Team 18
CS 3240
Design Document
Client Server Communication

1.

Name: CS Commie (Client-Server Communication)
Responsibilities:
Uploading files
Downloading files
User registration
User authentication
Important data:
Manages files as they are uploaded or downloaded from server
Connects to database where user information is stored
Lifecycle:
A web server framework runs continuously on the server. The client application then submits HTTP requests (GET and POST) to this server when necessary. This means that when the application finds that files need to be updated, it can send the appropriate requests to the server, and doesn't need to be continuously interacting with the sever.

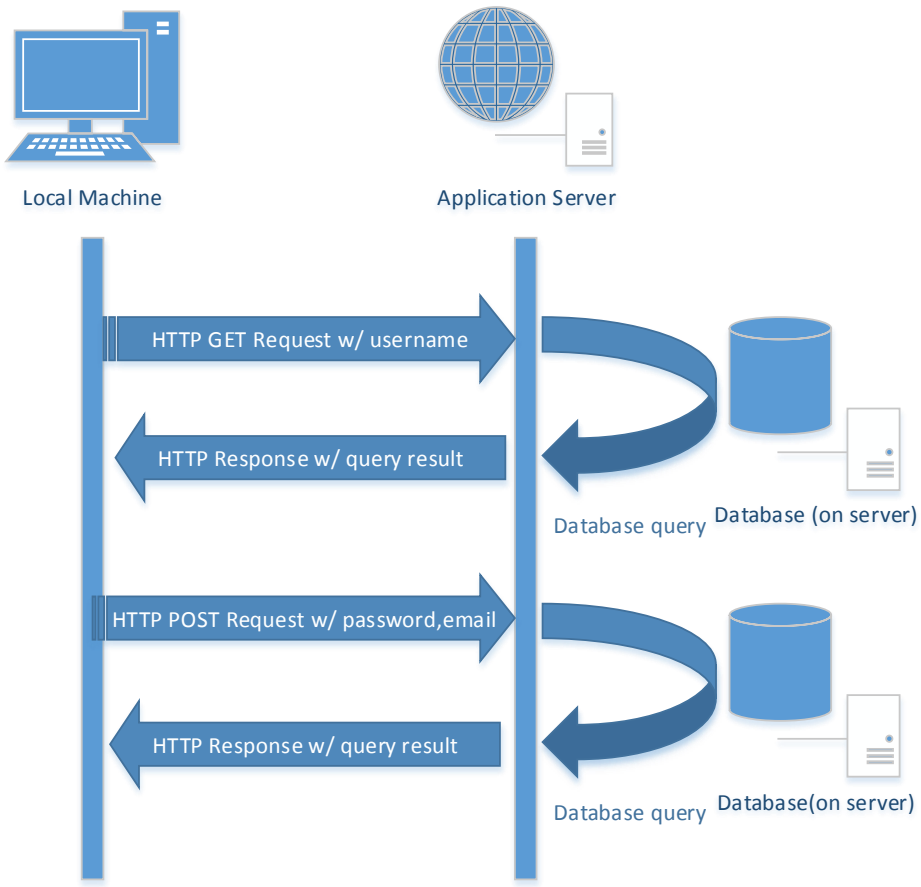
2.

Technology:
Built using Flask and Requests. Flask is used on the server to create application routes for the client to access. The Requests framework is used on the client to build HTTP requests that interact with the server.

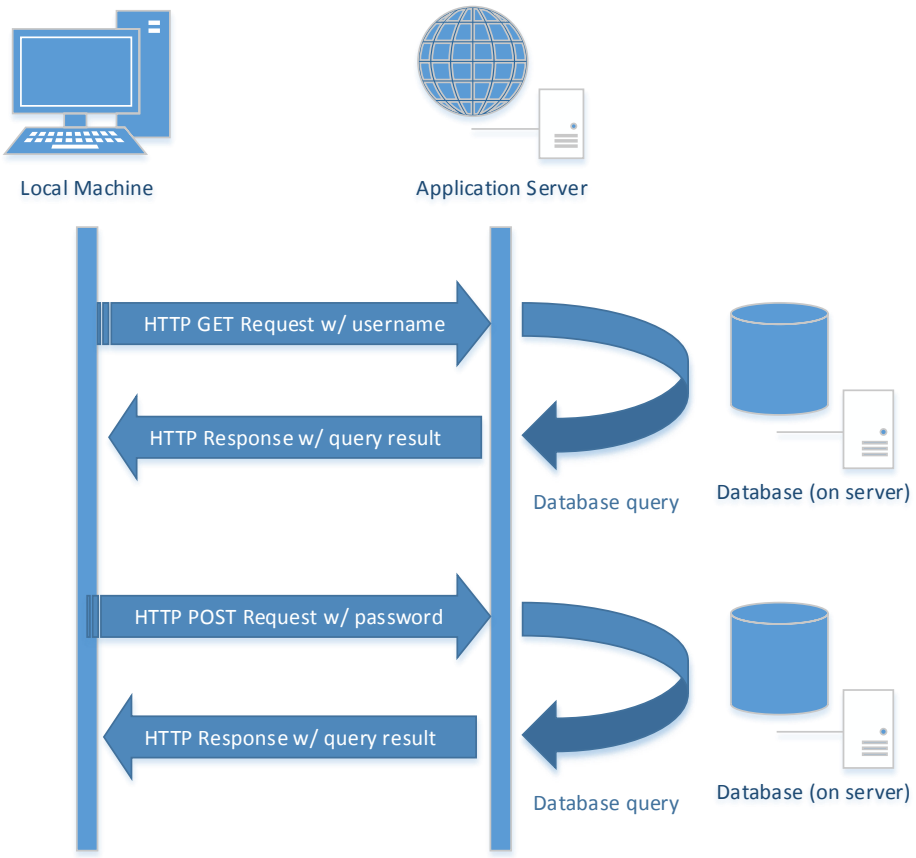
3.

Interactions:

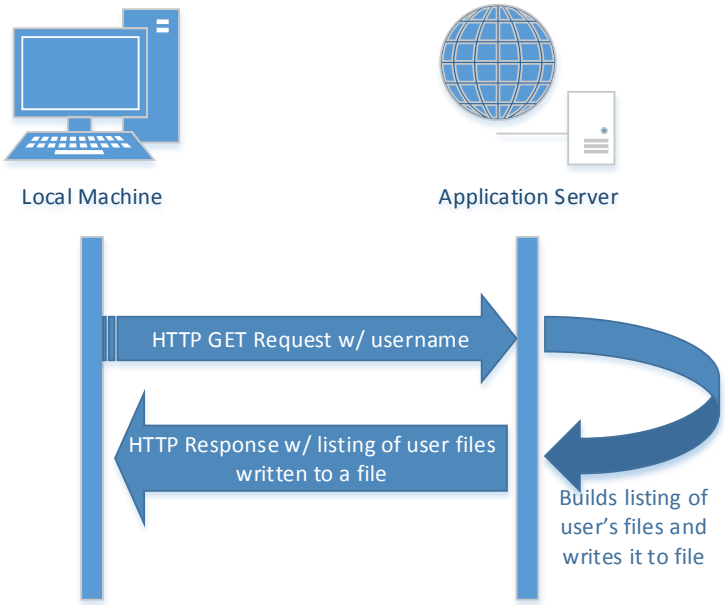
1) User Registration
    i. The user will need to register for the application from the local machine client. This will allow them to log onto the server and store files there. Once the username is entered, it is sent to the server via an HTTP GET request. The server then takes this username and checks if it is in the user database. For registering a user, they will not be a database. The application will then prompt them for a password and email address, and use this information to create a new entry in the user database for them. This information can be passed to the server through a POST request.
    ii. See next page for UML diagram



2) User Authentication
   i. The user will need to be able to log into the application on the server, to store securely store and access their files. Once the username is entered, it is sent to the server via an HTTP GET request. The server then takes this username and checks if it is in the user database. If the user is in the database, the local machine client will prompt them for their password, send this to the server using a POST request and check if the password provided matches the password for this user in the database. The result of this will be in the HTTP response the server sends back to the local machine.
   ii. UML Diagram on next page

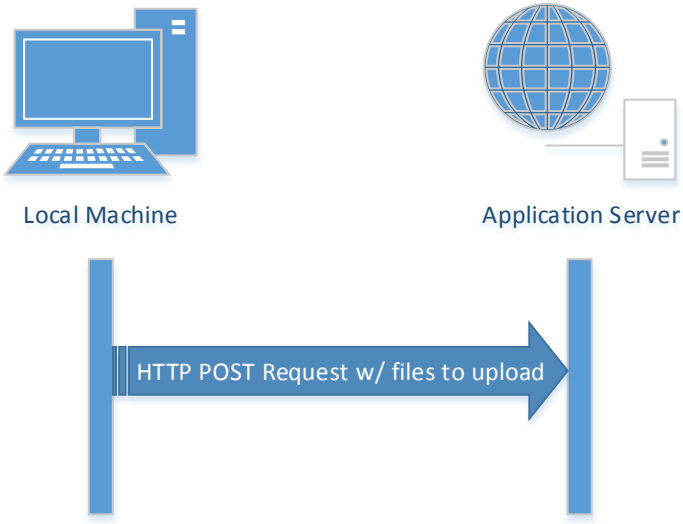


3) Create listing of user files on server
    i. The local machine client will need to be able to download a listing of the files stored on the server, and some information associated with these files so file updates can be detected. This detection is done by a separate process running on the local machine. Then, an HTTP GET request is made using the Python module *requests*, and the server responds with a file containing a listing of the user's files stored on the server.
    ii.



4) File Upload
    i. The local machine client will need to be able to upload files and file changes to the server. HTTP POST requests (as provided by the python module *requests*) work well for this, if the server can process the incoming request properly. In our application, we use Flask to capture and respond to incoming requests.

ii.



5) File Download
   i. The local machine client will need to be able to download files from the server. HTTP GET requests (as provided by the python module *requests*) work well for this, if the URL is known.
   ii.