getting access to any confidential information is what i think is the safest solution.

Nearly a 30 days were spent on just designing the login/signup system. And the result was just incredible! We have designed a series of authentication mechanism to make sure that the password is transported to the server securely without client asking users to enter the password.

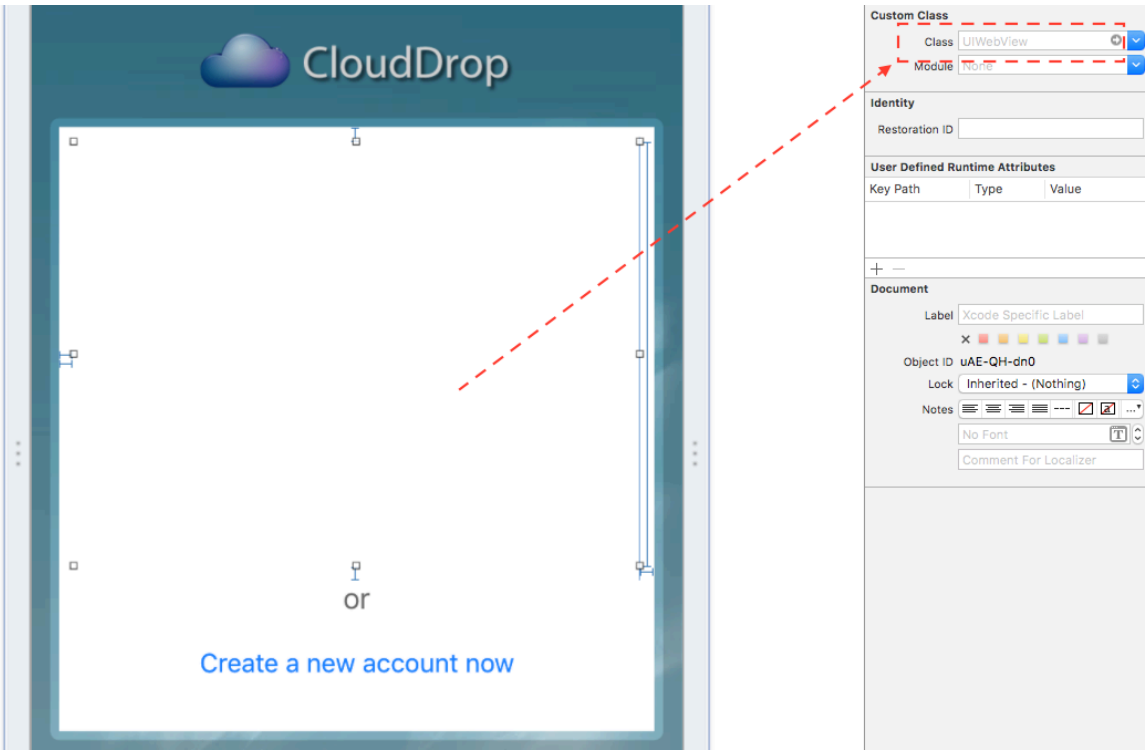

Figure 4. Login View Skeleton

From the above figure, it’s very clear that there are no Text Boxes to enter User-Id and Password. Instead we see a WebView and it’s directly linked to the CloudDrop server.

Server takes care of displaying all the contents required for logging into the account and the client doesn't have to implement anything on its end.

This way, the client doesn't have an access to the user’s login credentials, hence no need to worry about device getting compromised.

6. LOGIN/SIGNUP MECHANISM

The following figure depicts the sequences for requesting login view from the CLoudDrop server.

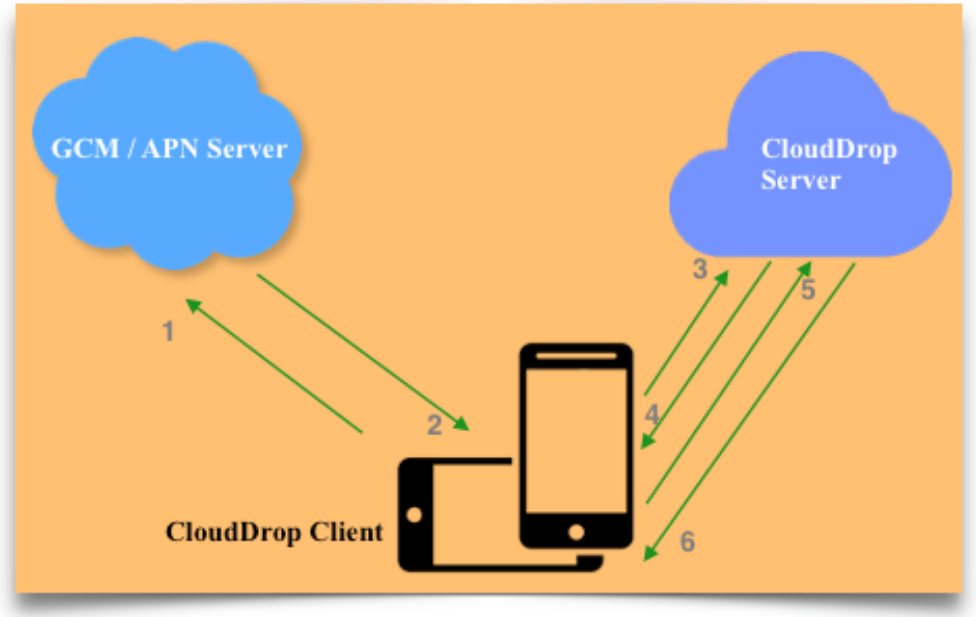

Figure 5. Login Sequence

6.1 Login Sequence

1. After CloudDrop is installed, the Android requests Google's GCM server (Now FCM) and the IOS requests Apple's APN server for it's push notification credentials.
2. CloudDrop receives such notification credentials.

3. The client sends the push notification credentials along with other device information to the CloudDrop server.
4. CloudDrop server processes the information and via a push notification, sends one time access-key and access-token which can be used to request a login/signup view
5. Client then uses these access tokens to request for a login view
6. Server processes this request and returns a http response with required html contents to be displayed to the user.

User enters sensitive credentials directly into the html form and submits it to the CloudDrop server. Depending on weather the client requested for Signup or Login service, server process this request as follows.

- **Signup Request**: Sends an email with OTP(One Time Password) to authenticate the email address. The user is expected to enter this OTP when prompted to complete the signup process.
- **Login Request**: After the user is authenticated, server stores device's push notification credentials, device name, device model, and the Operating System; and responds with a JSON encoded Device Id, and Access Token. Server also sends a push notification to the device to move the user to the application's home view.

  The client is expected to securely store these credentials, and these credentials

will be used as a primary means of authentication hereafter. The client will use these credentials for all the requests made by it. If authentication fails, the server just ignores the request.

7. CLOUDDROP SERVER

CloudDrop's server has been built completely on PHP and MySql. Gateway.php is the reporting page for all the incoming requests.

There are currently 23 general purpose methods and 20 cloud methods supporting both GET and POST methods.

Basic request syntax looks like ~gateway.php? request_type=<some request> & device_id=<device id> & device_token=<some token>

7.1 General Purpose Methods

Table 2. General Purpose Methods

Method Name	Description
isValidUserNameAndPassword	Returns returns user id if true or returns false for the given user id and password
validateDeviceIdAndAccessToken	Returns returns user id if true or returns false
getPushIdForDeviceId	Returns push id if true else false for the given device id
getIsInvalidatedForDeviceId	Returns true if devices is invalidated else returns false
logOutDevice	Logs out the device by sending push notification

Method Name	Description
isEmailExisting	Returns true if email exist or returns false
getValidateLoginViewAccessKey	Returns the login view access code if true else false
getValidateLoginViewId	returns the id of loginViewRequest if true else false. validates post(access_id,access_key)
validateLoginView	validates login view access. validates post(access_id,access_key) returns true or false
validateLoginViewRequest	pushes login access_id and access_key. validates post(device_notification_id,device_OS,device_name,device_model)
generateRandomString	Returns a random string of specified length
signupView	Returns signup webview
loginView	Returns login web view
getDeviceNameForDeviceId	Returns device name for device id
dispatch_request	Outputs the file with appropriate headers
getFileIdForOutboxId	Return the file id of a record in the outbox table
decrementReferenceForFileId	Decrements the reference count of a file in the database
incrementReferenceForFileId	Increments the reference count of a file in the database
getReferenceForFileId	Returns a reference count for the file id
getDeviceOsForDeviceId	Returns device OS ex:IOS,android... -1 if error
deleteFile	deletes the file from disk and database; returns true or false in failure

7.2 Cloud Methods

Table 3. Cloud Methods

Method Name	Description
addFileToCloud	adds fileId to cloud.if folder id is -1 inserts to the root folder else to its respective folder
checkFolderOrFileExistForUserId	Return true if folder or file exist for user id ;else returns false
getParentFolderForFileId	Returns parent folder id.0 is invalid file specified, -1 if the fileId is root or will return the actual parent id
createRootFolderIfNotExistAndReturnId	Creates root folder if not exist and returns root folder id
getFolderContents	Returns count -1 in JSON if folder doesn't exist else returns json data of the folder contents with contents
validateAccessForFolder	Checks if the userid has access(is an owner) to the folder
variableDump	Returns var_dump
writeToFile	Writes the content to file
dispatch_file	Outputs the file with appropriate headers
fileIdForCloudFile	Outputs the file actual id for the cloud file id
createCloudFolder	Will create a new folder in the parent folder and return its id or -1 if error
moveCloudFolder	Will mmode cloud folder to new parent directory, will return true/false
renameCloudFolder	Will name a cloud folder or -1 if error
isCloudEnabledForUser	Will check if cloud is enabled for user id; true or false or -1 for error
deleteDirectoryRecursively	Deletes the cloud folder and its contents recursively; and deletes the file if reference count ==0

Method Name	Description
isFolder	Checks if the fileId is a folder or not, true if folder else false
getFileIdForCloudFile	Returns the actual file id for the cloud fileId

7.3 Server Request APIs

The following table lists all the requests that can be made to the server

Basic Syntax: http://clouddrop.steelwing.us/gateway.php?request_type=<Request> & <parameter> = <Value for the parameter > & . .

Table 4. Server Request API Calls

Request	Parameters	Description
login_view_request	• device_notification_id (GCM/APN) • device_OS(Android/IOS)	• Login view can be requested by providing notification id and device OS type • Response JSON Encoded Via Push Notification: {response_type:login_view_request, access_id=<access id>, access_key=<access key>}
signup_view	• access_id • access_key	• To request for a signup view • Direct response : HTML content to be displayed on the web view
login_view	• access_id • access_key	• To request for a login view • Direct response : HTML content to be displayed on the web view

Request	Parameters	Description
get_device_credentials	• access_id • access_key	• To receive device id and device token after login • Direct Response JSON Encoded: {device_id:<device id>, device_token:<Device Token>, } else {error:1, message:"Access Denied"}
device_list	• device_id • device_token	• Returns the list of devices associated with the account • Direct Response JSON Enccoded: "message"==>{response_type:device_list,devices: [{device_id:<Device Id>, device_token:<Device Token> }]} or error "message"==>{"error"==>error_type:"incorrect_credentials"}
upload_file	• device_id • device_token • uploaded_file • destination_id • destination_type(device/cloud)	• Upload a file to the destination device • uploaded_file is the "File" tag of an html form painting to the binary file
fetch_file	• device_id • device_token • request_id	• To fetch a file from the server • Request_id is required to fetch a file. Its received through push notification which will be discussed in the next section • Returns binary
incoming_file_list	• device_id • device_token	• Returns a JSON list of files to be downloaded by the client • "message"==>{response_type:multiple_response,commands=>[{response_type:fetch_file,file_name:<File Name>,request_id=<Request Id>,mime_type:<MIME Type>}]}

Request	Parameters	Description
cloud_folder_contents	• device_id • device_token • folder_id(optional) • root_folder(true/ false)(optional)	• Returns a JSON list of all the files in the cloud directory • folder_id or root_folder should be specified and not both • folder_id is the folder id we are interested in • root_folder, if true, returns contents of the root directory associated with the account
junk_devices	• device_id • device_token	• Use to report a duplicate device or a junk device which is no longer reachable via push notification. • Used during situations when user uninstalls the app without signing out. And when installs back, the device is registered twice in the server
download_complete_notify	• device_id • device_token • request_id	• Used to notify the server that the client has finished downloading the file and no longer requires the request. • The request id will then be invalidated by the server and will no longer be usable
buffer_file	• device_id • device_token • file_id	• User can download cloud files through this method call • file_id is the cloud id of the file
create_cloud_folder	• device_id • device_token • parent_folder • folder_name	• User can create a new folder on the cloud
rename_cloud_folder	• device_id • device_token • folder_id	• User can rename a cloud folder

Request	Parameters	Description
move_cloud_folder	• device_id • device_token • folder_id • new_folder_id	• User can move a folder to a new parent folder(new_folder_id)
delete_cloud_file	• device_id • device_token • folder_id	• User can delete a cloud file/folder where folder_id can be a file or a folder

7.4 Server Responses

This table represents the list of responses received by CloudDrop server via Push Notification. These are responded regardless of any request made. For an example "log_out" command may be issued to the client from the server anytime.

Basic Syntax: {response_type:<Response>,<Parameters>:<Parameter Values>}

Table 5. Server Responses

Response	Syntax	Description
multiple_response	{response_type:multiple_response,commands: [{response_type:<some type, <parameter>:parameter value>…}]}	• When there are multiple responses, CloudDrop couples multiple responses together . • The client is expected to parse all the individual responses accordingly.
fetch_device_credentials	{response_type:fetch_device_credentials}	• Server is asking the client to fetch its new credentials i.e Device Id and Device Token
login_device	{response_type:login_device}	• Server is asking the client to login the user to its home view. Happens when the user has successfully logged in
log_out	{response_type:log_out}	• Server is asking the client to logout the device. Happens usually when the user has deleted the device or the device has been invalidated • Push notification service
user_alert	{response_type:user_alert,message:<some message>}	• Server is sending a message which needs to be shown to the user • Push notification service

Response	Syntax	Description
fetch_file	{response_type:fetch_file, file_name:<File Name>, request_id:request id>, mime_type:<file mime type>}	• Client receives this when it has a new file to be fetched from the server • In the message section of the notification, the client is supposed to display the message to the user. It contains information like “Received image from iPhone”

8. SERVER DATABASE

8.1 About database

CloudDrop is powered by mysql database with currently eight tables online. Its designed to be efficient meeting the third normal form to avoid any duplications.

Third normal form is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that [1] the entity is in second normal form, and [2] all the attributes in a table are determined only by the candidate keys of that table and not by any non-prime attributes. 3NF was designed to improve database processing while minimizing storage costs. 3NF data modeling was ideal for online transaction processing (OLTP) applications with heavy order entry type of needs

8.2 Cloud Files

This table abodes all the files and folders that are stored on the cloud. We can also see that the files are pointing to the actual “files” table in the database and does not have any duplication.

This table also simulates the directory feature as cloud directory technically doesn't exist! So if the user wants to move a file to a new folder, technically the parent_folder of the file is updated with its new parent id.

Table 6. Cloud Files

cloud_files	
id	Primary key
user_id	Reference key to the “users” table; Owner of the file
parent_folder	Reference key to the same table; Null if its a root folder
file_type	“file” if its a binary file or “folder” if its a folder
file_id	Reference key to the “files” table; Null if its a folder
folder_name	Stores folder name; Null if its a file
is_root	“yes” if its a root or “no” if its not a root folder or a file
date_created	Timestamp of this record

8.3 Devices

This table abodes devices associated with the user account. User 0 or more devices associated with the account. Every time a user logs in, the device used to log in is automatically registered and will show up on the devices list.

Table 7. Devices

devices	
id	Primary key
device_name	Reference key to the “users” table
device_os	“android” or “ios”
device_model	Model name of the device
user_id	Reference key to the “users” table; Owner of the device
date_created	Timestamp of this record
push_id	Code used to send push notifications to the device
access_code	Code used to authenticate the device
invalidated	Set to 0 if device is active or set to 1

8.4 Email Verification

This table help in validating email id when the user signs up for a new account. CloudDrop send an OTP(one time password) to the users email to verify if they truly own the email id.

Table 8. Email Verification

emailVerify	
id	Primary key
email	Email id of the user to be registered
otp	One time password generated for the email id

8.5 Master Files

This table stores all the file informations uploaded on the server. When a file is uploaded, CloudDrop automatically extracts headers like the file name and the file extension to safeguard user's privacy. The original file name is replaced with a randomly generated 50 characters long name which is unique. When a client requests for a file, CloudDrop server automatically builds the file with its original headers before dispatching it. To efficiently store files on the server without having to make duplicates, CloudDrop has a

feature which keeps a count of members currently accessing the file. Lets say that a user wants to upload a file to 3 devices. Its really important that we retain that file on the server until the client acknowledges that it has received the file. We may come across situations where the client is not connected to the internet and the server has to wait until the client is back online and receives it package. In these type of scenarios instead of keeping 3 copies of the same file, CloudDrop only keeps 1 copy and just increments the reference count to 3. Storing the file on the cloud will also increment the reference out of the file. The files are only deleted when there are no more references made to the file i.e when the reference count is 0.

Table 9. Master Files

files	
id	Primary key
file_name	Original name of the file
random_name	Random name generated by the server to securely store it
mime_type	The mime type of the file
date_added	Record date
user_id	The id of the user owning this file
reference_count	Count of objects accessing this file
file_location	The location of the file stored in the server

8.6 Login View Requests

When a device requests for a login view to login its users, it is first supposed to register itself with its notification provider and then share it's device name, device model, and device type with the CloudDrop server. CloudDrop processes this information and generates a one time password and sends it via push notification to the client which can later be used to request the login view. This tables stores all the information used to support this feature.

Table 10. Login View Requests

loginViewRequest	
id	Primary key
otp	One time password generated to view login view
device_type	Its "IOS" or "Android"
device_name	Name of the device, expected from client
device_model	Device model name
device_push_id	APN/FCM id for sending push notification
request_time	Record creating time stamp

8.7 File Outbox

When a file is to be sent from a device to another device, a transaction record is created between the sender and a receiver. If the sender is sending the file to multiple devices, multiple transaction records are created between the sender and the receiver. This table keeps all the transaction information. From_id is the device id of the sender and to_id is the device id of the receiver. File_id is the file to be sent which is already residing in the server. There are 4 possible status for each transaction as shown in the following table. The status is set as “delivered” when the receiver is finished downloading the file, or set to “initiated” when when the receiver has requested to download the file but hasn't completed downloading it, set to “declined” when the receiver refuses to download the file, or set to “unattended” when the receiver hasn't initiated the download process. By default the status is set to “unattended”.

Table 11. File Outbox

outbox	
id	Primary key
from_id	device id sending the file
to_id	target device id
file_id	file id to be sent, referenced to files table
date_created	record time stamp

outbox	
status	“delivered”,”initiated”,”declined”,”unattended”

8.8 Playback Duration

As mentioned earlier, CloudDrop also supports streaming music files directly from the cloud. So that user doesn't have to download the entire music file or buffer the whole file to listen to the music. In order to stream any media file, the client needs to know the duration of the media file so that it can show a progress bar of the playing media. CloudDrop calculates the length of the music file and stores it in this table to be used later by the client.

Table 12. Playback Duration

playback_duration	
id	Primary key
file_id	Id of the file, references files table
duration	duration of the media in seconds

8.9 Users

This table stores details of registered CloudDrop users. Cloud_activated is set to "1" when the user has activated cloud feature or set to "0" when the user has deactivated the cloud feature.

Table 13. Users

users	
id	Primary key
email	Email id of the registered user
password	Hashed version of the password
created_date	Time stamp of when the account was created
cloud_activated	"1" is enabled or "0" is disabled
user_ip	Ip address from where the account was created

8.10 Device Groups

Users can group their devices to share files across those devices with one click. This table holds details of the groups created by the user.

Table 14. Device Groups

device_groups	
id	Primary key
group_name	Email id of the registered user
user_id	Hashed version of the password
date_created	Time stamp of when the account was created

8.11 Device Group Members

This table is used to associate a device with a group. When a device is added or removed from a group, its information is stored here. A group cannot have duplicate devices. CloudDrop makes sure that a user will not be able to add duplicate devices in the group or an unauthorized device in the group.

Table 15. Device Group Members

device_groups	
id	Primary key
group_id	Group id referencing device_groups table

device_groups	
user_id	Hashed version of the password
device_id	Device id referencing devices table
date_created	Time stamp of the record

9. ENABLING CLOUDDROP

This section of the document will guide setting up CloudDrop on a development environment. Since CLoudDrop was completely developed and tested on a MacBook, the instructions provided is targeted for MacOS. g CloudDrop.

9.1 System Requirements

-Apple computer running OSX 10.10 or higher

-XAMPP supporting PHP 5

-Xcode 8 or higher

-Android Studio 4 or later

Install XAMPP on your MacOS. XAMPP will install Apache server for servicing our web requests and also it installs PhpMyAdmin which is useful for managing your database. Copy the server source code i.e “clouddrop” folder to the htdocs directory. Under connections, open “clouddrop.php” and set your database user name and password.

Open phpmyadmin and create a new database called “clouddrop” and import clouddrop.sql into your database. This file has all the sql commands to create tables to support CloudDrop.

Before we go ahead and start configuring the client applications, make sure an Apple developers account is purchased if you are planning to run this application on an Apple device. It's not possible to receive notifications on simulator so you might have to test this application on real device. You can google for setting up provisioning profiles to test your application on your device. You will also need to link your .pem file in functions.php.

Use developers login credentials to sign in to xcode developers account and select the appropriate team for the provisioning profile.

Once the server has been configured, the client application needs to be configured with the right server IP address. The IP address is hardcoded on config.m on IOS and values.xml on Android. Once this is configured, Client apps can be compiled and you should be able to see the application running normally.

10. CONCLUSION

Although similar applications like CloudDrop are available on Android's Play Store, it is still unique considering that it supports multiple operating systems and provides cloud storage.

Without much efforts, server can be extended to support other operating systems like Windows and MacOS. The server side is completely stateless and without having to access its source code, developers can build their own client application by using the right API calls as described in the previous section of this document. For an example IOS supports streaming only music files. But developers can take advantage of CloudDrop's streaming API to stream any media files stored on the cloud. The communication channel is encrypted in the production environment so the users need not have to worry about their data being stolen.

Apple's *Aqua* design guidelines has been referred to make sure that the IOS client conforms their rules to provide best possible user experience.

11. REFERENCES

1. "What is Third Normal Form?" Cory Janssen, Technopedia, retrieved 24 April 2014
2. Codd, E.F. "Further Normalization of the Data Base Relational Model." (Presented at Courant Computer Science Symposia Series 6, "Data Base Systems," New York City, May 24th–25th, 1971.) IBM Research Report RJ909 (August 31st, 1971). Republished in Randall J. Rustin (ed.), Data Base Systems: Courant Computer Science Symposia Series 6. Prentice-Hall, 1972.
3. Third Normal Form: https://en.wikipedia.org/wiki/Third_normal_form
4. Apple's UI Design Guidelines: https://developer.apple.com/ios/human-interface-guidelines/
5. Android's UI Design guidelines: https://developer.android.com/design/index.html
6. View Controller: https://developer.apple.com/reference/appkit/nsviewcontroller
7. Wagner, David; Schneier, Bruce (November 1996). "Analysis of the SSL 3.0 Protocol" (PDF). The Second USENIX Workshop on Electronic Commerce Proceedings. USENIX Press.

8. Intents On Android: https://developer.android.com/guide/components/intents-filters.html

9. Android Custom Views: https://developer.android.com/training/custom-views/index.html

10. IOS Custom Views: https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/Lesson5.html

11. File sharing apps in android: http://www.androidauthority.com/android-apps-to-transfer-files-from-android-to-pc-600659/

12. AirDrop: https://support.apple.com/en-us/HT204144

13. File sharing apps for IOS: http://appcrawlr.com/ios-apps/best-apps-file-sharing