The OO Climbing Gym

Team Members: Colin Murphy and Oscar Delgado

Final State of System

We concluded our project having satisfied nearly every goal we created for ourselves originally. Our project's major feature is a working tkinter python GUI that users can interact with. This GUI is linked to a MongoDB database where we keep a collection of all our users and their specific information, and also a collection of receipts that keeps track of the pricing and gear rentals for members when they enter the gym. This pricing is based firstly on what kind of membership the user has and secondly what gear they want to rent when they enter. Premium members get in for free regardless of gear, regular get a free day pass but pay for gear, and casual members have to pay for everything. We used the Decorator pattern to adjust the final cost based on what gear they rent. Different gear is offered on the Gear Page depending on what type of climber the member is. We have a seperate page for them to change their climber type if they desire, which will change the gear options. When a customer finishes choosing gear they are led to the Check Out page where all their pricing information is displayed including what price everything was and their final total based on their membership.

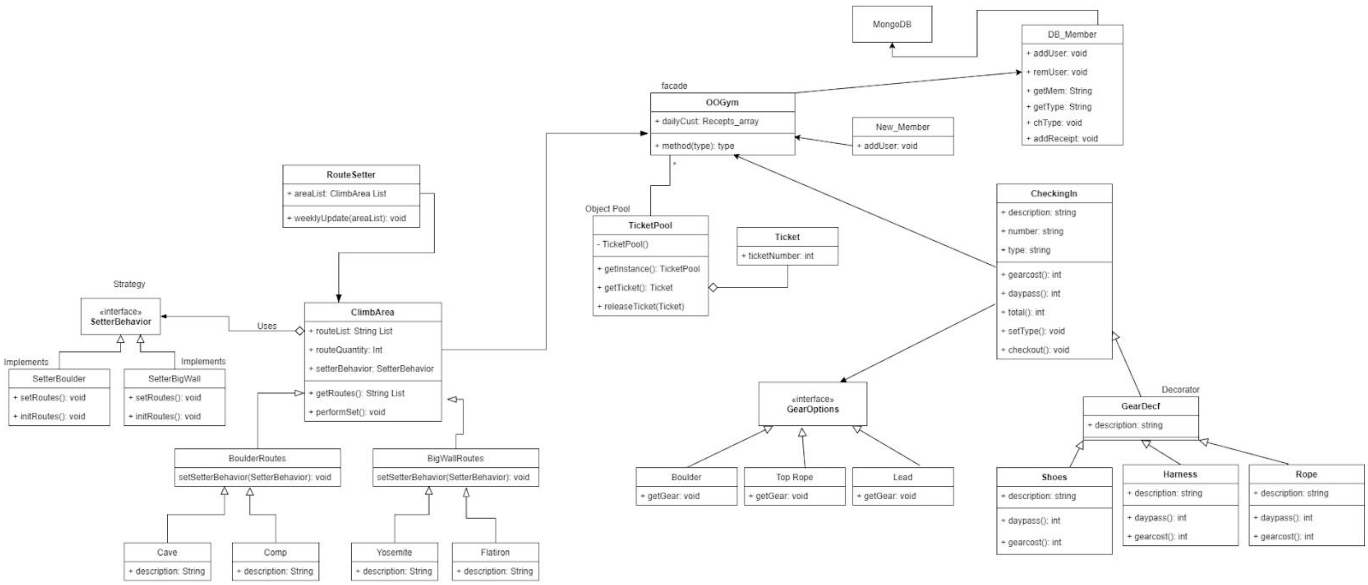
When a user first enters the GUI they have multiple options for what they can do. If they have never been to the gym before they can make an account through the "New Member?" button. When clicked they are led to a page where they can enter all of their information which will be sent and added to our MongoDB database. We also have checks in place to make sure the user fills out every information box and also the waiver. Then they are led to the Gear Page. If a user has already registered with us then from the Main Page they can click on the "Check-In" button to be led to the Check-In page. There they can enter their phone number as a login and if they are in the database they will be recognized in our system and treated accordingly. Using the Object Pool pattern we give out tickets every time someone enters the gym to keep track of capacity. Once the gym hits capacity the system stops any more users from entering. We were originally going to keep track of the inventory of all gear too but we were not able to implement this feature in time. We also concluded that the gym's capacity would account for inventory supply in a realistic setting anyway.

Another thing we implemented was climbing areas and routes in our gym. Using the Strategy pattern we can add new climbing areas to our gym easily and make them either for Bouldering or Big Wall with different features for each. When routes are created the climbing grades follow a distribution that we created so that the range of difficulty is realistic. The areas and routes are initialized at the beginning and can be seen in the GUI when you click on the "Check Out Our Routes!" button. There every area and route is presented to the user, with difficulty listed sequentially. In the Employee page that we created you can update the gym by

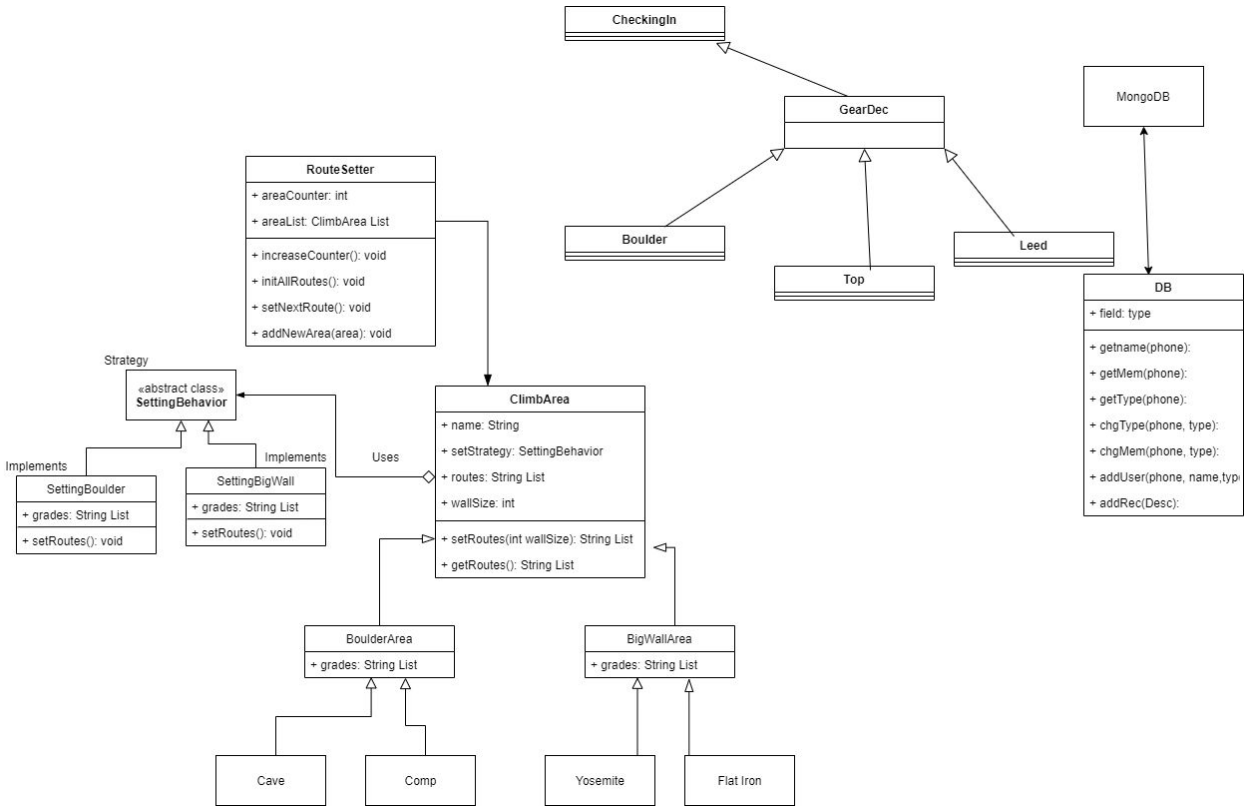
changing the climbing area's routes sequentially, it will display what area was changed and what the old and new routes are.

The other two buttons under the employee page are "Open Gym" and "Close Gym" when you hit Open Gym our system notes what time the gym was opened that day and sends it to the database. When you press Close Gym a final receipt is sent to the database that has a list of all the orders that have been completed since the GUI has been open and then it closes the GUI.

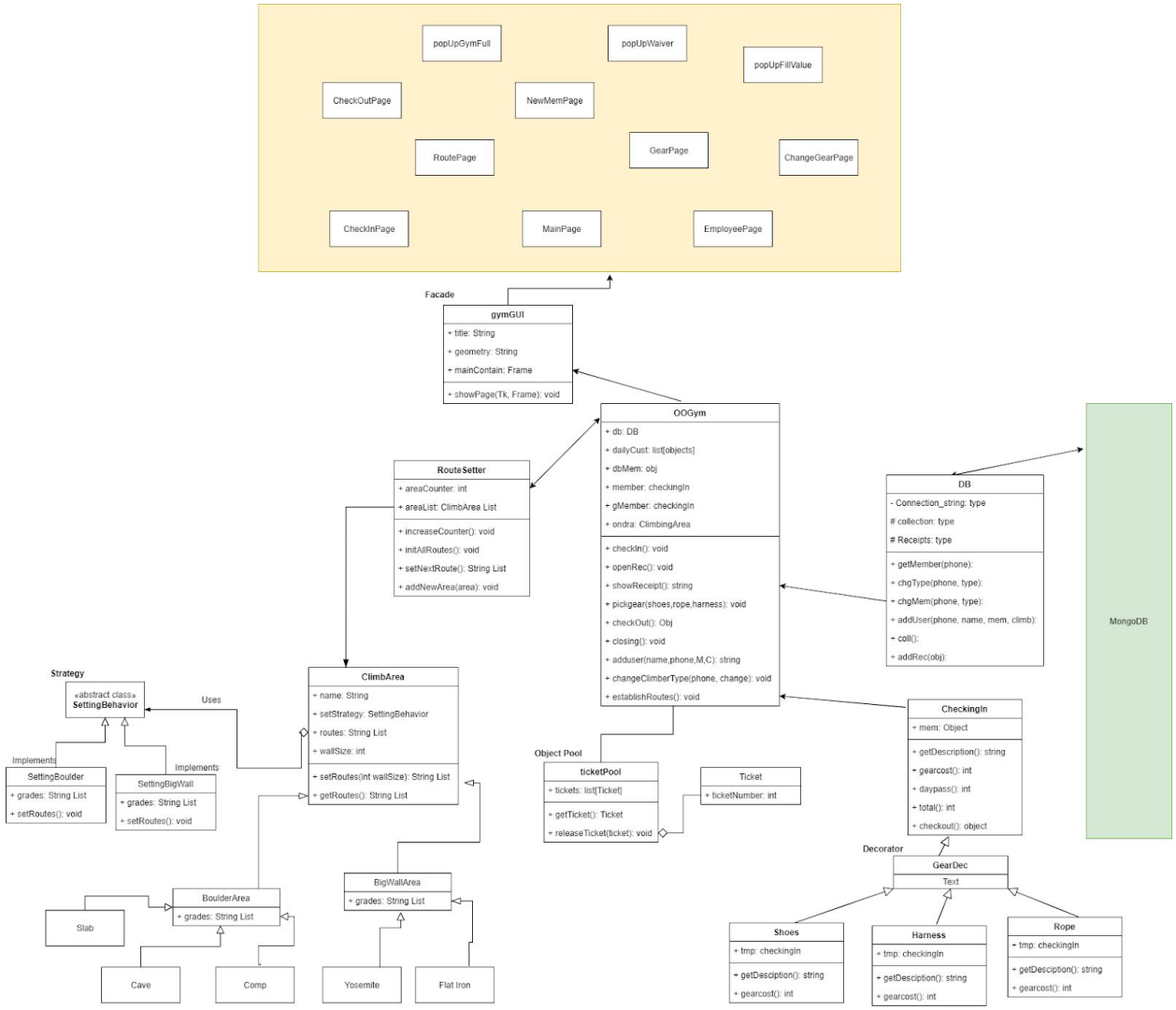
Final Class Diagram and Comparison



Project 4 class diagram (https://app.diagrams.net/#G1sY6ZO3b21-9vEC9Keo-fq64459aj-GkU)



Project 5 class diagram
(https://app.diagrams.net/#G1o0PmUvqifXHmHCLkhWbA4gvJAiVPpxkS)



Project 6 class diagram
https://app.diagrams.net/#G1OEtosAN7NmihkWFdQld1f4dOMr-LTAlJ

Key Changes:

- Removed gear interface because we had wrongly planned out how it would be implemented
- Removed class of NewMember - instead just had it as a function in OO Gym
- Did not make UML for GUI originally, at the end we decided the best way to represent it was a high level diagram just to show the main pages and the facade pattern
- Some function within classes were changed but the functionality remained very similar
- DB instead of having function for name, type, member we only had return the member object which had all of these
- The OOGym class ended up containing many more things than we had originally anticipated, it is the central hub for our functionality

Third-Party Code vs. Original Code

During the creation of the GUI it was new territory for us so we followed some online examples and tutorials to get us started. The sources for these tutorials are linked below.

https://pythonprogramming.net/change-show-new-frame-tkinter/
https://stackoverflow.com/questions/37068708/how-to-change-font-size-in-ttk-button
https://pythonprogramming.net/tkinter-popup-message-window/

These tutorials were just a starting point and we ended up changing and creating nearly all the code to be our own. The creation and connection of the MongoDB database was actually not looked up because the process of how to create it was learned in other classes. To create code as patterns we followed along with the slides given in class. Some outside sources were used just for clarity, those are listed below:

https://sourcemaking.com/design_patterns/object_pool/python/1
https://medium.com/@sheikhsajid/design-patterns-in-python-part-1-the-strategy-pattern-54b24897233e

The major tools we used in this assignment were MongoDB Atlas and Mongoclient for the database and the tkinter python library for the GUI. The large majority of the code created for this project was created by ourselves.

Statement on the OOAD Process

1. Integrating a Database - this went very well and was fairly simple. We used MongoDB Atlas to create the database and used Mongoclient to connect and send queries. We connected this to a class to make calls a lot easier (encapsulation).
2. Integrating the GUI - This was a very interesting process and definitely a positive part of this project. Once we got the hang of how tkinter worked we were able to quickly make more pages and functionality for our GUI. Connecting the GUI to our code's functions was surprisingly easy, we had very little trouble connecting the GUI to our database. It's also worth noting how object oriented based tkinter is, everything is created as instances of objects and classes. Skills learned in this class were definitely applicable to this part of our project.
3. Making the UML- When making all the UML (the use cases, flow charts, class diagrams) we put a lot of work into it to try to make it as close to our end product. By doing this it made starting to code far more simple and also separating the work and working separately go more smoothly than our other projects. Other times we used to discuss how we wanted to format codes or how to approach it while writing code and this was very time consuming. By doing the discussions

early we didn’t have to discuss too much and were able to write code and do tests on code a lot easier.