Building a simplified replica of Dropbox

1. Note : This Is a Python Flask project that need to build in Flask framework only. Plus, you are only allowed to use the Google Datastore for database and firebase for only authentication(Oauth 2.0).

Frameworks/libraries that are permitted: Bootstrap Flask Flask-WTF React.js (frontend) Flask-datepicker WTForms JSONForms JQuery

Frameworks/libraries that are not permitted: Django TkInter Firebase python MongoDB SQLite SQLAlchemy

2.

Project brief : first before implementing it contains very important information In this project you will be building a simplified replica of the Dropbox cloud service. You will be required to have storage available for users where they can create arbitrary directory structures and can upload files and download files from the service. There should also be some access to some basic file sharing between accounts. It is recommended that the structure of the project is closely followed with the brackets as you will need to get the directory structure working first before you will be able to upload and download files or attempt the later brackets after this.

3 Submission and Penalties

You are required to submit two separate components to the Moodle

• An archive containing your complete Google App Engine Python project. The accepted archive formats are: zip, rar, 7z, tar.gz, tar.bz2, tar.xz. The use of any other archive format will incur a 10% penalty before grading.

• A PDF containing documentation of your code. If you do not provide documentation your code will not be marked. Copying and pasting code into a PDF does not count as documentation.

• The use of libraries outside the SDK will incur a 20% penalty before grading. You have all you need in the standard SDK. I shouldn’t have to figure out how to install and use an external library to get your app to work.

You are also required to submit as part of your archive a working Git repository.

• When I unpack your archive there should be a .git directory as part of it.

• This should be a fully working local git archive. It should not require access to a remote repository.

• You are not permitted to upload your work to Github, Gitlab, or any other publicly visible git repository (project will be marked as a zero if it is)

• There must be a minimum of seven commits in the git repository, one per completed bracket.

Very Important: Take note of the grade brackets listed below. These are meant to be completed in order. If you skip a bracket or do not complete a bracket following brackets will not be considered for marking. You should be well capable of producing strong and generally robust software by now. For example if there are six brackets and you fail the third one, then the fourth, fifth, and sixth brackets will 2 not be marked. Documentation brackets will be treated separately from Coding brackets.

I want robust bug free code that also validates all user input to make sure it is sensible in nature. Also note that the percentage listed after the bracket is the maximum mark you can obtain if you complete that many brackets without error. Everything in all brackets is mandatory.

Coding Brackets (80%)

1. Bracket 1 (10%)

• Write the shell of an application that has a working login/logout service.

• Create models of a user, directory, and file using appropriate datatypes. – If a user logs in for the first time a user model should be created for them. – a default root directory (path of /) should also be created for this user.

1. Bracket 2 (20%)

• Add the ability for a user to create a directory (Bracket failure if the same directory name is allowed in the current directory)

• Add the ability for a user to delete a directory (Bracket failure if the wrong directory is removed)

1. Bracket 3 (30%)

• Add the ability to change into a directory

• Add the ability to go up a directory with the special entry (path of ../)

• If a user is in their root directory don’t display (path of ../)

1. Bracket 4 (40%)

• Allow the user to upload a file to the current directory and store it in the cloud storage bucket. This should not overwrite a file that is already there. If a file is already there ask the user if they want to overwrite it.

• Allow the user to delete a file from the current directory (Bracket failure if the wrong file is deleted.

• Allow the user to download a file from the current directory to the local machine.

5. Bracket 5 (50%)

• Prevent the deletion of a directory that still has files remaining. Provide an appropriate warning to the user.

• Prevent the deletion of a directory that still has directories remaining. Provide an appropriate warning to the user.

1. Bracket 6 (60%)

• Add in the ability to detect duplicate files in the current directory. Display the files that match

• Add in the ability to detect duplicate files in a user’s entire dropbox. Display the files and paths that match 5

1. Bracket 7 (70%)

• Add the ability to share files read only between multiple user accounts

1. Bracket 8 (80%)

• UI design: Well thought out UI design that is intuitive and easy to use.