Part B: Essay Question

In the real world, software systems are almost never simple implementations. Even a system that appears simple has layers of design and thought baked into the implementation in order to achieve maintainability, robustness, reduced coupling, and the desired functionality. Design diagrams allow the systems analysts and designers to plan their route through the problem domain in advance and catch many inconsistencies and problems before they arise. However, some of the diagrams can grow to be complex tasks in and of themselves - therefore, a number of diagrams are built to specifically target aspects of the problem and understand them before implementation.

Component Diagrams: used to model physical aspects of the system. Without knowing the environment or hardware being developed for, mistakes will inevitably occur.

Deployment Diagrams: used to describe how the application / system / software will be spread across the physical hardware of the system. Responsibilities for what shows / does what.

Design Class Diagram: used to model attributes, accessibility, flow, and methods of classes that will be used in the system. This is important for programming the system, and will be referenced directly during programing.

Sequence Diagram: used to expand and model a use case scenario. Sequence diagrams go hand in hand with Design Class Diagrams for identifying the needs of the classes, as well as what methods / information will be passed between objects in the system.

Design State Machine Diagrams: used to show the states that a system or use case can be in. These help the developer model for eventualities before they're accidently discovered, and provide proper state control for the system.

Package Diagrams: A **package diagram** depicts the dependencies between the packages that make up a model. A package is a collection of related classes that work together to achieve a specific form of functionality.

<u>Part C: Reflection</u>

When I compare my opinions of these assignments to other taking the course, I notice an inverse trend. For the most part, the intensity of the assignments seems to grow the further everyone proceeds for the course, but for me, the work has become a bit more comfortable. With my background in software engineering, system design is a familiar task, including UML Design Class Diagrams and Sequence Diagrams (This isn't to say I'm amazing at them - just that I've encountered them a number of times before).

This is the assignment where all the previous work begins to take structure. Building up on all our past diagrams, you can see where holes in an old understanding of the problem may have existed before. The synergy between the Sequence Diagram and the Design Class Diagram really worked to show - many of the methods for the controllers and reservation class objects became flushed out as the sequence diagram showed the kinds of communication needed between objects.

The quality of the diagrams is becoming more and more important to me on a personal level as well. Although I usually work within Agile methodologies as a developer, should I ever need to implement a system to be integrated into another (or work over a distance) I can see how these diagrams are key to guiding implementation according to users vision for the project.

My approach to solving the problem: I used the Class Diagrams from assignment 2 and began to add design class elements to each instance, adding controllers and more as I walked through the use cases in my mind. Then I began on the sequence diagram, and realized the elements I needed to carry back to my Design Class Diagram to make it more complete and robust.

All in all, the assignment was a fun exercise in applying the textbooks examples. I look forward to assignment 5.