Names: Brian Nguyen, Atul Dhungel
Homework 2 - Individual Project Proposal

Project description:
Our project is to create a database that is a key-value store in C++ with an associated REST server using Microsoft cpp REST SDK library that allows users to use http to interact with the database and use http verbs to store/update key and values. For example, a user could use a post request to post "{"foo": "bar", "key": "2ffaf213"}" to localhost:8000, and that key would then be stored in the database.

Technologies using:
Scale 1-5

Creating database:

- C++
  - Brian: 4
  - Atul: 3

Creating the GUI

- Qt with C++:

Essential parts:
My project won't work if it doesn't have the the data structure to store the data

- The data structures we could use are a hash table or B-tree to store the key of the data and a pointer to where the data is

My project won't work if it doesn't have data persistence (data is stored on disk and on memory)

- We could use fstream to write to a file, but for performance and to do it properly, we have to use the mmap system call

My project won't work if it doesn't have a http server that has handles for post, delete, and put

- To interact with the database, users can use http. Microsoft cpp REST SDK is a simple way to create to a http server as seen here

Backup plan:

- If we cannot get the http server to work, we won't include it, which would only allow us to manipulate the data in the database; users won't be able to use our project.

Outside resources:
None

Architecture:

Frontend:

A GUI written in Qt, which will be able to interact with the backend through the Microsoft cpp REST SDK. The GUI will allow users to interact with the database using buttons, such as post, get, update, and delete. Besides this, there will also be another page that allows users to see the keys and values as two columns.

Backend:

We make our database a key-value store, and to do this, we will use a hashmap to store these key value pairs. Essentially, a file will be used to store the data, and the reason for using a hashmap is that it will allow us to access the data faster than other data structures with O(1) time complexity. *We decided not to create a web application. Instead of using the React + axios library to make api calls, we decided to use Microsoft cpp REST SDK to handle the api requests from and to the frontend.*

Classes:

Database:

- **class** OurDB
  - **methods:**
    - CreateDB(std::string name)
      - Description: creates the database
    - LoadDB(std::string name)
      - Description: loads the database along with the contents of it
    - AddToMem(std::string name)
      - Description: adding the data to disk
    - GetPath()
      - Description: gets the full path to the disk
  - **attributes:**
    - path
      - Description: the full path of where the file is located at
- **class** OurStore()
  - **method:**
    - SetKeyVal(std::string key, std::string val)
      - Description: sets a key and its value
    - GetKeyVal(std::string key)
      - Description: returns the value of the given key
  - **attributes:**
    - store
      - Description: the hash map (or set) that stores the key-value pairs
- **class** User
  - **methods**:
    - SetUsername(std::string name)
      - Description: sets the user's username
    - GetUsername()
      - Description: gets the user's username
    - MakeApiRequest(std::string api_call)
      - Description: make an api call, which will in turn call the corresponding Microsoft cpp REST SDK method

        - **attributes**
            - Name
    - **class** Superuser (derived from the Base class, User)
        - **methods**:
            - RemoveUser(std::string name)
                - Description: removes a User
            - DeleteDB()
                - Description: deletes the database

GUI:

- **class** Button
    - **methods**:
        - LoadQt(std::string repo_name, std::string local_name, User curr_user)
- **class** Page
    - **Methods**
        - LoadPage()

Inheritance

The Superuser class will be the derived class of the base class, User.

Design patterns

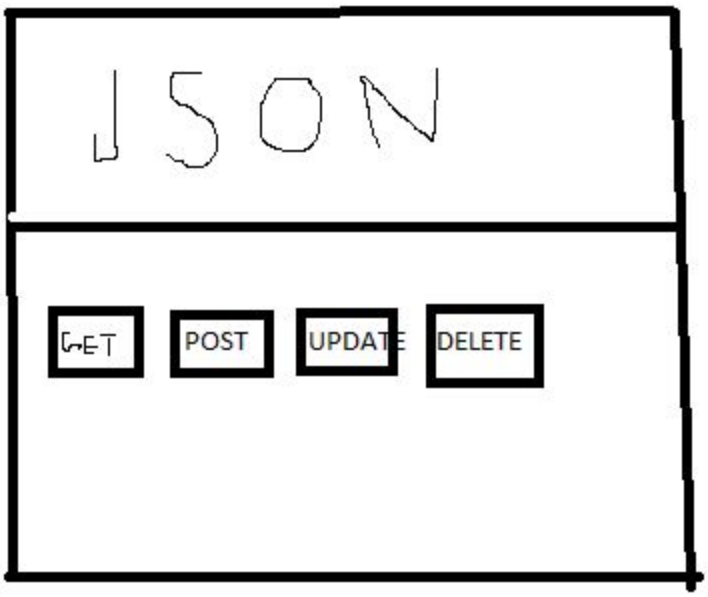
We will use Iterators and Prototype.

Database

Upon creating the database, we will store various information like name, password, dates, etc. This project will essentially use the GUI to interact with our database.

GUI's user interface:

The user will be able to click on the API request and it will return the appropriate data as JSON or success message where JSON is marked.



Detailed plan:

Week 7:
- Brian: Work on creating the Database class' backbone
- Atul: Work on implementing the methods of the database class

Week 8:
- Brian: Unit tests for the Database class' methods
- Atul: Setting up CI (Travis CI); set up the http server using Microsoft cpp REST SDK

Week 9:
- Brian: Have a working Database class
- Atul: Have a working Database class

Week 10:
- Brian: Start working on developing the User class
- Atul: Start working on the Superuser class

Week 11:
- Brian: Have a working User class and a working Superuser class
- Atul: Start integrating and working on the GUI

Week 12:
- Brian: Continue with GUI/integration
- Atul: Continue with GUI/integration

Week 13:
- Brian: Unit tests for the GUI and finishing it up
- Atul: Unit tests for the GUI and finishing it up

Week 14:

Brian: Integration + debugging
Atul: Integration + debugging

Week 15:

Brian: Final touches + improve UI
Atul: Final touches + improve UI

Week 16:

Brian: Work on presentation
Atul: Work on presentation

When HW2 is due, we will have a detailed class structure.
When HW3 is due, we will have a working database class.
When HW4 is due, we will have a working user and superuser class.
When HW5 is due, we will have a checkpoint/integration for the above 3 done.
GUI and finished product will be done by the final.

How we plan to stay engaged in this course:

We will need to learn Qt to build our GUI, we will need to learn the various types of design, and we will need to utilize office hours and seek TA/instructor advice on various obstacles we will face.