Design Patterns

By Group 11

Our group decided to write a program that helps the user determine what restaurant they want to eat at in Lawrence, and the user is able to make an account with the features of rating a restaurant or removing a restaurant. It is sort of a Yelp lite. Our thought process of the design of the creational pattern was similar to a builder pattern. This pattern builds complex object using simple objects. We had a builder class which was the Executive class. The constructor of this class used an account object and an UI object. The aim of this class is to conduct the interaction with the restaurant txt file, the users txt file (for accounts), and a user choosing different menu items from the UI class. It was a builder class that used objects from other classes to finalize the end result. We came to this conclusion that this would be the best way when we planned the type of classes we would need. Since this project was a C++ program, we felt comfortable using an executive class to handle interaction with reading in a file and writing out to a file. A builder pattern also allowed us to work independently on the other classes that helped ‘build’ up the executive class. For example, we first finished those files such as the Restaurant class, Account class, and UI class before executive could be completed. Since executive was the builder class, it had the majority of the code and contained the “recipe” for putting together the final product. Overall, the builder pattern was an easy choice for this program because of its ability to build complex objects.