Major Project Report

on

MineCloud

Submitted by

Digvijay Bahree	110600
Amit Yadav	110269
Akash Gupta	110250

(Students of B. Tech 8th semester Computer Engineering)

Under the Guidance of

R.M.Sharma

(Assistant Professor, Department of Computer Engineering)

Department of Computer Engineering
National Institute of Technology, Kurukshetra

May, 2014

To Whomsover It May Concern

This is to certify that we have worked on **major project** titled **MineCloud** as a part of B. Tech 8th semester Computer Engineering. Work submitted in this report is our original work and has not been submitted elsewhere for any other Degree/Diploma.

Digvijay Bahree	Amit Yadav	Akash Gupta
110600	110269	110250

(Students, B. Tech, Computer Engineering)
(NIT-Kurukshetra)

I hereby certify that the above mentioned major project was carried out under my guidance during even semester 2013-14.

`

R.M. Sharma
Assistant Professor
Department of Computer Engineering

Contents

1. Introduction

1.1 Problem Statement

Data loss on a personal computer due to hard-disk failure or any other reason costs high for a user. Therefore to avoid this, we usually backup our data on external drive and maintain a consistent backup time to time. This creates an extra overhead for a user.

This creates a need of software to remove the overhead of data backup by using a costly hard drive. This software removes these limitations from our present day systems.

1.2 Overview of the Solution

MineCloud uses the free cloud storage provided by vendor (Google Drive) to backup the data of hard-disk. Basically software creates a folder on desktop and if you put any file/folder in it then it automatically syncs the data between your system and backend (cloud).

To maintain a backup of say 150GB: software uses 10 accounts of google drive storage. MineCloud maintains a variable for each of the drive accounts to determine available memory to provide such an abstraction level.

Therefore, if storage for current account overflows; then it splits the file according to size constraints, maintain a mapping in a database and store the splitted ones in different accounts at back-end (cloud).

1.3 Acronyms and Abbreviations

JSON:	Java Script Object Notation
GACLP:	Google API Client Library For Python
API:	Application Programming Interface
OAuth:	Open Authorization

1.4 Features

- Upload file on cloud without thinking of file size and vendor.
- Download file to your machine.
- Delete file.
- Organize data by creating folders.
- Restore system in case of hard-disk failure.

2. Phase 1: Requirements Analysis

Description: Idea finalization, feasibility check, generation of the SRS

The first task was to freeze an idea. Various factors including practical applicability, innovation of approach, feasibility in given time constraints and availability of infrastructure were considered so as to make the choice. Out of 2 major options presented and discussed at length, the final decision was made to make a cloud app.

The team agreed to proceed with the development of *MineCloud* as:

- Practical Applicability: It has wide practical applicability as the problem statement it addressed (specified in section 1.1) is a very commonly found problem.
- Innovation of Approach: No such software is present in the market.Cloud as backup was a new thought.In addition to this availability of users files on remote machines(smartphone, tablets) can be possible now.
- Feasibility Check: One of the primary concerns restraining the start of design phase was the technical feasibility of the idea. To be sure on it a thorough research was carried out on drive api.

Following the idea finalization and feasibility check, the next milestone accomplished was the formal documentation of the entire idea in form of an SRS.

2.1 Excerpts from the SRS

2.1.1 Problem Statement: (same as section 1.1)

2.1.2 Overview of the Solution: (same as section 1.2)

2.1.3 Technologies used: Python 2.7, Drive api v2, OAuth 2.0, JSON

2.1.4 Assumptions:

It is assumed user has a main gmail account.Now it wants to explore cloud potential.

2.1.5 External Dependencies:

❖ OAuth 2.0 Version: 2.0 Auth. Library	An API which is capable to give an offline access to drive accounts.
❖ GACLP Version:2.0 Client Library	Client library for python to interact with the drive.

2.1.6 Software Requirements:

Desktop PC with OS installed, python 2.7,internet connection.

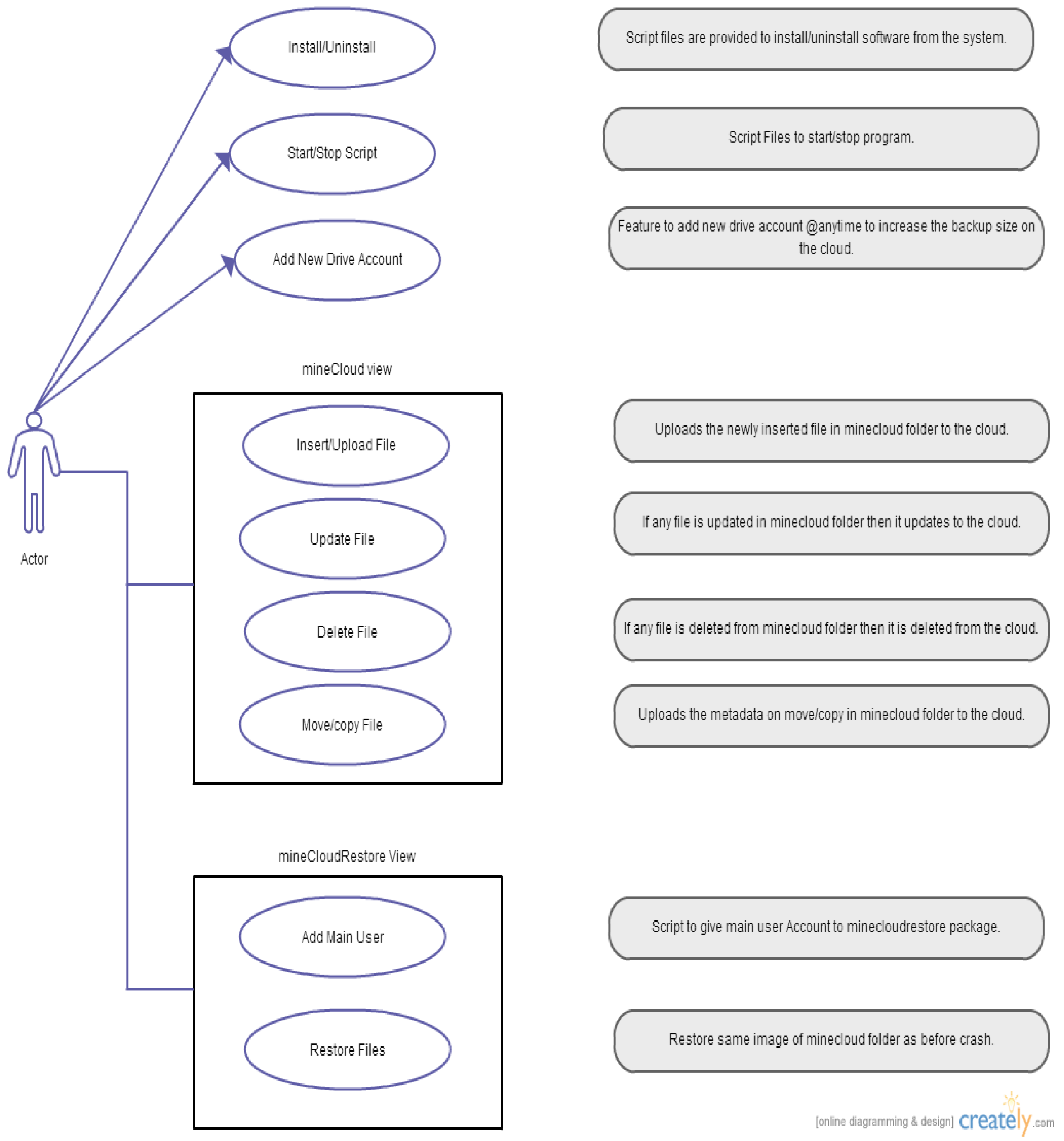
2.1.7 Hardware Requirements (optional):

mobile capable of USB tethering,
Wifi adapter, Wifi access point
LAN card, internet access through Ethernet cable

2.1.8 Constraints:

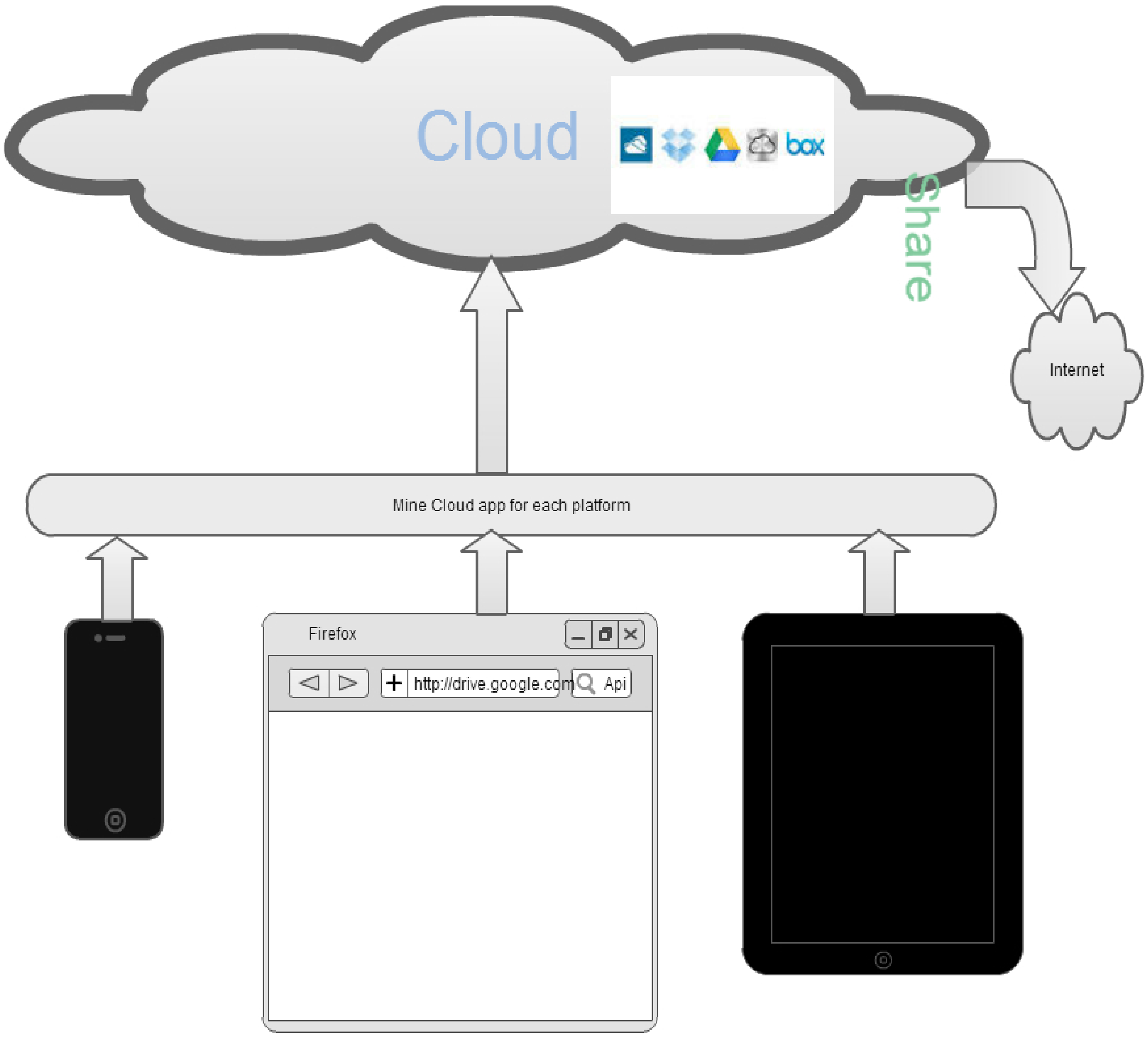
- In our current implementation, single filesize can’t be greater than 10GB.
- Currently it can work for google drive storage only.

2.1.9 Use Case Diagram:


Install/Uninstall
Start/Stop Script
Add New Drive Account
Actor
mineCloud view
Insert/Upload File
Update File
Delete File
Move/copy File
mineCloudRestore View
Add Main User
Restore Files
Script files are provided to install/uninstall software from the system.
Script Files to start/stop program.
Feature to add new drive account @anytime to increase the backup size on the cloud.
Uploads the newly inserted file in minecloud folder to the cloud.
If any file is updated in minecloud folder then it updates to the cloud.
If any file is deleted from minecloud folder then it is deleted from the cloud.
Uploads the metadata on move/copy in minecloud folder to the cloud.
Script to give main user Account to minecloudrestore package.
Restore same image of minecloud folder as before crash.
[online diagramming & design] creately.com

2.1.10 Architectural Design

To attain the required objective our software adds one abstract layer between application layer and different cloud storage users: thus acting as an intermediate for forwarding requests to different user accounts depending upon properties set in the software. Pictorial view of same is:



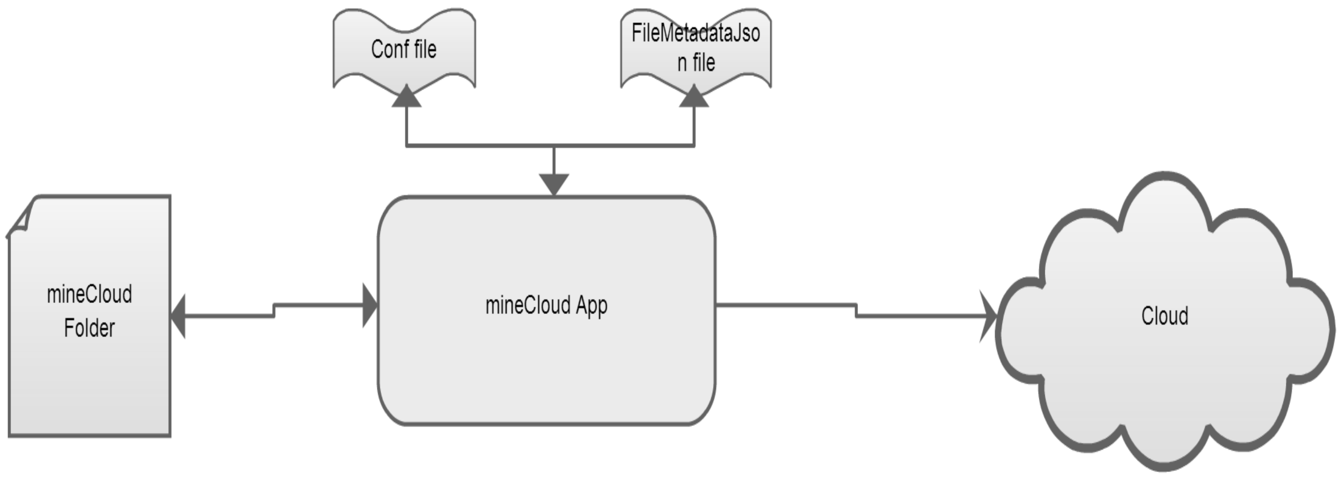
3. Phase 2: Software Design

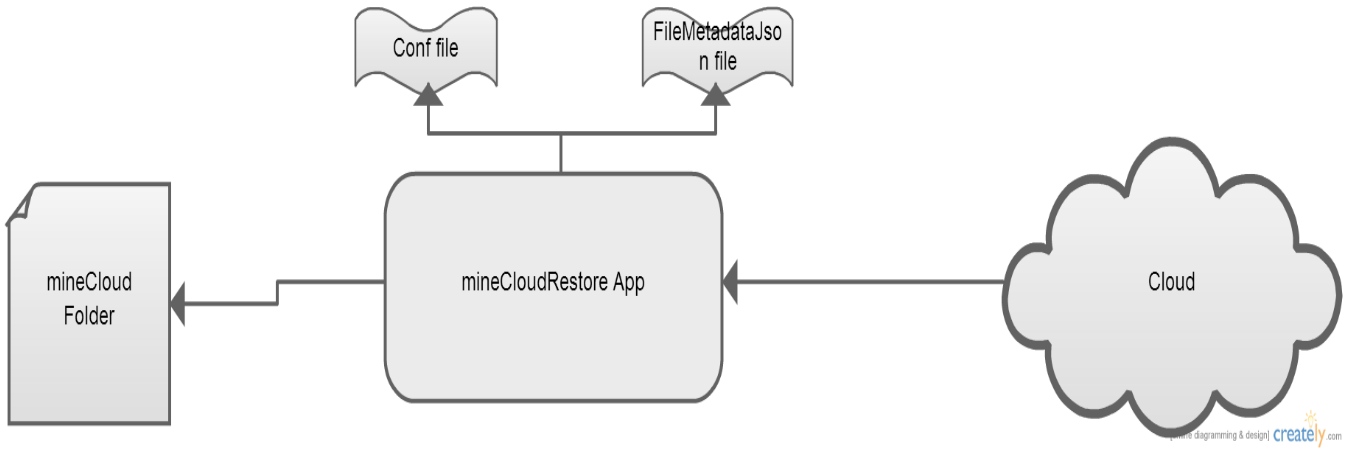
Description: SDD

An SDD (Software Design Document) was created as a part of this phase of our software lifecycle. SDD involved Data Flow Diagrams.

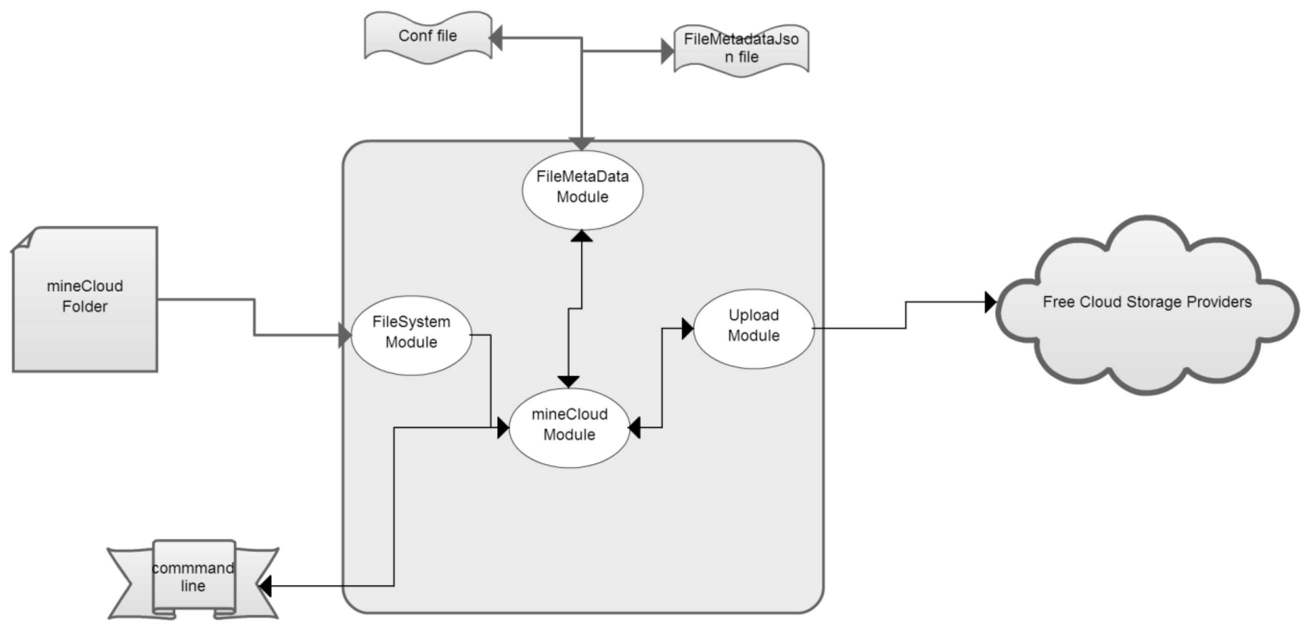
3.1 Excerpts from the SDD

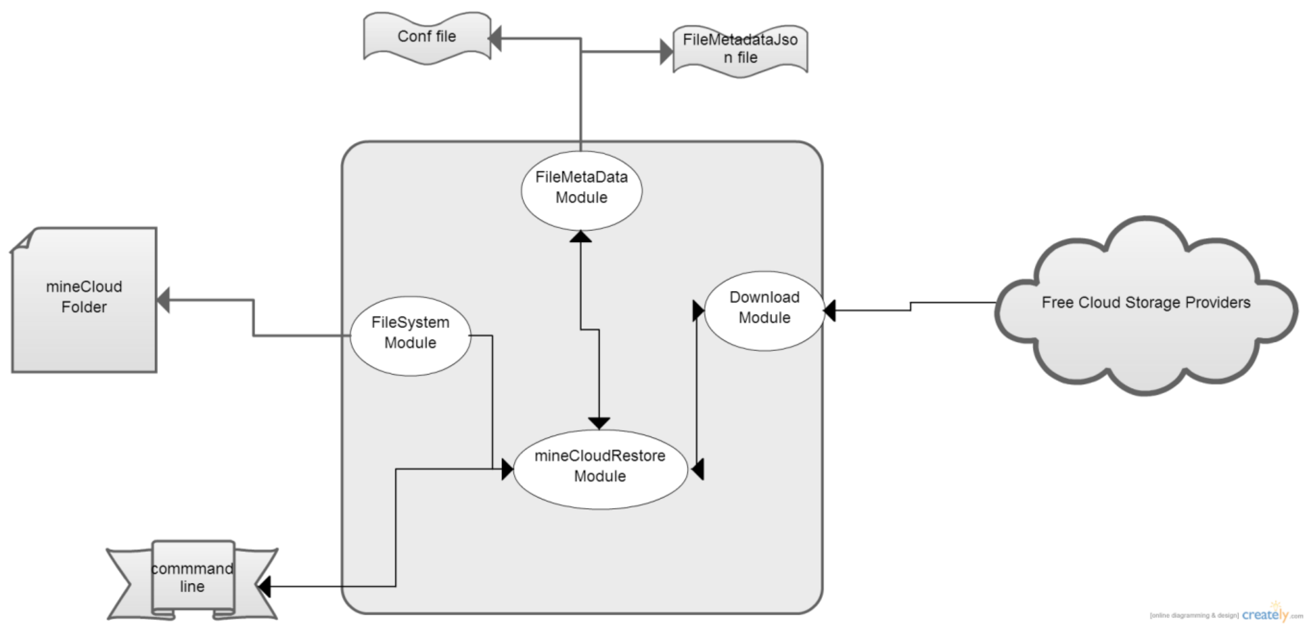
3.1.2 DFD level 0:





3.1.3 DFD level 1:





4. Phase 3: Software Development

Description: Module wise development task allocation and integration

Entire development of the software was done in this phase. The development strictly followed the guidelines and conventions of the modular programming. Various modules were identified and developed independently and sometimes in parallel.
Incremental approach was followed.

Major modules of the software (as also indicated in the DFD level 1 in section 3.1.2) are:

- minecloud module
- minecloudrestore module
- upload module
- download module
- FileMetaData module
- FileSys module

4.1 Division of work:

The workload of this phase was divided amongst the team members in the following way:

- Digijay Bahree was assigned to minecloud Module and minecloudrestore Module.
- Akash Gupta was assigned to upload and download module.
- Amit Yadav was assigned to FileMetaData and FileSys module.

4.2 Implementation

Implementation is done using 3 step process.

1. OAuth 2.0 Authorization:

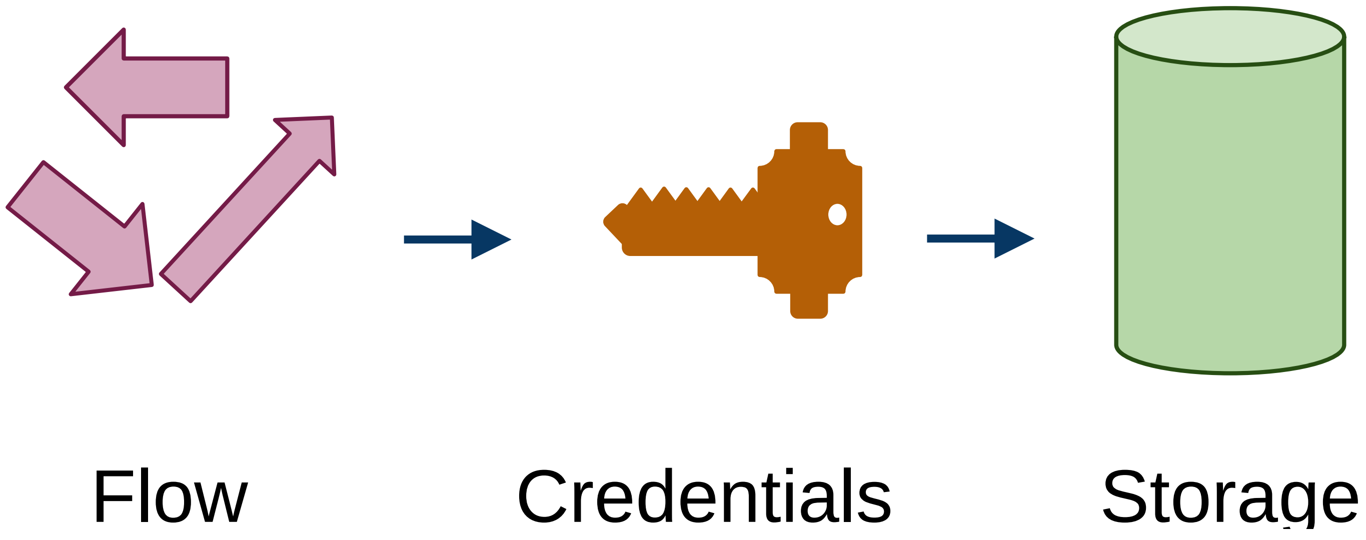
   It is trying to solve a tricky problem.If you, the developer, are building an application. And your users have data in another service that your application needs to function such as their tasks list, or their photos. how do you go about getting it?

   You could ask the user for their name and password.But then the user has given your application access to all their data on that service. That's not safe. Don't do that.The user's name and password are like keys to their digital kingdom, you should never ask for them.
   What we really want is a special key, one that only allows access to a limited set of data in the API.A special key that the User can let the App acquire and use without the use of their name and password.But for that to work, everyone has to confirm that everyone else is who they say they are.Which is where the complexity comes from.

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |        Authorization Grant &  +---------------+
|        |--(C)--- Client Credentials -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

It's actually a little more complicated than even that, because that special key can change over time to keep things secure.
Now that we know what OAuth 2.0 looks like, how does it work in the Google API Client for Python?

The key is held in a Credentials object.All the steps needed to go through getting Credentials is in a Flow object.And finally, because keys can change over time there is a Storage object for storing and retrieving keys.



2. Upload(sync file with cloud)

   Upload module provide functionalities like uploading a new file inserted into the minecloud folder, if file content changed(file updated) or any file is deleted from minecloud folder; then it syncs this minecloud image with cloud.And finally its metadata is maintained using FileMetaData module.

   It asks the user to add a new user using addnewuser module;if there is not enough space on drive accounts(cloud) to upload a file.

3. Restore minecloud folder

   Firstly it requires authorization from main user account.Then it restores the same image as before crash to the minecloud folder using mainly minecloudrestore and download module.

4.3 Algorithm Used:

Files are uploaded with their relative path as metadata on the drive.Folders are not uploaded on the drive.Therefore file hierarchy you see on the desktop is maintained in the backend.This file hierarchy can be maintained using when we restore the system using relative path(metadata) attached to each of the files.

Whenever a new file is inserted into the minecloud folder, system uploads it upon one of the drive accounts named numerically in conf folder i.e. 1,2,3 etc depending upon the file size and drive storage remaining in that account.
As soon as file is uploaded a dictionary object dict[filepath]={fileid,modify_time,drive number}: where filepath is the key,fileid is the drive file id returned while uploading that file,modify_time is modification time of that file and drive number to which it is uploaded,is created.This dictionary object is then dumped into a json file.
Maintaing this json tells beforehand that a particular file is uploaded or not.And if modify_time is changed then file is updated also on the drive.
To delete a file from drive garbage entries from json file is deleted and same is deleted from the cloud.

5. Phase 4: Software Quality Assurance

Description: Modular and Integration testing

The various features developed were tested against parameters of the features they promised and the quality of the same.

Test Cases executed were:

- Software is able to resume syncing if internet connection is suddenly unlpugged.
- Restoring also can resume from previous image in case of internet failure.
- Running individual modules to check if they work fine in isolation.

We found that the software was not that huge and preferred not to go for a formal Unit testing. Yet integration testing was an important part to ensure whether it works properly in certain situations.

5.1 Tool Used:

Python inbuilt feature to module test is used.

5.2 Results:

Results of the integration testing and module testing were in unison with the program behavior.

6. Phase 5: Installation

Below are the steps to be followed by an end-user to install and start using our product on his system.

- Step 1: Before running the software you must be sure about that these packages are installed in your computer- python 2.7(or higher) and drive api client library for python

  Installation instruction for python2.7 :
  1. Go to www.python.org/downloads/ using browser.
  2. Follow the instructions to install python depending upon os installed in the system.

  Installation instruction for drive api client library for python through terminal :
  1. Run command:
     pip install --upgrade google-api-python-client

- Step 2: Running the software
  Type this command in terminal for starting the software
  ```
  python minecloud.py
  ```

7. Future Scope

- The *future plan* is to extend this commutative storage to different platform like mobile, tablets. Thus all of the user data could be accessed on any machine.

8. References

- [1] Drive API
  Title: Drive API
  Source: https://developers.google.com/drive/v2/reference/
- [2] GACLP
  Title: Google api client library for python
  Link: https://developers.google.com/api-client-library/python/guide/
- [3] Wikipedia