5-2 Milestone Four: Enhancement Three: Databases

Brandon Goucher

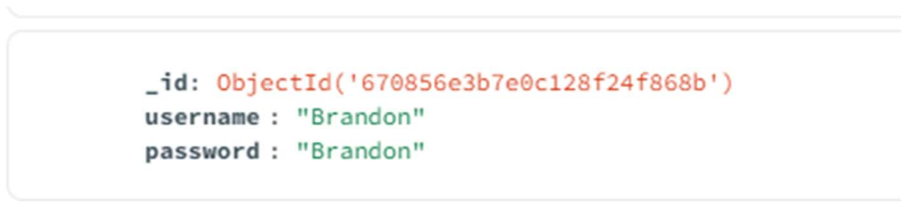
CS 499 Computer Science Capstone

Prof Brooke

10/18/2024

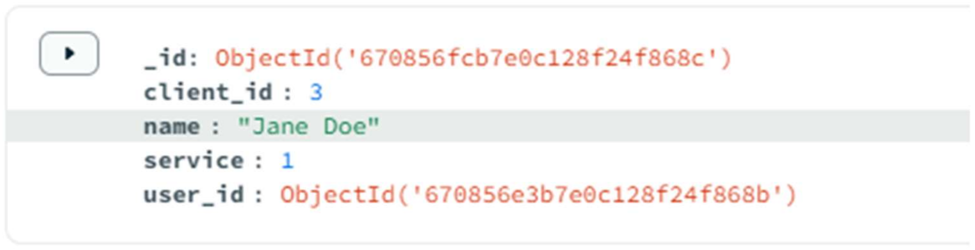
Introduction:

Similarly to Software Engineering and Design, I used my CS 410 Software Reverse Engineering file project. I wanted to change this since the only data being used and accessed was about a 6-line HashMap that could be added if changed, but I felt like it needed to be changed a lot to be considered good. My goal for this milestone was to change those six lines into a Mongo Database that will include a user and the clients associated with that user.



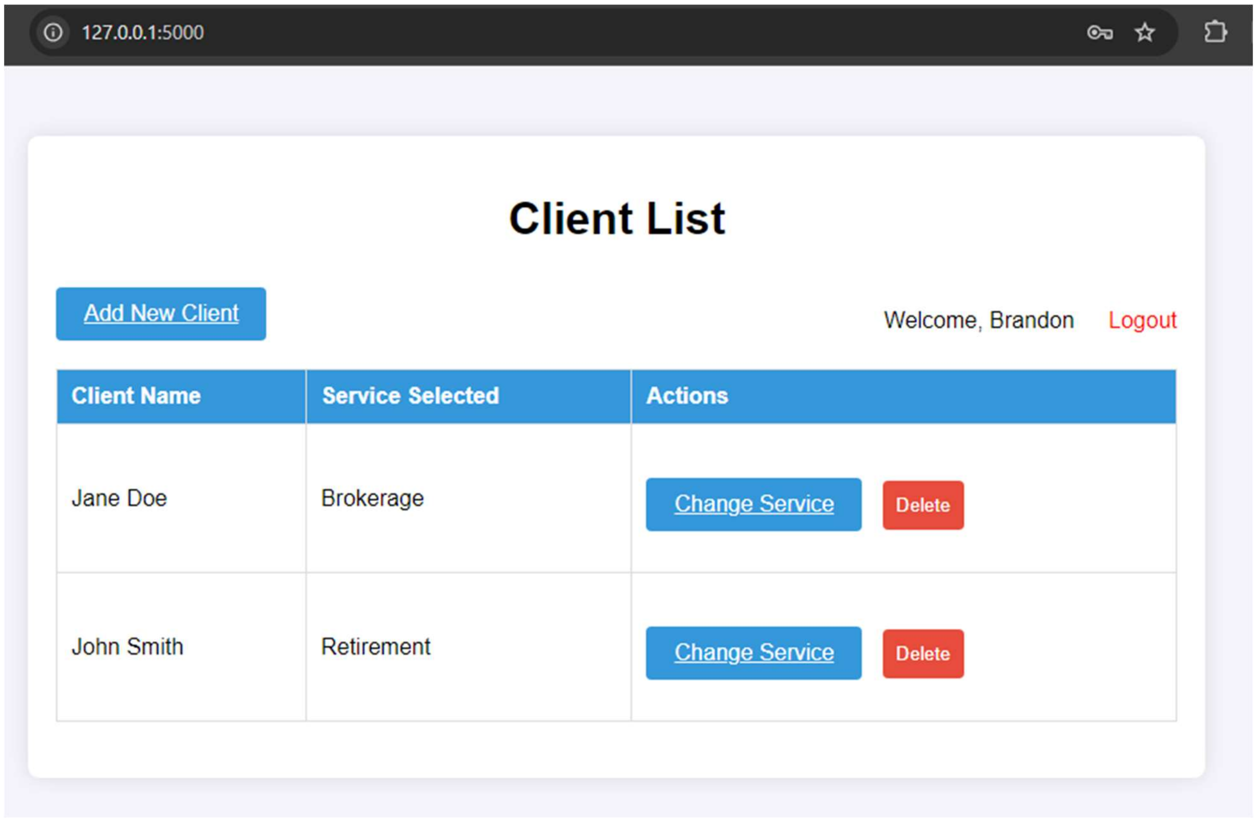
```
_id: ObjectId('670856e3b7e0c128f24f868b')
username : "Brandon"
password : "Brandon"
```

The user part of this database was pretty simple; it required the user to type in a username and password. The _id is automatically generated using MongoDB, which is very convenient for the next part of this database.



```
_id: ObjectId('670856fcb7e0c128f24f868c')
client_id : 3
name : "Jane Doe"
service : 1
user_id : ObjectId('670856e3b7e0c128f24f868b')
```

Other than the client's name and service, most client aspects are also auto-generated. To be able to connect these collections together, I needed a primary key. So, noticing that MongoDB autogenerates that _id, I connected each client with their respective users by setting up a session each time a user signs in, grabs the _id of that user, and applies it to the variable user_id. From there, when we go to the HTML that displays all of the clients, it checks to see if the client's user_id is the same as the current user in session, and if it is, then the client is displayed.



127.0.0.1:5000

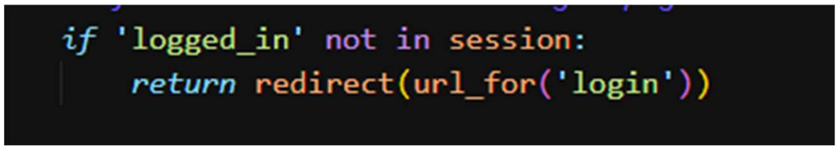
Client List

Add New Client   Welcome, Brandon   Logout

Client Name	Service Selected	Actions
Jane Doe	Brokerage	Change Service Delete
John Smith	Retirement	Change Service Delete

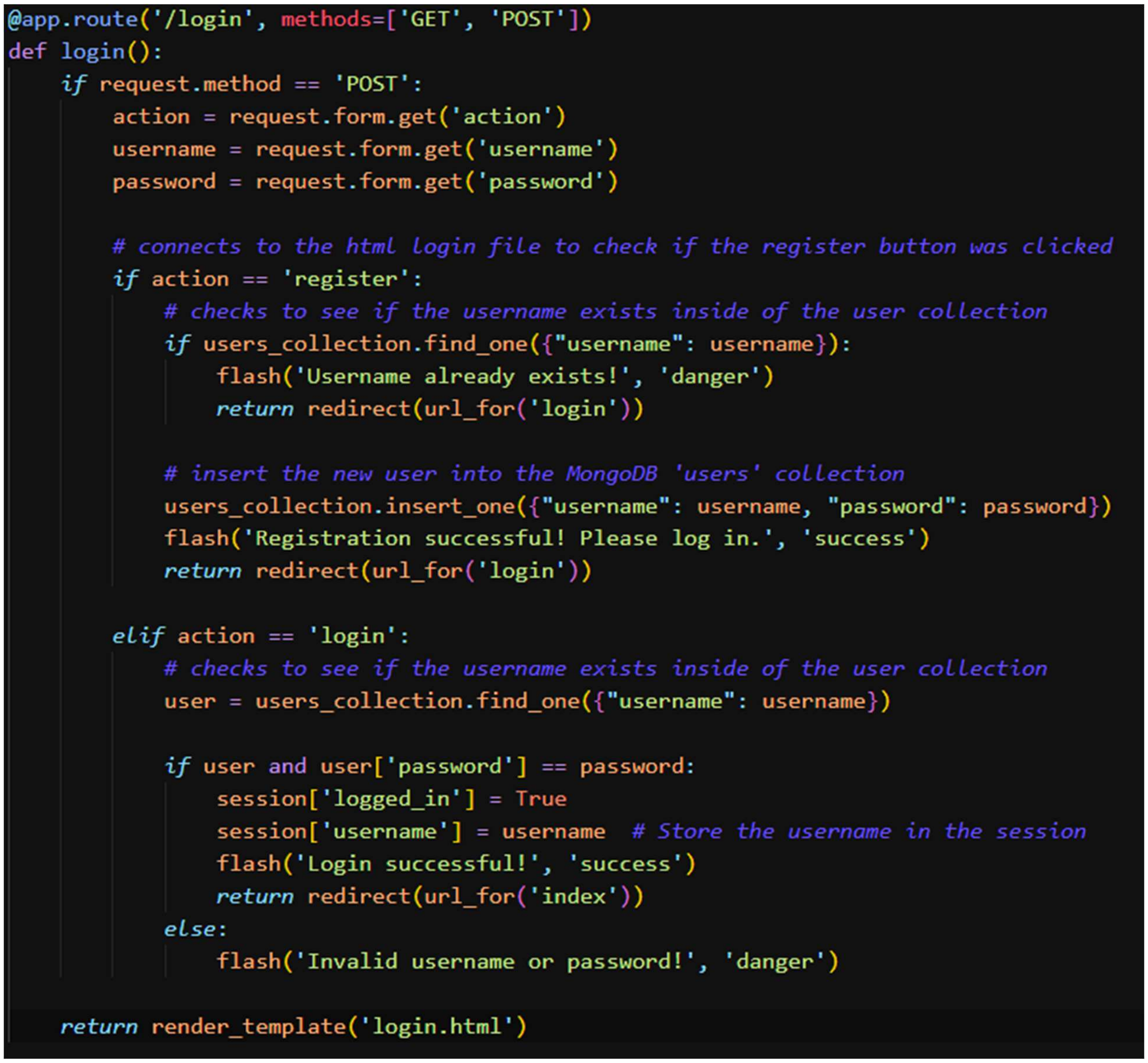
As we can see here, in the top right, it shows the current username, “Brandon,” and then displays the clients associated with me. As shown in the previous image, Jane Doe is associated with me, so it is displayed here with the Brokerage account service, which is labeled as a 1 inside of the database.

Authentication for security:



```
if 'logged_in' not in session:
    return redirect(url_for('login'))
```

Check to see if the user is already in an active session; if not, then redirect them to the login page.



```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        action = request.form.get('action')
        username = request.form.get('username')
        password = request.form.get('password')

        # connects to the html login file to check if the register button was clicked
        if action == 'register':
            # checks to see if the username exists inside of the user collection
            if users_collection.find_one({"username": username}):
                flash('Username already exists!', 'danger')
                return redirect(url_for('login'))

            # insert the new user into the MongoDB 'users' collection
            users_collection.insert_one({"username": username, "password": password})
            flash('Registration successful! Please log in.', 'success')
            return redirect(url_for('login'))

        elif action == 'login':
            # checks to see if the username exists inside of the user collection
            user = users_collection.find_one({"username": username})

            if user and user['password'] == password:
                session['logged_in'] = True
                session['username'] = username  # Store the username in the session
                flash('Login successful!', 'success')
                return redirect(url_for('index'))
            else:
                flash('Invalid username or password!', 'danger')

    return render_template('login.html')
```

There are two different aspects here; we have the register side, which doesn't require any authentication other than making sure that the username doesn't already exist, and if it does, then it will show a flash message saying that the username already exists.

Otherwise, the user will have to log in, first making sure that the user does exist by trying to find them inside of the database, if found then it takes the password that was typed in and double checks to make sure that the password inside of the database is the same. If it is, then they are redirected to the main page. If not, it displays a flash message saying that either the username or password is invalid. This is important because we don’t want people to know what part of the sign-in process is wrong just in case there is a person trying to guess the password of a user over and over again.

The next two course outcomes I have met are:

- Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources

Implementing user authentication into the system was one of my main priorities since the C++ file was very lackluster, allowing almost any user to sign in since it was entirely hard coded. Also, using the _id from the user collection and then using that as the user_id for the clients' collection, allowing only that specific person to view their clients, was another level of security added. In a financial space, we want to have the most security possible. So, having only the client's respective users able to see and adjust their accounts is necessary.

- Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science

The comments on each of the files explain exactly what is happening and why. This is my way of ensuring this activity could become more collaborative, enabling anyone in the computer science community to use it and know what is going on without running any of the code. Also, the simplicity of the HTML enables people from all different backgrounds in computer science to look at it and know what to expect when viewing each page.

In the end, my goal was to develop a solid secure database enabling new users to register and old users to be able to sign in to see the clients that are associated with them. As proven by the images above I have perfectly achieved that.